Deliverable D3.3

SWForum Research Roadmap – v1

Editor(s):	Leire Orue-Echevarria
Responsible Partner:	TECNALIA
Status-Version:	Final – v1.0
Date:	22.12.2021
Distribution level (CO, PU):	PU

Project Number:	GA 957044
Project Title:	SWForum.eu

Title of Deliverable:	D3.3 – SWForum Research Roadmap – v1
Due Date of Delivery to the EC	31.102.2021

Workpackage responsible for the Deliverable:	WP3 - Sustainable SWForum.eu and Research & Innovation Roadmaps
Editor(s):	Leire Orue-Echevarria (TECNALIA)
Contributor(s):	Juncal Alonso (TECNALIA), Galia Novakova Nedeltcheva (Polimi), Elisabetta Di Nitto (Polimi)
Reviewer(s):	David Wallom, UOXF
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3, WP4

Abstract:	There will be three iterations of this roadmap. The first iteration will include the scoring methodology valorising the software technology, digital infrastructures and cybersecurity research and innovation roadmap and the necessary steps for its implementation and the initial set of research priorities identified, including also the input from the cross fertilization workshops, the industry, as well as from the desktop research performed and the landscape reports from task 3.1. This roadmap will be made available for the open (online) consultation. The second iteration will contain the final set of prioritized research areas, as well as a set of recommendations and actions on research priorities for Horizon Europe and Digital Europe, which will be presented in different events and workshops in order to gather more feedback. Interim and working versions of the roadmaps will be presented in the cross-fertilization workshops in order to foster discussions and gather input and feedback from the whole software community. The final version will include the received feedback and will be ready for publication. This is the result of Task 3.2.
Keyword List:	Research roadmap, literature analysis, literature analysis, context analysis, research challenges
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	16.07.2021	TOC	TECNALIA
v0.2	13.12.2021	(Continuous) contributions from participating partners	TECNALIA, Polimi
V0.3	14.12.2021	Sent for internal QA review	TECNALIA
V1.0	22.12.2021	Addressed comments from the internal QA review and prepared it for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

ACM CCS	ACM Computing Classification System
CSA	Coordination and Support Action
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
IDE	Integrated Development Environment
KPI	Key Performance Indicator
NLP	Natural Language Processing
OSS	Open Source Software
ROI	return on investment
RQ	Research Question
SDLC	Software Development Lifecycle
SW	Software
SWForum.eu	European forum of the software research community

Executive Summary

This document is the first of a series of three documents, whose aim is to provide policymakers with research priorities identified in the field of software technologies, cybersecurity and digital infrastructure.

To achieve this, a roadmap methodology has been defined, structured in three steps, namely identification of research topics, classification and scoring, consultation and further analysis. The outcome of these activities will be a set of recommendations. This document focuses solely on the identification of research topics.

To achieve that, a multi-sourced analysis has been used. The main sources have been a literature review, the landscape reports and surveys carried out as part of SWForum, the SWForum workshops and a desktop research of current on-going and past European initiatives. From this analysis, a set of initial findings are reported, namely six. These initial findings respond to the research questions: which are the research trends, gaps and challenges and which are the research topics uncovered or covered in a limited way in the literature. These research challenges are described using the same structure explaining the main research challenge, and the impacts at societal, business and technology level that this challenge would bring when resolved as well as from where it comes.

The next versions of this document will include a scoring methodology, more detailed research topics discussed with the SWForum constituency, and the final set of research challenges, prioritized, which will be delivered to the European Commission as recommendations.

1 Introduction

1.1 About this deliverable

This document is the first of a series of three documents that aims to present research challenges in the field of software technologies, including open-source software, cybersecurity and digital infrastructures.

While initially the goal of this document aimed at presenting a scoring and classification methodology, the final methodology used implied a reordering of the activities, becoming that activity the second one. Hence, the goal for this iteration has been in the identification of research challenges and topics. The scoring methodology and the application of such will be presented in subsequent version.

For the identification of research topics, SWForum follows a multi-sourced methodology, where the input from surveys, the landscape report, current European initiatives, the SWForum workshops as well as a literature review has been considered. The main outcome of this document is a set of research challenges, which in posterior documents can be improved, updated, scored and prioritized, so recommendations can be later on provided to the European Commission.

1.2 Document structure

The document is structured as follows.

Section 2 presents the approach and methodology followed for the research roadmap.

Section 3 provides an analysis of the current context where SWForum revolves. This analysis includes past and current research projects in the last 3 years, an in-depth literature review of research venues on the topics of SWForum, complemented with inputs from previous SWForum deliverables, namely the landscape report and the cross-fertilization workshop report.

Section 4 presents the initial findings of the research challenges, trends and gaps, in a structured way. They are the result of the analysis performed in section 3.

Section 5 contains the conclusions and future work.

Appendix 1 shows the details of the research venues analysed.

Appendix 2 presents the details of the European research activities considered.

Appendix 3 includes the in-depth information of the ACM Computing Classification System (CCS) used as the baseline taxonomy in SWForum.

2 Road mapping approach and methodology

This report is the first version of the SWForum research roadmap, with future versions coming in D3.4 (M24) and D3.5 (M30). The main aim of this set of reports is to provide contributions to the European research roadmap in software technologies, including open source software, digital infrastructure and cybersecurity, taking into consideration different input sources, such as the landscape reports [1], surveys [1], results from the SWForum workshops [2] and an analysis of academic venues. The views from different stakeholders such as the industry, practitioners and academia are included in this analysis.

Figure 1. Multi-source analysis in SWForum

The methodology followed in SWForum for the road mapping can be organized in three steps, as seen in Figure 2 and described below. This methodology has already been used in other actions such as HUB4CLOUD [3] for the main purpose, which demonstrates the repeatability and the scientific soundness of the approach.:

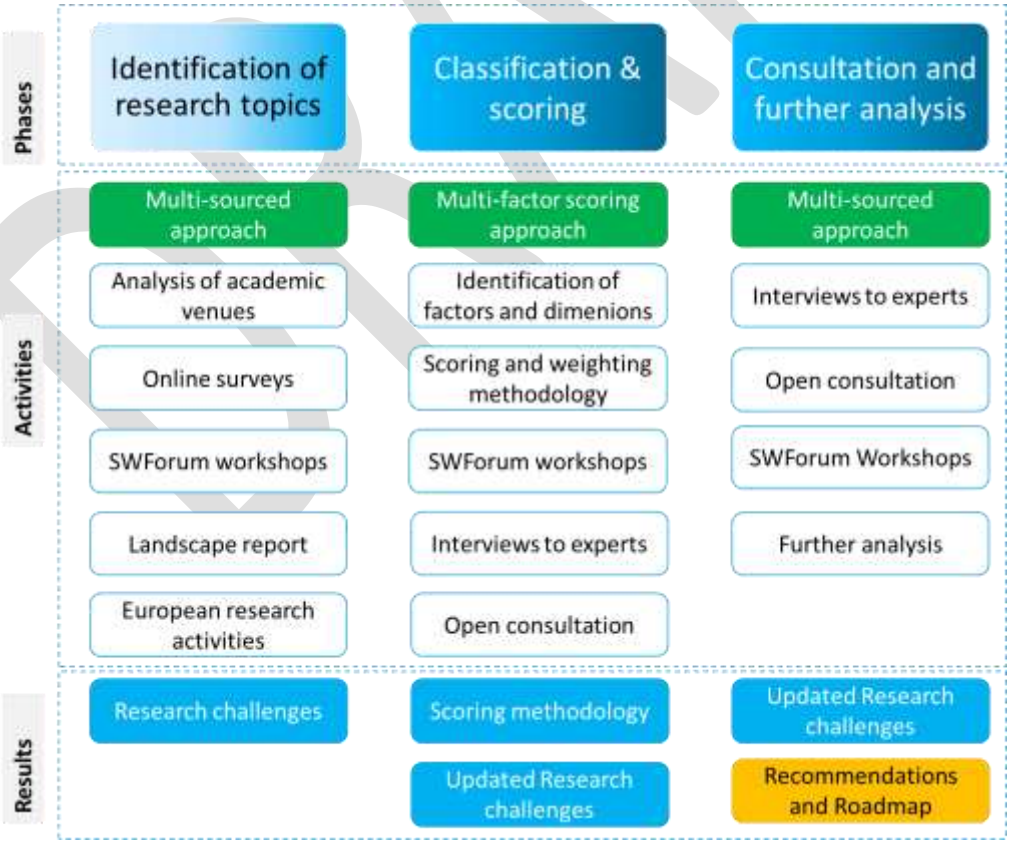

Figure 2. Methodology followed in SWForum for the roadmapping (adapted and extended from [3]). The current document focuses on the phase "Identification of research topics", hence the different shading in the colours

Next, a brief description of the followed methodology is presented:

1. **Identification of research challenges from various sources,** with a clear definition of the research questions under evaluation in order to limit the scope (section 3.1 of the current document)
    a. Analysis of publications from the most relevant academic venues (journals and conferences) (section 3.1 of the current document). The search was carried out between June – September 2021. The results and details of the venues can be found in Appendix 1.
    b. Identification of current and past European Research activities in the field of software technologies, digital infrastructures and cybersecurity (section 3.3 of the current document).
    c. Analysis of the landscape report [1], which included surveys as well as an analysis of existing initiatives (section 3.4 of the current document).
    d. Analysis of the results of the SWForum workshops [2], especially the second one, devoted to research challenges (section 3.5 of the current document).

    The outcome of this phase is this report, the first version of the research challenges, trends and gaps.

2. **Classification and scoring:** The classification and scoring methodology (see figure below) aims at providing a first prioritization of the research challenges identified in the previous step. This prioritization will be initially based on a scoring methodology that will be developed taking into consideration several factors such as the timeframe, the impact, the added value for Europe, and more. These factors will be also given a weight taking as input the opinion of subject matter experts. The third step is the scoring, where also input coming from various stakeholders will be included, apart from the initial prioritization based on the scoring methodology. The results will be represented in a graphical manner so that they can be seen easily at a glance. It is important to note that thanks to this methodology it can happen that the software challenges identified in the previous step need to be updated.

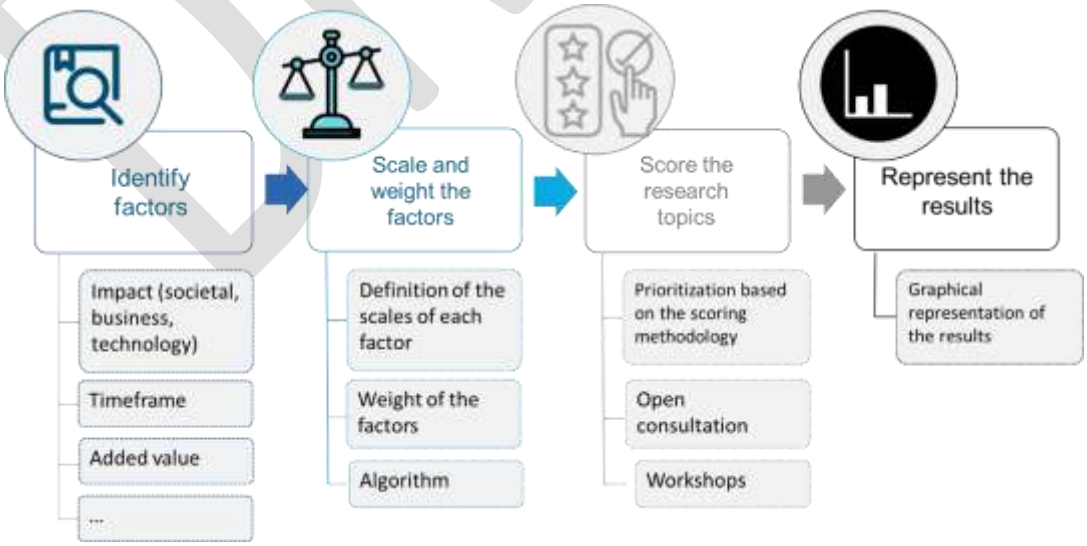

Figure 3. Multi-factor scoring methodology phases (adapted and extended from [3])

The outcome of this phase will be a scoring and classification methodology as well as a more detailed set of research challenges.

3. **Consultation and further and analysis:** The initial classification performed will be shared with the constituency through various means such as SWForum workshops, online surveys, interviews, and as such, the final update to the research challenges will be performed and these derived into recommendations for the European Commission.
   The outcome of this phase will be a set of detailed research challenges, discussed with stakeholders as well as a set of recommendations for the European Commission to include as research topics in the upcoming workprogrammes.

This deliverable (D3.3) reports the work done for the first pillar in Figure 2, namely "Identification of research topics".

3 Context analysis

3.1 Analysis of research venues

3.1.1 Research questions

When performing an analysis of academic and research venues, it is important to establish clear research questions. These research questions help establish the scope of the searched venues. In the case of SWForum the research questions (RQ) are as follows:

RQ1: Identify trends, gaps and challenges in software engineering and cybersecurity in:

- The top 20 academic venues in software and systems engineering
- The 5 most cited papers from these top 20 venues
- In a timeframe of 2016 – 2021

RQ2: Identify research topics uncovered or covered in a limited way in:

- The top 20 academic venues in software and systems engineering
- The 5 most cited papers from these top 20 venues
- In a timeframe of 2016 – 2021

Due to the large scope of software engineering and cybersecurity topics, a traditional systematic literature review (SLR) using methodologies like PRISMA [4] where keywords are used to search for the most relevant publications has proven to be not useful in the scope of SWForum. The returned number of publications was extremely high and the added value with respect to the methodology finally followed and described is rather small.

3.1.2 Methodology

The analysis of the inputs from the academia has been implemented following the next approach:

Figure 4. Methodology followed in SWForum for the analysis of academic venues. (adapted from [3])

The description of the methodology is as follows:

1. **Phase 1 - Selection of academic venues:** The academic venues (journals and conferences) were selected by looking into the list of the top publications on Google

scholar under the category of "Engineering & Computer Science > Software Systems" [1]. The timeframe for this search was June – September 2021.

	Publication	h5-index	h5-median
1.	ACM/IEEE International Conference on Software Engineering	77	116
2.	Journal of Systems and Software	62	94
3.	IEEE Transactions on Software Engineering	59	96
4.	Information and Software Technology	59	78
5.	ACM SIGSOFT International Symposium on Foundations of Software Engineering	57	76
6.	Empirical Software Engineering	52	90
7.	IEEE Software	48	85
8.	Mining Software Repositories	46	64
9.	IEEE/ACM International Conference on Automated Software Engineering (ASE)	45	89
10.	International Conference on Software Analysis, Evolution, and Reengineering (SANER)	44	61
11.	Software & Systems Modeling	42	57
12.	International Symposium on Software Testing and Analysis	40	59
13.	IEEE International Conference on Software Maintenance and Evolution	33	49
14.	Software: Practice and Experience	31	53
15.	International Journal on Software Tools for Technology Transfer	28	44
16.	IEEE Annual Computer Software and Applications Conference (COMPSAC)	27	49
17.	Software Quality Journal	27	41
18.	IEEE International Conference on Software Testing, Verification and Validation (ICST)	27	35
19.	International Symposium on Empirical Software Engineering and Measurement, ESEM	25	36
20.	ACM Transactions on Software Engineering and Methodology (TOSEM)	24	39

Figure 5. List of the selected academic venues (source: Google Scholar)

2. **Phase 2- Selection of relevant publications:** The criteria to select the publications per venue has been the following:
    a. Selection of publications from the last 5 years (2016-2021). This criterion was selected because of the fast evolution of Software technologies.
    b. Selection of top 5 publications per venue based on the number of citations. The source for citations is Google Scholar.
    c. Free/open access to the publications.

An example of the publications covering a and b is shown in the figure below.

[1] https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_softwaresystems

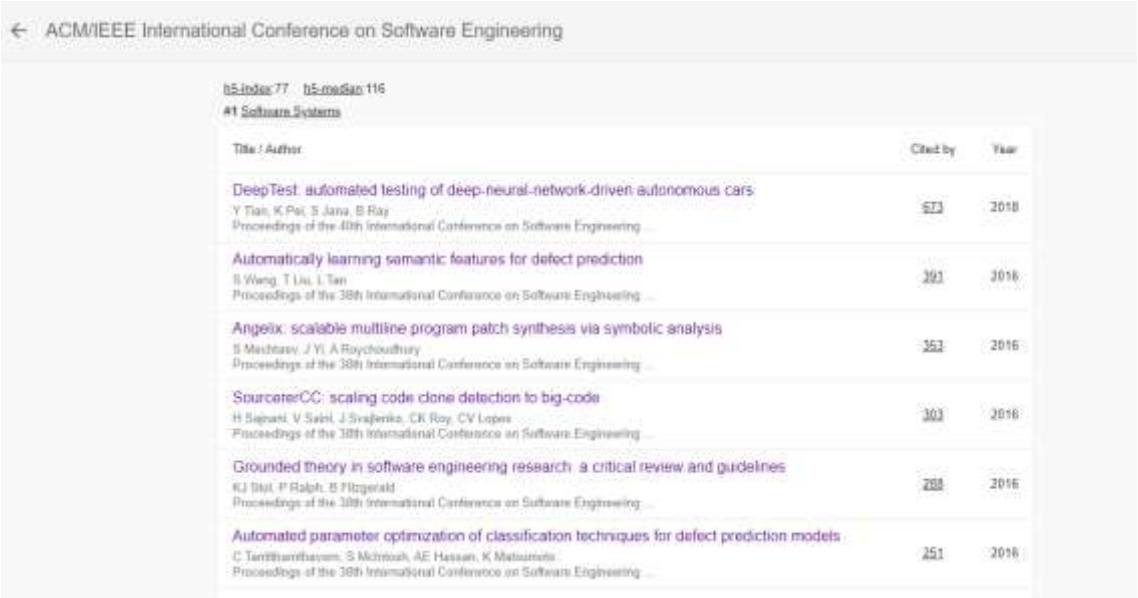

Figure 6. Excerpt of the top 5 publications in the period 2016 – 2021 of one of the selected venues (source: Google Scholar)

In Appendix 1 the details of the analysed papers are presented, extracted from Scopus.

It is to be noted that when performing the literature review some publications were discarded because:

- they were finally found irrelevant for the topics selected
- the paper was retracted
- the search result ended in a complete book, and not in a journal or conference articles.

As a result, the number of the selected publications for the phase 3 has resulted in 84.

3. **Phase 3 - Analysis of Software Technologies, Cybersecurity and Digital Infrastructure research topics:** The selected articles were analysed with the objective of extracting the main research challenges, trends and gaps as well as the uncovered or partially uncovered areas, in accordance with the Research Questions in section 3.1.1

3.3 Analysis of research topics

In order to identify the challenges, trends and gaps, each article has been analysed in-depth, with a special focus on the motivation, discussion and future work of each paper, but without forgetting the approach presented as it sometimes also potential challenges are discussed. Each article and its content have been mapped to one or various of the topics of the ACM Computing Classification System (ACM CCS ~~from now on~~)[2], which is the reference taxonomy in SWForum [5]. The detailed mapping can be found below in Table 2.

Since the ACM CCS taxonomy is rather broad and cover many topics (see figure below), some of which are not relevant to the scope of SWForum such as “Networks” or “Hardware”, the first step performed has been to identify which are the main topics and subtopics that are relevant

[2] https://dl.acm.org/ccs

for SWForum's thematic areas, namely software technologies, digital infrastructures and cybersecurity.

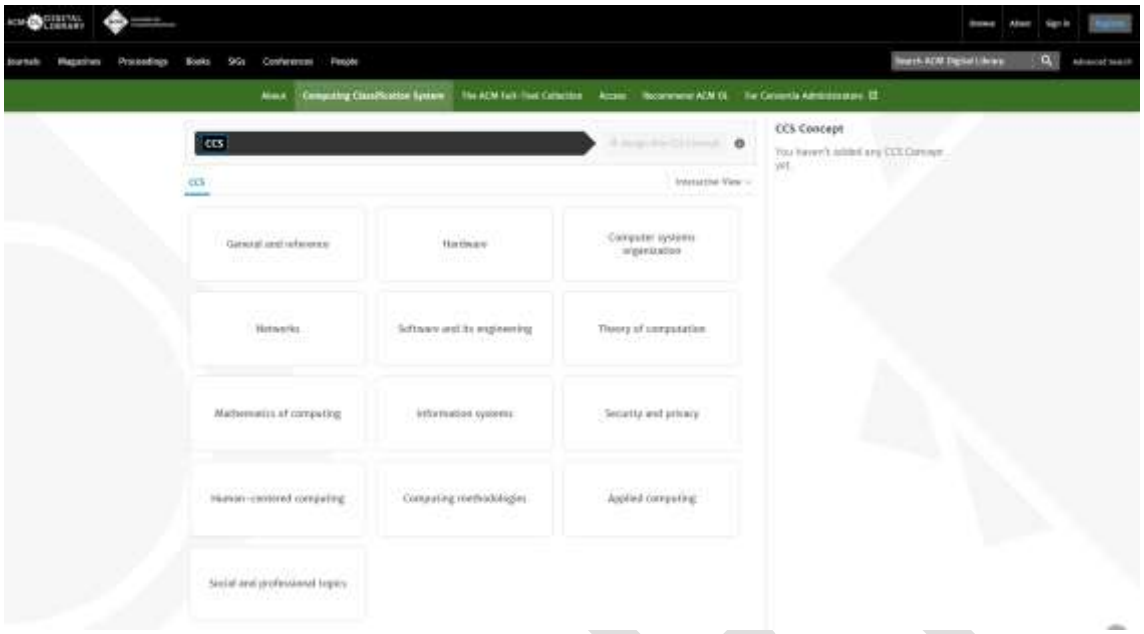

Figure 7. Main topics of the ACM Computing Classification System taxonomy (source: ACM)

Out of the main big topics of the ACM CCS taxonomy, the most relevant ones for SWForum have been deemed to be the ones shown in the table below. For each topic, the subtopics have been also considered in order to be able to perform a more granular analysis. The table in Appendix 3 shows the identified topics, subtopics and the index terms, providing therefore more detail.

Table 1. Relevant topics from the taxonomy ACM CCS for SWForum (source: ACM)

ACM CCS Topic	ACM CCS Subtopic
Software and its engineering	Software organization and properties
	Software notation and tools
	Software creation and management
Theory of Computation	Models of computation
	Formal languages and automata theory
	Computational complexity and cryptography
	Logic
	Design and analysis of algorithms
	Randomness, geometry and discrete structures
	Theory and algorithms for application domains
	Semantics and reasoning
Computing methodologies	Parallel computing methodologies
	Artificial intelligence
	Machine learning
	Modelling and simulation
	Distributed computing methodologies
	Concurrent computing methodologies
Computer Systems organization	Computer Systems organization

ACM CCS Topic	ACM CCS Subtopic
Security and privacy	Cryptography
	Formal methods and theory of security
	Security services
	Intrusion/anomaly detection and malware mitigation
	Security in hardware
	Systems security
	Network security
	Database and storage security
	Software and application security
	Human and societal aspects of security and privacy

The following table shows the coverage of the analysed papers with respect to the ACM CCS taxonomy topics identified above. The green cells in table 2 mark the main topics addressed by the papers.

Table 2. Mapping of the coverage of the analysed articles with respect to the relevant ACM CCS taxonomy (source: SWForum's own contribution)

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Angelix: scalable multiline program patch synthesis via symbolic analysis [6]																												
Automatically Learning Semantic Features for Defect Prediction [7]																												
DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars [8]																												
SourcererCC: Scaling Code Clone Detection to Big-Code [9]																												
Grounded Theory in Software Engineering Research: A Critical Review and Guidelines [10]																												
A survey of the use of crowdsourcing in software engineering [11]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Challenges and success factors for large-scale agile transformations: A systematic literature review [12]																												
Continuous deployment of software intensive products and services: A systematic mapping study [13]																												
Teamwork quality and project success in software development: A survey of agile development teams [14]																												
Continuous software engineering: A roadmap and agenda [15]																												
A Survey of App Store Analysis for Software Engineering [16]																												
A Survey on Software Fault Localization [17]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
An Empirical Comparison of Model Validation Techniques for Defect Prediction Models [18]																												
Coverage-Based Greybox Fuzzing as Markov Chain [19]																												
Nopol: Automatic Repair of Conditional Statement Bugs in Java Programs [20]																												
How to design gamification? A method for engineering gamified software [21]																												
Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? [22]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Guidelines for including grey literature and conducting multivocal literature reviews in software engineering [23]																												
Business process maturity models: A systematic literature review [24]																												
Identification and management of technical debt: A systematic mapping study [25]																												
What Would Users Change in My App? Summarizing App Reviews for Recommending Software Changes [26]																												
Why We Refactor? Confessions of GitHub Contributors [27]																												
Are Deep Neural Networks the Best Choice for Modeling Source Code [28]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Fairness Testing: Testing Software for Discrimination [29]																												
Comparing and experimenting machine learning techniques for code smell detection [30]																												
An in-depth study of the promises and perils of mining GitHub [31]																												
An empirical study of the impact of modern code review practices on software quality [32]																												
Curating GitHub for engineered software projects [33]																												
What are mobile developers asking about? A large scale study using stack overflow [34]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Fairness Testing: Testing Software for Discrimination [29]																												
DevOps [35]																												
Microservices Architecture Enables DevOps Migration to a Cloud-Native Architecture [36]																												
Microservices. The Journey So Far and Challenges Ahead [37]																												
Reference Architectures for the Internet of Things [38]																												
Ivy: Safety Verification by Interactive Generalization [39]																												
Program Synthesis from Polymorphic Refinement Types [40]																												
Bringing the Web up to Speed with WebAssembly [41]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Synthesizing Highly Expressive SQL Queries from Input-Output Examples [42]																												
Repairing Sequential Consistency in C/C++11 [43]																												
A Look at the Dynamics of the JavaScript Package Ecosystem[44]																												
AndroZoo: Collecting Millions of Android Apps for the Research Community [45]																												
Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub [46]																												
Studying the Effectiveness of Application Performance Management (APM) Tools for Detecting Performance Regressions for Web																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Applications: An Experience Report [47]																												
TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration [48]																												
DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems [49]																												
Deep Learning Code Fragments for Code Clone Detection [50]																												
Learn&Fuzz: Machine Learning for Input Fuzzing [51]																												
Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects [52]																												
Modelling the ARMv8 Architecture,																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Operationally:Concurrency and ISA [53]																												
Automatic Patch Generation by Learning Correct Code [54]																												
'Cause I'm Strong Enough:Reasoning about Consistency Choices in Distributed Systems [55]																												
Dependent Types and Multi-monadi Effects in F⋆ [56]																												
Learning Invariants using Decision Trees and Implication Counterexamples [57]																												
An Investigation into the Use of Common Libraries in Android Apps [58]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software [59]																												
Future Trends in Software Engineering Research for Mobile Apps [60]																												
History Driven Program Repair [61]																												
Under-Optimized Smart Contracts Devour Your Money [62]																												
The Tensor Algebra Compiler [63]																												
RustBelt: Securing the Foundations of the Rust Programming Language [64]																												
MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts [65]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
code2vec: Learning Distributed Representations of Code [66]																												
An Abstract Domain for Certifying Neural Networks [67]																												
Clafer: unifying class and feature modeling [68]																												
Formalizing and applying compliance patterns for business process compliance [69]																												
Scalable process discovery and conformance checking [70]																												
Model transformation intents and their properties [71]																												
A Learning-to-Rank Based Fault Localization Approach using Likely Invariants [72]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Practitioners' Expectations on Automated Fault Localization [73]																												
Sapienz: Multi-objective Automated Testing for Android Applications [74]																												
ASTOR: A Program Repair Library for Java (Demo) [75]																												
CCLearner: A Deep Learning-Based Clone Detection Approach [76]																												
Better bitmap performance with Roaring bitmaps [77]																												
The anatomy of big data computing [78]																												
ContainerCloudSim: An environment for modelling and simulation of containers in cloud data centers [79]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
iFogSim: A toolkit for modelling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments [80]																												
Microservices migration patterns [81]																												
A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution [82]																												
A Needle in a Haystack: What Do Twitter Users Say about Software? [83]																												
Automated Extraction of Conceptual Models from User Stories via NLP [84]																												
SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews [85]																												

Article title	Software organization and properties	Software notation and tools	Software creation and management	Models of computation	Formal languages and automata theory	Computational complexity and cryptography	Logic	Design and analysis of algorithms	Randomness, geometry and discrete structures	Theory and algorithms for application domains	Semantics and reasoning	Parallel computing methodologies	Artificial intelligence	Machine learning	Modelling and simulation	Distributed computing methodologies	Concurrent computing methodologies	Computer Systems organization	Cryptography	Formal methods and theory of security	Security services	Intrusion/anomaly detection and malware mitigation	Security in hardware	Systems security	Network security	Database and storage security	Software and application security	Human and societal aspects of security and privacy
Gunrock: A High-Performance Graph Processing Library on the GPU [86]																												
Keep Calm and React with Foresight: Strategies for Low Latency and Energy-Efficient Elastic Data Stream Processing [87]																												
S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters [88]																												
Groute: An Asynchronous Multi-GPU Programming Model for Irregular Computations [89]																												
SuperNeurons: Dynamic GPU Memory Management for Training Deep Neural Networks [90]																												

3.4 European Research activities

The *Information and Communication Technologies* part of the H2020 Framework Programme has supported industrial roadmaps within six main activity lines. The strategic focus of *advanced computing*, one of these six activity lines and the one that is closest to the purpose of this report, was to reinforce and expand Europe's industrial and technology strengths in low-power ICT, addressing different market segments through an integrated cross-layer (hardware, system, programming, algorithms) and cross-application/cross-market approach. Some of the recent calls to be considered are ICT-16-2018, ICT-11-2018-2019, ICT-01-2019, ICT-15-2019-2020, ICT-50-2020, ICT-56-2020 and ICT-40-2020:

- ICT-16-2018 Software Technologies[3]: This call addressed software engineering methods and tools applicable across different domains. The focus of this call was 1) on integrated programming models and techniques, and more specifically on abstractions of code and data and 2) on software ecosystems exploiting the potential of existing code bases.
- ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications[4]: The call addressed the building of industrial large-scale application test-beds that integrated Internet of Things technologies and made use of the convergence of HPC, Big Data and Cloud computing technologies.
- ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems[5]. The call addressed challenges of Cyber-physical System of Systems (CPSoS). More specifically, the aim was to develop engineering techniques to support the design-operation continuum of CPSoS in order to overcome the limitations of current computing system architectures and software design practices.
- ICT-15-2019-2020 Cloud Computing[6]: The call addressed the development of cloud solutions based on advanced cloud platforms and services and cloud-based software and data applications, as well as the opportunities brought by considering the edge devices capacities. Such solutions had to address stringent security, data protection,
- ICT-50-2020 Software Technologies[7]: The call addressed the management of the increased complexity in software systems in order to respond to emerging user needs in terms of connectivity, increasing computing power, unlimited data access independently of the underlying infrastructure and throughout shortened lifecycles, covering development and operations, across heterogenous and self-healing systems.
- ICT-56-2020 Next Generation Internet of Things[8]: The call addressed the development and demonstration of novel IoT concepts and solutions to underpin the NGI (Next Generation Internet) vision and make provision for predicting future events, trigger actions and moving decisions to the point of interest in order to better serve the end-user.

3 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-16-2018

4 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-11-2018-2019

5 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-01-2019

6 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-15-2019-2020

7 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-50-2020

8 https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-56-2020

- ICT-40-2020 Cloud Computing: towards a smart cloud computing continuum[9]: The call addressed the development of cloud solutions and testbeds combining various execution platforms for ubiquitous and seamless execution computing environments as a foundation for a complete computing continuum.

The following table shows the relationship and coverage between the ACM Computing Classification System (CCS) and the projects funded under the above topics. The details of the projects can be found in Appendix 2.

To complete this table, all the projects funded on the topics above have been analysed using CORDIS (both the project fiche and the results tab, which includes the demonstrators and public deliverables), initially, complemented by additional information checked from their websites and publications. This classification is an initial one and can evolve also as more results become available.

Table 3. European funded projects related to the topics of the research roadmaps (source: SWForum's own contribution)

ACM CCS Topic	ACM CCS Subtopic	Project acronym
Software and its engineering	Software organization and properties	ASSIST-IoT, iNGENIOUS, IoT-NGIN, TERMINET, VEDLIoT, IntellIoT, RAINBOW, Pledger, FogProtect, IoTwins, SODALITE, PIACERE, XANDAR, VeriDevOps, FOCETA, RADON, ELEGANT, COSMOS, FASTEN, UNICORE, DECODER, 1-SWARM, ADEPTNESS, ADMORPH, AMPERE, UP2DATE
	Software notation and tools	AI-Sprint, CHARITY, Datacloud, PHYSICS, SERRANO, ASSIST-IoT, Pledger, SmartCLIDE, ACCORDION, INFINITECH, EVOLVE, SODALITE, PIACERE, XANDAR, VeriDevOps, FOCETA, RADON, ELEGANT, COSMOS, FASTEN, UNICORE, DECODER, 1-SWARM, ADEPTNESS, ADMORPH, AMPERE, CPSoSaware, TEACHING, UP2DATE
	Software creation and management	AI-Sprint, CHARITY, Datacloud, PHYSICS, SmartCLIDE, MORPHEMIC, FogProtect, ACCORDION, INFINITECH, SODALITE, PIACERE, XANDAR, VeriDevOps, FOCETA, RADON, ELEGANT, COSMOS, FASTEN, UNICORE, DECODER, 1-SWARM, ADEPTNESS, ADMORPH, AMPERE, CPSoSaware, TEACHING, UP2DATE
Theory of Computation	Models of computation	CHARITY
	Formal languages and automata theory	-

[9] https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-40-2020

ACM CCS Topic	ACM CCS Subtopic	Project acronym
	Computational complexity and cryptography	-
	Logic	-
	Design and analysis of algorithms	AI-Sprint, CHARITY, ASSIST-IoT, MORPHEMIC, PIACERE, XANDAR, VeriDevOps, FOCETA, 1-SWARM
	Randomness, geometry and discrete structures	-
	Theory and algorithms for application domains	AI-Sprint, CHARITY, ASSIST-IoT, SELENE
	Semantics and reasoning	PHYSICS, RAINBOW, INFINITECH, CYBELE, SODALITE, RADON, DECODER, ADEPTNESS,
Computing methodologies	Parallel computing methodologies	XANDAR, AMPERE
	Artificial intelligence	AI-Sprint, CHARITY, Datacloud, PHYSICS, VEDLIoT, IntellIoT, SmartCLIDE, RAINBOW, MORPHEMIC, ACCORDION, IoTwins, DeepHealth, CYBELE, SODALITE, PIACERE, XANDAR, VeriDevOps, ELEGANT, COSMOS, 1-SWARM, AMPERE, CPSoSaware, SELENE, TEACHING
	Machine learning	AI-Sprint, CHARITY, ASSIST-IoT, IoT-NGIN, INFINITECH, SODALITE, PIACERE, XANDAR, VeriDevOps, FOCETA, COSMOS
	Modelling and simulation	CHARITY, Datacloud, PIACERE, XANDAR, VeriDevOps, FOCETA, SODALITE, RADON, 1-SWARM, MORPHEMIC
	Distributed computing methodologies	SERRANO, MORPHEMIC, ACCORDION, EVOLVE
	Concurrent computing methodologies	-
Computer Systems organization	Computer Systems organization	ASSIST-IoT, iNGENIOUS, IoT-NGIN, TERMINET, VEDLIoT, RAINBOW, Pledger, FogProtect, EVOLVE, IoTwins, CYBELE, XANDAR
Security and privacy	Cryptography	-
	Formal methods and theory of security	VeriDevOps
	Security services	Datacloud, SERRANO, RAINBOW, Pledger, INFINITECH, CPSoSaware
	Intrusion/anomaly detection and malware mitigation	-
	Security in hardware	SELENE
	Systems security	-
	Network security	iNGENIOUS
	Database and storage security	FogProtect
	Software and application security	AI-Sprint, Datacloud, IoT-NGIN, VEDLIoT, IntellIoT, iNGENIOUS, TERMINET, IntellIoT,

ACM CCS Topic	ACM CCS Subtopic	Project acronym
		Pledger, ACCORDION, COSMOS, INFINITECH, PIACERE, XANDAR, VeriDevOps, ELEGANT, ADMORPH, CPSoSaware, UP2DATE
	Human and societal aspects of security and privacy	-

3.5 Input from the landscape report

As part of the Landscape report [1], SWForum carried out a survey answered by academia, the public and the private sectors, where among other topics, questions about the research agenda were included.

With respect to the question on where their biggest research interest was, the following topics were selected [1]:

- Artificial intelligence, and AI combined with big data, context aware systems, intelligent systems and distributed computing
- Cybersecurity, security, security of “runtime issues”, security and privacy by design, cybersecurity for emerging technologies
- Data Analytics, Big data, Data governance, data sharing
- Robotics
- Logistics, blockchain, “ontochain” type of technologies
- Smart cities
- Industrial, energy and health
- Bioengineering
- Software engineering, Computer games software, IoT software engineering, software process mining,
- Reproducibility,
- Space/Defence
- Digital sovereignty, quantum computing and its integration with traditional computing
- Quantum computing
- Security, formal methods of security,
- New interaction technologies to empower users and increase the bandwidth of communication between operators/users and computing devices

While software engineering is mentioned as a topic in itself, the key takeaway is that organizations aim at researching in software engineering applied to a specific domain (e.g. smart cities), combined with other technologies (e.g. artificial intelligence, cybersecurity) or for new paradigms (e.g. quantum computing).

As part of the survey [1], a question regarding what is missing from most software technologies research agendas at national and European level, the following items were highlighted:

- Open source
- (methods, tools, techniques for) Cybersecurity, especially security "runtime issues", such as Blockchain, Ontochain type of technologies, Security and Privacy by design, formal methods for security and digital sovereignty

- (methods, tools, techniques for the) Software development lifecycle, and more specifically Integration of ML/AI techniques in development, modern software development processes, Reproducibility, traceability, Big Code and formal methods.
- (methods, tools, techniques for the) operation of software
- Compliance, with a special focus on Quality in evolving software, Security by design, Cross regulatory issues, standards, certification, Data governance

3.6 Input from the SWForum workshops

The First SWForum workshop on trustworthy software and open source

Nowadays, the increasing importance of software for business, industry and life of common citizens makes the notion of trustworthiness a crucial one. A trustworthy software is one for which some important properties such as correctness, compliance, reliability, availability, performance, safety, security, maintainability, privacy of data, energy efficiency, sustainability, and certified interaction with humans are ensured.

SWForum project aims to create a self-sustainable online forum that facilitates and encourages both researchers and practitioners as well as projects in software, digital infrastructure and cybersecurity to create intersections of expertise and a multidisciplinary approach to research and innovation. In this light, the project has organised two SWForum Workshops so far and reported in [2].

The First SWForum.eu workshop was held virtually on 23 - 25 March 2021 and focused on Trustworthy Software and Open Source. The stated aim of the event was to trigger discussion around the relationship between trustworthiness and open-source tools and methods to support the development of software.

The workshop was quite successful and has captured the interest of a significant number of researchers and practitioners in the area. The discussion and exchange of ideas was very lively and useful as a basis to create a community. The online First SWForum workshop aimed at focusing on the following problems:

- Characteristics of trustworthy software
- Developing trustworthy software
- The role of open source in the development of trustworthy software

The Second SWForum.eu workshop

While the first workshop has been useful for the support action to gather visibility and for the audience to solicit the discussion around the important themes of open source and trustworthiness, the second workshop has been focusing more on organizational reflections.
The second SwForum.eu workshop focused on Research Challenges, Collaboration, and Coordination among European Software Engineering Projects and was held online from 29th of June 2021 to the 1st of July 2021.

The purpose of the Second SWForum workshop was threefold: a) to discuss about open research challenges that are of interest in the European Software Engineering community; b) to discuss about collaboration opportunities among European projects in the area of software engineering; c) to understand how to develop a self-sustaining and self-governed research forum at the European level.

To achieve these objectives there was an opened call for contributions to collect research challenges and an invitation to various projects to introduce themselves at the workshop so to start the discussion about collaboration opportunities and the development of a long-lasting research forum. All the submitted challenges were accepted:

Challenge 1: An ecosystem for analysis software
Challenge 2: IaC Cost and Performance Optimization
Challenge 3: The role of AI in the edge-to-cloud context

The second SWForum.eu workshop had a total of 29 registrants from 10 different European Funded software, cybersecurity and digital infrastructure projects. The workshop was divided in three days, with specific topics and objectives for each one.

The three pillars of the SwForum.eu (digital infrastructures, software engineering and cybersecurity) were addressed by the challenges presented and by the projects represented in the SwForum.eu workshops.

During the first session two challenges among the three selected ones were presented. At the end of the session was presented the SwForum.eu Radar to classify and understand the software projects. The projects represented in the workshop were Full5G, DICE, SODALITE, PIACERE, RADON, MEDINA, EOSC, Policy Cloud, Gaia-X, Cyberwatching.eu. Seeking collaboration among the projects is essential mainly from two perspectives:

- Technical collaboration. In this case SwForum.eu can act as facilitator helping projects identifying collaboration opportunities.
- Cross-fertilization through initiatives such as SwForum.eu. One of the assets offered by SWForum.eu is the Online SWForum.eu to create interesting discussions about cross projects topics and the Projects' Hub where the projects can have their own mini-sites.

The complete details of the workshops are reported in [2].

4 Initial findings

In this section, the initial set of research challenges and topics are described, answering to the research questions mentioned above. The primary source for these challenges is the analysis of academic venues but this has been complemented with the context analysis presented in section 3, which includes the results from the landscape report (see D3.1), the SWForum workshops [2], especially the second one, the surveys [1], and the analysis of the current funded projects (see section 3.4 and appendix 2) in the topics relevant to SWForum.

All research challenges are presented following the same structure:

- Description: brief description of the research challenge.
- Expected time: expected time in which this challenge is expected to become deployed
- Societal impact: expected impact that that research challenge would have, when solved, in the society as a whole
- Technological impact: main technology result
- Business impact: expected impact that research challenge would have, when solved, in the business domain
- Source: it indicates the source of the research challenge, such as the research venue analysis, the context analysis, surveys, etc.

3.2 Challenge 1: Open-source software

Description: Open-source software (OSS) has become a reliable alternative to proprietary software. There are number of possibilities open-source offers to the users. Being free from licensing costs and supported by shared R&D, and programmers, OSS allows smaller players, with more limited financial capacity, to enter the market within home technology services, for which proprietary licence prices have kept profit margins low.

In regard to the analysed scientific venues, our results show that popular OSS software tools are intensively used in both academia and industry for the formal verification of large-scale distributed software and hardware systems. The case studies involved for instance specification of the compliance requirements (using the CRL) that are applied to companies' processes with the main objective of investigating the applicability and expressiveness of compliance patterns that have been introduced as an integral part of the compliance management approach.

Societal impact: OSS has a societal influence in terms of:

- Free software also broadens access to employment by providing a range of possible knowledge and the means to acquire those skills.
- Influence of OSS on security in general, and safety.
- In addition to security and safety as public goods, OSS is perceived to become relevant in environmental science, but also disaster impact assessment, and energy efficiency.
- legislation awareness

Business impact: In addition to the impacts on the whole economy and the companies, OSS adoption could increase companies' profitability, savings related to the development of software, improved productivity, as well as shorten time to market, and enhance innovation capability.

Moreover, OSS could have an impact on economic development in terms of:

- Cost synergy between OSS and proprietary software in product development and marketing.

- Competition effects of OSS can have impacts on the price and quality of proprietary software.
- Common goal is cost reduction, as well as to reduce energy consumption

Technology impact: Companies enter OSS communities to extend their resources, align their strategies, as well as integrate and share results.
In addition, opening up source codes leads to improved security (cloud security, security by design), and greater transparency of processes.

Source: SWForum workshop [2], academic venues [6], [7], [20], [33], [46], [52], [59], [69], survey [1].

3.3 Challenge 2: Self-repairing and self-healing: Defect prediction and fault localization using artificial intelligence

Description: Bugs are prevalent in software. Software, also mature commercial software, is released and deployed on a regular basis with defects, known and unknown. Many of these defects, including security-related ones, remain unresolved for long periods of time. The business impact of these defects is huge. These bugs can happen for many reasons: faults introduced by missing code, code smells, and more.

In order to prevent these defects, there is the need to develop tools, methods and algorithms that would allow an automatic program repair (self-repairing), that is, a software able to translate a specification into a machine-executable activity that would automatically generate a fix for that fault (self-healing). This could be achieved by the development of means for obtaining a semantic representation of programs automatically from the source code, where deep learning algorithms could be later applied. Other options could include the analysis of execution traces of a program running on the test cases in a test suite and defining defect localization algorithms, as well as machine learning algorithms and techniques that automatically learn and exploit properties of correct code.

Societal impact: More trustworthy software

Business impact: Decrease in the economic loss of not having a trustworthy software. Increase reputation and trust.

Technology impact: use of AI for self-repairing and self-healing of software applications

Source: SWForum workshop [2], survey [1], European Research activities, academic venues [6], [7], [8], [18], [32], [54], [61], [72].

3.4 Challenge 3: Continuous software engineering

Description: Continuous Software Engineering includes DevOps, Agile software development, continuous deployment, continuous delivery, continuous testing, continuous integration techniques pertain to the continuous delivery model.

Continuous integration refers to the process of adding new code commit to source code.
Continuous delivery builds on continuous integration and each code commit is automatically tested at the time it is added.

Continuous testing adds manual testing to the continuous delivery model.

Continuous deployment adds more automation to the software development process.

Delivering the Continuous agenda inspires a number of significant challenges which need to be addressed. Continuous concept has to be perceived when one considers approaches as Enterprise Agile, DevOps, Leann, Beyond Budgeting, and other similar concepts in Lean Thinking. Those philosophies require a holistic and integrated approach across all the activities in the software development lifecycle. As well, it is necessary to be highlighted the need for tighter connection between the various phases of business strategy, development and execution.

Our analysis for instance, has found evidence that the IDE used by the developers affects the adoption of automated refactoring tools.

Societal impact: It is relevant to the environmental aspects (reducing carbon footprints) but also to energy efficiency; security in general, and more trustworthy software, standardization; skills development, increased development teams and customer collaboration.

Business impact: better quality products that meet customers' needs and behaviours, shorten time to market, cost reduction, faster return on investment (ROI).

Technology impact: Shorten development cycles, improved security and privacy, more trustworthy software.

Source: SWForum workshop [2], survey [1], European Research activities, academic venues [11], [12], [14], [15]

3.5 Challenge 4: Requirements, Architecture and development

Description: Requirements engineering is a cornerstone of the software development lifecycle. Requirements are expressed in natural language, which often leads to misconceptions, especially when trying to create the conceptual and architectural models of said requirements. The use of Natural Language Processing and heuristics could help in deriving conceptual models ensuring thereof a traceability between requirements and the next phases in the development life cycle.

Systems nowadays are more and more complex. Microservices are becoming more popular in certain domains such as the cloud, edge and distributed computing in general. The migration from traditional architectural models to a more loosely coupled one, based on microservices pose several challenges, such as in how to achieve a proper communication or in performance related issues.

Deep learning in programming languages, abstractions, semantic representations of syntax of programming languages and (supervised) machine learning algorithms could benefit the quality of the code finally delivered, as code smells and faults can be easier localized and resolved. The need to release code faster and faster often implies to suffer from technical debt in the mid and long term. Technical debt is caused when code delivery is promoted over the quality of the code, which will later on will have to be refactored. However technical debt occurs in all phases of the SDLC and different actions and strategies need to be performed in order to alleviate the consequences of such decisions. Some of these can include automation of tests, defect prediction (see challenge 2), or the application of continuous engineering practices but there are more.

Finally, new paradigms such as quantum computing will affect the way in which software is developed, where abstractions for modelling, designing and building quantum applications will play a prominent role.

Societal impact: Better quality products and services.

Business impact: better quality products and services that meet customers' needs and behaviours, shorten time to market, cost reduction, faster return on investment (ROI).

Technology impact: shorten development cycles, more maintainable software, trustworthy software

Source: Academic venues [28], [30], [38], [50], [66], [68], [71], [76]

3.6 Challenge 5: Cybersecurity and privacy

Description: A complex issue, securing all parts still does not imply that the whole system is secure. Security is a responsibility of all, not only technical but also organisational. This problem is increasing as the Edge is becoming more popular as edge devices are more prone to be attacked. A common security framework in Europe that will enable the possibility to automate Security tests might ease the pain. Crucial in the software systems is to assess risks in network and software design, risks in information processing, transmission and storage; detect, prevent and respond to attacks or system failures; and regularly test and monitor the effectiveness of key systems and procedures. So, to do software security better it is inevitable to "shift left" - conduct security testing from the beginning and throughout the software development life cycle (SDLC). The modelling and verification of task allocation and authorization constraints have also gained significant interest, particularly in the information systems security field. Task allocation and authorization constraints represent an important aspect in compliance requirements.

Moreover, a continuous analysis of third-party libraries used for the development of software systems to detect security and privacy issue becomes now more necessary than ever. The use of machine learning techniques for malware detection to detect malicious libraries and applications in large and complex systems, as well as large datasets need to be further evaluated.

Another important issue is related to data transparency and sharing data with others, and more specifically, in terms of formats, interoperability, quality, reliability and licensing.

Societal impact: legislation awareness (e.g. GDPR, Cybersecurity Act), more trustworthy software, improved data protection (GDPR), more trust and privacy compliance (GDPR)

Business impact: increased productivity and effectiveness.

Technology impact: standardisation, increased security and safety.

Source: SWForum workshop [2], survey [1], academic venues [16], [38], [39], [54], [58], [60], [61], [62], [69], [71], [78], [79], [80], [81]

3.7 Challenge 6: Specific technologydomains

Description: Specific application domains include IoT, embedded systems and Blockchain among others. These have been selected as they were recurring in the venues analysed.

The embedded systems and Internet of Things subsector exhibits a massive impact in the ICT sector as an enabling technology for a variety of applications. It creates the possibility to cover basic hardware and software functionality using Open Source Software and Hardware (OSSH) and immediately start at the point of innovation.

Societal impact: The positive societal and business impact of IoT, embedded systems and blockchain, among other technology domains is significant, ranging from:

- Legislation awareness, personal data protection (GDPR), helping governments improve quality of life, reducing carbon footprints. However, Blockchain technology in particular could have a vast amounts of energy consumption.
- Optimization of energy consumption in data centres as well as energy aware scheduling.
- Increasing access to education in remote communities and improving transportation safety.
- Applicable to domains such as traffic telematics, mHealth, mEducation, technology enhanced learning, customers can choose honest and worthy suppliers on real information.
- More trust, transparency, privacy laws compliance (GDPR), legitimacy, digital identity.

Business impact: IoT significantly increases productivity; helping governments reduce healthcare costs (mHealth) for instance (cost reduction), shortened time to market, more satisfied customers, better return on investment.

Technology impact: Building digital ecosystems, increased security and privacy, safety, more trustworthy software, standardization.

Source: Survey [1], academic venues [11], [54], [63], [67], [78], [80]

5 Conclusions

This deliverable, the first of a set of three, has as its main goals the identification of research challenges, gaps and trends, which will finally result in the research roadmaps to be delivered to the European Commission by the end of the project.

The document has introduced the methodology that will be followed for such roadmapping, stating the main focus of this particular deliverable. Then, the methodology followed for the identification of the high-level research topics is presented. A multi-sourced approach has been followed, where a literature review analysis is rather prominent. This has been combined with the results from the SWForum workshops, the survey carried out in the landscape report, and an evaluation of several European research activities.

Finally, the initial findings have been reported following all the same structure. Currently, a total of six has been identified. However, it is to be noteworthy that these initial findings will be presented to the SWForum constituency, classified, prioritized and detailed further in subsequent versions of this deliverable.

6 References

[1] SWForum consortium, 'D3.1 Landscape report - v1'.

[2] SWForum consortium, 'D2.1 Cross-fertilization workshop report - v1'. 2021.

[3] SWForum consortium, 'D1.4 Contributing to the European Cloud Computing Strategic Research and Innovation Agenda Q3-2021'.

[4] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, and for the PRISMA Group, 'Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement', *BMJ*, vol. 339, no. jul21 1, pp. b2535–b2535, Jul. 2009, doi: 10.1136/bmj.b2535.

[5] SWForum consortium, 'D2.4 Project Radar Taxonomies'.

[6] S. Mechtaev, J. Yi, and A. Roychoudhury, 'Angelix: scalable multiline program patch synthesis via symbolic analysis', in *Proceedings of the 38th International Conference on Software Engineering*, Austin, Texas, May 2016, pp. 691–701. doi: 10.1145/2884781.2884807.

[7] S. Wang, T. Liu, and L. Tan, 'Automatically Learning Semantic Features for Defect Prediction', in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, May 2016, pp. 297–308. doi: 10.1145/2884781.2884804.

[8] Y. Tian, K. Pei, S. Jana, and B. Ray, 'DeepTest: automated testing of deep-neural-network-driven autonomous cars', in *Proceedings of the 40th International Conference on Software Engineering*, Gothenburg, Sweden, May 2018, pp. 303–314. doi: 10.1145/3180155.3180220.

[9] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, 'SourcererCC: scaling code clone detection to big-code', in *Proceedings of the 38th International Conference on Software Engineering*, Austin, Texas, May 2016, pp. 1157–1168. doi: 10.1145/2884781.2884877.

[10] K.-J. Stol, P. Ralph, and B. Fitzgerald, 'Grounded theory in software engineering research: a critical review and guidelines', in *Proceedings of the 38th International Conference on Software Engineering*, Austin, Texas, May 2016, pp. 120–131. doi: 10.1145/2884781.2884833.

[11] K. Mao, L. Capra, M. Harman, and Y. Jia, 'A survey of the use of crowdsourcing in software engineering', *Journal of Systems and Software*, vol. 126, pp. 57–84, Apr. 2017, doi: 10.1016/j.jss.2016.09.015.

[12] K. Dikert, M. Paasivaara, and C. Lassenius, 'Challenges and success factors for large-scale agile transformations: A systematic literature review', *Journal of Systems and Software*, vol. 119, pp. 87–108, Sep. 2016, doi: 10.1016/j.jss.2016.06.013.

[13] P. Rodríguez *et al.*, 'Continuous deployment of software intensive products and services: A systematic mapping study', *Journal of Systems and Software*, vol. 123, pp. 263–291, Jan. 2017, doi: 10.1016/j.jss.2015.12.015.

[14] Y. Lindsjørn, D. I. K. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå, 'Teamwork quality and project success in software development: A survey of agile development teams', *Journal of Systems and Software*, vol. 122, pp. 274–286, Dec. 2016, doi: 10.1016/j.jss.2016.09.028.

[15] B. Fitzgerald and K.-J. Stol, 'Continuous software engineering: A roadmap and agenda', *Journal of Systems and Software*, vol. 123, pp. 176–189, Jan. 2017, doi: 10.1016/j.jss.2015.06.063.

[16] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, 'A Survey of App Store Analysis for Software Engineering', *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 817–847, Sep. 2017, doi: 10.1109/TSE.2016.2630689.

[17] W. E. Wong, R. Gao, Y. Li, R. Abreu, and F. Wotawa, 'A Survey on Software Fault Localization', *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 707–740, Aug. 2016, doi: 10.1109/TSE.2016.2521368.

[18] 'An Empirical Comparison of Model Validation Techniques for Defect Prediction Models'. https://ieeexplore.ieee.org/document/7497471/ (accessed Oct. 04, 2021).

[19] M. Böhme, V.-T. Pham, and A. Roychoudhury, 'Coverage-Based Greybox Fuzzing as Markov Chain', *IEEE Transactions on Software Engineering*, vol. 45, no. 5, pp. 489–506, May 2019, doi: 10.1109/TSE.2017.2785841.

[20] J. Xuan *et al.*, 'Nopol: Automatic Repair of Conditional Statement Bugs in Java Programs', *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 34–55, Jan. 2017, doi: 10.1109/TSE.2016.2560811.

[21] B. Morschheuser, L. Hassan, K. Werder, and J. Hamari, 'How to design gamification? A method for engineering gamified software', *Information and Software Technology*, vol. 95, pp. 219–237, Mar. 2018, doi: 10.1016/j.infsof.2017.10.015.

[22] Y. Yu, H. Wang, G. Yin, and T. Wang, 'Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?', *Information and Software Technology*, vol. 74, pp. 204–218, Jun. 2016, doi: 10.1016/j.infsof.2016.01.004.

[23] V. Garousi, M. Felderer, and M. V. Mäntylä, 'Guidelines for including grey literature and conducting multivocal literature reviews in software engineering', *Information and Software Technology*, vol. 106, pp. 101–121, Feb. 2019, doi: 10.1016/j.infsof.2018.09.006.

[24] A. Tarhan, O. Turetken, and H. A. Reijers, 'Business process maturity models: A systematic literature review', *Information and Software Technology*, vol. 75, pp. 122–134, Jul. 2016, doi: 10.1016/j.infsof.2016.01.010.

[25] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, 'Identification and management of technical debt: A systematic mapping study', *Information and Software Technology*, vol. 70, pp. 100–121, Feb. 2016, doi: 10.1016/j.infsof.2015.10.008.

[26] A. Di Sorbo *et al.*, 'What would users change in my app? summarizing app reviews for recommending software changes', in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seattle, WA, USA, Nov. 2016, pp. 499–510. doi: 10.1145/2950290.2950299.

[27] D. Silva, N. Tsantalis, and M. T. Valente, 'Why we refactor? confessions of GitHub contributors', in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seattle, WA, USA, Nov. 2016, pp. 858–870. doi: 10.1145/2950290.2950305.

[28] V. J. Hellendoorn and P. Devanbu, 'Are deep neural networks the best choice for modeling source code?', in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, Paderborn, Germany, Aug. 2017, pp. 763–773. doi: 10.1145/3106237.3106290.

[29] S. Galhotra, Y. Brun, and A. Meliou, 'Fairness testing: testing software for discrimination', in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, Paderborn, Germany, Aug. 2017, pp. 498–510. doi: 10.1145/3106237.3106277.

[30] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, 'Comparing and experimenting machine learning techniques for code smell detection', *Empir Software Eng*, vol. 21, no. 3, pp. 1143–1191, Jun. 2016, doi: 10.1007/s10664-015-9378-4.

[31] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, 'An in-depth study of the promises and perils of mining GitHub', *Empir Software Eng*, vol. 21, no. 5, pp. 2035–2071, Oct. 2016, doi: 10.1007/s10664-015-9393-5.

[32] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, 'An empirical study of the impact of modern code review practices on software quality', *Empir Software Eng*, vol. 21, no. 5, pp. 2146–2189, Oct. 2016, doi: 10.1007/s10664-015-9381-9.

[33] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, 'Curating GitHub for engineered software projects', *Empir Software Eng*, vol. 22, no. 6, pp. 3219–3253, Dec. 2017, doi: 10.1007/s10664-017-9512-6.

[34] C. Rosen and E. Shihab, 'What are mobile developers asking about? A large scale study using stack overflow', *Empir Software Eng*, vol. 21, no. 3, pp. 1192–1223, Jun. 2016, doi: 10.1007/s10664-015-9379-3.

[35] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, 'DevOps', *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016, doi: 10.1109/MS.2016.68.
[36] 'Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture'. https://ieeexplore.ieee.org/document/7436659/ (accessed Oct. 04, 2021).
[37] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, 'Microservices: The Journey So Far and Challenges Ahead', *IEEE Software*, vol. 35, no. 3, pp. 24–35, May 2018, doi: 10.1109/MS.2018.2141039.
[38] M. Weyrich and C. Ebert, 'Reference Architectures for the Internet of Things', *IEEE Software*, vol. 33, no. 1, pp. 112–116, Jan. 2016, doi: 10.1109/MS.2016.20.
[39] O. Padon, K. L. McMillan, A. Panda, M. Sagiv, and S. Shoham, 'Ivy: safety verification by interactive generalization', in *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Santa Barbara, CA, USA, Jun. 2016, pp. 614–630. doi: 10.1145/2908080.2908118.
[40] N. Polikarpova, I. Kuraj, and A. Solar-Lezama, 'Program synthesis from polymorphic refinement types', *SIGPLAN Not.*, vol. 51, no. 6, pp. 522–538, Aug. 2016, doi: 10.1145/2980983.2908093.
[41] A. Haas *et al.*, 'Bringing the web up to speed with WebAssembly', in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Barcelona, Spain, Jun. 2017, pp. 185–200. doi: 10.1145/3062341.3062363.
[42] C. Wang, A. Cheung, and R. Bodik, 'Synthesizing highly expressive SQL queries from input-output examples', in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Barcelona, Spain, Jun. 2017, pp. 452–466. doi: 10.1145/3062341.3062365.
[43] O. Lahav, V. Vafeiadis, J. Kang, C.-K. Hur, and D. Dreyer, 'Repairing sequential consistency in C/C++11', *SIGPLAN Not.*, vol. 52, no. 6, pp. 618–632, Jun. 2017, doi: 10.1145/3140587.3062352.
[44] E. Wittern, P. Suter, and S. Rajagopalan, 'A look at the dynamics of the JavaScript package ecosystem', in *Proceedings of the 13th International Conference on Mining Software Repositories*, Austin, Texas, May 2016, pp. 351–361. doi: 10.1145/2901739.2901743.
[45] K. Allix, T. F. Bissyandé, J. Klein, and Y. L. Traon, 'AndroZoo: Collecting Millions of Android Apps for the Research Community', in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, May 2016, pp. 468–471.
[46] M. Beller, G. Gousios, and A. Zaidman, 'Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub', in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, May 2017, pp. 356–367. doi: 10.1109/MSR.2017.62.
[47] T. M. Ahmed, C.-P. Bezemer, T.-H. Chen, A. E. Hassan, and W. Shang, 'Studying the Effectiveness of Application Performance Management (APM) Tools for Detecting Performance Regressions for Web Applications: An Experience Report', in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, May 2016, pp. 1–12.
[48] M. Beller, G. Gousios, and A. Zaidman, 'TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration', in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, May 2017, pp. 447–450. doi: 10.1109/MSR.2017.24.
[49] L. Ma *et al.*, 'DeepGauge: multi-granularity testing criteria for deep learning systems', in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Montpellier, France, Sep. 2018, pp. 120–131. doi: 10.1145/3238147.3238202.
[50] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, 'Deep learning code fragments for code clone detection', in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2016, pp. 87–98.

[51] P. Godefroid, H. Peleg, and R. Singh, 'Learn amp;Fuzz: Machine learning for input fuzzing', in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Oct. 2017, pp. 50–59. doi: 10.1109/ASE.2017.8115618.

[52] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, 'Usage, costs, and benefits of continuous integration in open-source projects', in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2016, pp. 426–437.

[53] S. Flur *et al.*, 'Modelling the ARMv8 architecture, operationally: concurrency and ISA', in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St. Petersburg, FL, USA, Jan. 2016, pp. 608–621. doi: 10.1145/2837614.2837615.

[54] F. Long and M. Rinard, 'Automatic patch generation by learning correct code', in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St. Petersburg, FL, USA, Jan. 2016, pp. 298–312. doi: 10.1145/2837614.2837617.

[55] A. Gotsman, H. Yang, C. Ferreira, M. Najafzadeh, and M. Shapiro, ''Cause I'm strong enough: Reasoning about consistency choices in distributed systems', in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St. Petersburg, FL, USA, Jan. 2016, pp. 371–384. doi: 10.1145/2837614.2837625.

[56] N. Swamy *et al.*, 'Dependent types and multi-monadic effects in F*', in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St. Petersburg, FL, USA, Jan. 2016, pp. 256–270. doi: 10.1145/2837614.2837655.

[57] P. Garg, D. Neider, P. Madhusudan, and D. Roth, 'Learning invariants using decision trees and implication counterexamples', *SIGPLAN Not.*, vol. 51, no. 1, pp. 499–512, Apr. 2016, doi: 10.1145/2914770.2837664.

[58] L. Li, T. F. Bissyandé, J. Klein, and Y. Le Traon, 'An Investigation into the Use of Common Libraries in Android Apps', in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Mar. 2016, vol. 1, pp. 403–414. doi: 10.1109/SANER.2016.52.

[59] M. Beller, R. Bholanath, S. McIntosh, and A. Zaidman, 'Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software', in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Mar. 2016, vol. 1, pp. 470–481. doi: 10.1109/SANER.2016.105.

[60] M. Nagappan and E. Shihab, 'Future Trends in Software Engineering Research for Mobile Apps', in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Mar. 2016, vol. 5, pp. 21–32. doi: 10.1109/SANER.2016.88.

[61] 'History Driven Program Repair'. https://ieeexplore.ieee.org/document/7476644/ (accessed Oct. 04, 2021).

[62] T. Chen, X. Li, X. Luo, and X. Zhang, 'Under-optimized smart contracts devour your money', in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb. 2017, pp. 442–446. doi: 10.1109/SANER.2017.7884650.

[63] F. Kjolstad, S. Kamil, S. Chou, D. Lugato, and S. Amarasinghe, 'The tensor algebra compiler', *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, pp. 1–29, Oct. 2017, doi: 10.1145/3133901.

[64] R. Jung, J.-H. Jourdan, R. Krebbers, and D. Dreyer, 'RustBelt: securing the foundations of the Rust programming language', *Proc. ACM Program. Lang.*, vol. 2, no. POPL, pp. 1–34, Jan. 2018, doi: 10.1145/3158154.

[65] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis, 'MadMax: surviving out-of-gas conditions in Ethereum smart contracts', *Proc. ACM Program. Lang.*, vol. 2, no. OOPSLA, pp. 1–27, Oct. 2018, doi: 10.1145/3276486.

[66] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, 'code2vec: learning distributed representations of code', *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 1–29, Jan. 2019, doi: 10.1145/3290353.

[67] G. Singh, T. Gehr, M. Püschel, and M. Vechev, 'An abstract domain for certifying neural networks', *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 1–30, Jan. 2019, doi: 10.1145/3290354.

[68] K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski, 'Clafer: unifying class and feature modeling', *Softw Syst Model*, vol. 15, no. 3, pp. 811–845, Jul. 2016, doi: 10.1007/s10270-014-0441-1.

[69] A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou, 'Formalizing and applying compliance patterns for business process compliance', *Softw Syst Model*, vol. 15, no. 1, pp. 119–146, Feb. 2016, doi: 10.1007/s10270-014-0395-3.

[70] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, 'Scalable process discovery and conformance checking', *Softw Syst Model*, vol. 17, no. 2, pp. 599–631, May 2018, doi: 10.1007/s10270-016-0545-x.

[71] L. Lúcio *et al.*, 'Model transformation intents and their properties', *Softw Syst Model*, vol. 15, no. 3, pp. 647–684, Jul. 2016, doi: 10.1007/s10270-014-0429-x.

[72] T.-D. B. Le, D. Lo, C. Le Goues, and L. Grunske, 'A learning-to-rank based fault localization approach using likely invariants', in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Saarbrücken, Germany, Jul. 2016, pp. 177–188. doi: 10.1145/2931037.2931049.

[73] P. S. Kochhar, X. Xia, D. Lo, and S. Li, 'Practitioners' expectations on automated fault localization', in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Saarbrücken, Germany, Jul. 2016, pp. 165–176. doi: 10.1145/2931037.2931051.

[74] K. Mao, M. Harman, and Y. Jia, 'Sapienz: multi-objective automated testing for Android applications', in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Saarbrücken, Germany, Jul. 2016, pp. 94–105. doi: 10.1145/2931037.2931054.

[75] M. Martinez and M. Monperrus, 'ASTOR: a program repair library for Java (demo)', in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Saarbrücken, Germany, Jul. 2016, pp. 441–444. doi: 10.1145/2931037.2948705.

[76] L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder, 'CCLearner: A Deep Learning-Based Clone Detection Approach', in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2017, pp. 249–260. doi: 10.1109/ICSME.2017.46.

[77] S. Chambi, D. Lemire, O. Kaser, and R. Godin, 'Better bitmap performance with Roaring bitmaps', *Software: Practice and Experience*, vol. 46, no. 5, pp. 709–719, 2016, doi: 10.1002/spe.2325.

[78] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, 'The anatomy of big data computing', *Software: Practice and Experience*, vol. 46, no. 1, pp. 79–105, 2016, doi: 10.1002/spe.2374.

[79] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, 'ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers', *Software: Practice and Experience*, vol. 47, no. 4, pp. 505–521, 2017, doi: 10.1002/spe.2422.

[80] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, 'iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments', *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017, doi: 10.1002/spe.2509.

[81] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn, 'Microservices migration patterns', *Software: Practice and Experience*, vol. 48, no. 11, pp. 2019–2042, 2018, doi: 10.1002/spe.2608.

[82] E. Guzman, M. Ibrahim, and M. Glinz, 'A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution', in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 11–20. doi: 10.1109/RE.2017.88.

[83] E. Guzman, R. Alkadhi, and N. Seyff, 'A Needle in a Haystack: What Do Twitter Users Say about Software?', in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 96–105. doi: 10.1109/RE.2016.67.

[84] M. Robeer, G. Lucassen, J. M. E. M. van der Werf, F. Dalpiaz, and S. Brinkkemper, 'Automated Extraction of Conceptual Models from User Stories via NLP', in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 196–205. doi: 10.1109/RE.2016.40.

[85] T. Johann, C. Stanik, A. M. Alizadeh B., and W. Maalej, 'SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews', in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 21–30. doi: 10.1109/RE.2017.71.

[86] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens, 'Gunrock: a high-performance graph processing library on the GPU', in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Barcelona, Spain, Feb. 2016, pp. 1–12. doi: 10.1145/2851141.2851145.

[87] T. De Matteis and G. Mencagli, 'Keep calm and react with foresight: strategies for low-latency and energy-efficient elastic data stream processing', *SIGPLAN Not.*, vol. 51, no. 8, p. 13:1-13:12, Feb. 2016, doi: 10.1145/3016078.2851148.

[88] A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, 'S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters', in *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Austin, Texas, USA, Jan. 2017, pp. 193–205. doi: 10.1145/3018743.3018769.

[89] T. Ben-Nun, M. Sutton, S. Pai, and K. Pingali, 'Groute: An Asynchronous Multi-GPU Programming Model for Irregular Computations', *SIGPLAN Not.*, vol. 52, no. 8, pp. 235–248, Jan. 2017, doi: 10.1145/3155284.3018756.

[90] L. Wang *et al.*, 'Superneurons: dynamic GPU memory management for training deep neural networks', in *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Vienna, Austria, Feb. 2018, pp. 41–53. doi: 10.1145/3178487.3178491.

Appendix 1 – Academic venues analysed

This appendix shows the raw information of the analysed academic venues, as extracted from Scopus. No edits have been performed. They are included here for repeatability and transparency purposes.

Scopus yields its results under the following structure:

- Title
- Author 1
- Author 2
- Author n
- Venue
- Abstract
- Place (optional)
- Venue (optional)
- Date
- Year
- DOI
- Publishing editorial

Adaptable Blockchain-Based Systems: A Case Study for Product Traceability

Lu, Qinghua

Xu, Xiwei

IEEE Software

Tracing the origin of products across complex supply chains requires a transparent, tamper-proof metadata infrastructure that's not only trusted by all the involved parties but also adaptable to changing environments and regulations. Can such advanced infrastructure be implemented in a decentralized way? Qinghua Lu and Xiwei Xu share their story of developing the originChain system, which leverages emerging blockchain technology to do so.

2017/11//

2017

10.1109/MS.2017.4121227

IEEE Xplore

DevOps

Ebert, Christof

Gallardo, Gorka

Hernantes, Josune

Serrano, Nicolas

IEEE Software

Building on lean and agile practices, DevOps means end-to-end automation in software development and delivery. Hardly anybody will be able to approach it with a cookbook-style approach, but most developers will benefit from better connecting the previously isolated silos of development and operations. Many DevOps tools exist that can help them do this.

2016/05//

2016

10.1109/MS.2016.68

IEEE Xplore

Reference Architectures for the Internet of Things

Weyrich, Michael

Ebert, Christof

IEEE Software

The Internet of Things (IoT) is about innovative functionality and better productivity by seamlessly connecting devices. But a major threat is the lack of architecture standards for the industrial Internet and connectivity in the IoT. This article reviews recent IoT architecture evolution and what it means for industry projects.

2016/01//

2016

10.1109/MS.2016.20

IEEE Xplore

Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture

This article reports on experiences and lessons learned during incremental migration and architectural refactoring of a commercial mobile back end as a service to microservices architecture. It explains how the researchers adopted DevOps and how this facilitated a smooth migration.

Microservices Architecture Enables DevOps

https://ieeexplore.ieee.org/document/7436659/

2021/10/04/13:28:17

An in-depth study of the promises and perils of mining GitHub

Kalliamvakou, Eirini

Gousios, Georgios

Blincoe, Kelly

Singer, Leif

German, Daniel M.

Damian, Daniela

Empirical Software Engineering

With over 10 million git repositories, GitHub is becoming one of the most important sources of software artifacts on the Internet. Researchers mine the information stored in GitHub's event logs to understand how its users employ the site to collaborate on software, but so far there have been no studies describing the quality and properties of the available GitHub data. We document the results of an empirical study aimed at understanding the characteristics of the repositories and users in GitHub; we see how users take advantage of GitHub's main features and how their activity is tracked on GitHub and related datasets to point out misalignment between the real and mined data. Our results indicate that while GitHub is a rich source of data on software development, mining GitHub for research purposes should take various potential perils into consideration. For example, we show that the majority of the projects are personal and inactive, and that almost 40 % of all pull requests do not appear as merged even though they were. Also, approximately half of GitHub's registered users do not have public activity, while the activity of GitHub users in repositories is not always easy to pinpoint. We use our identified perils to see if they can pose validity threats; we review selected papers from the MSR 2014 Mining Challenge and see if there are potential impacts to consider. We provide a set of recommendations for software engineering researchers on how to approach the data in GitHub.

2016/10/01/

2016

10.1007/s10664-015-9393-5

Springer Link

What are mobile developers asking about? A large scale study using stack overflow

Rosen, Christoffer

Shihab, Emad

Empirical Software Engineering

The popularity of mobile devices has been steadily growing in recent years. These devices heavily depend on software from the underlying operating systems to the applications they run. Prior research showed that mobile software is different than traditional, large software systems. However, to date most of our research has been conducted on traditional software systems. Very little work has focused on the issues that mobile developers face. Therefore, in this paper, we use data from the popular online Q&A site, Stack Overflow, and analyze 13,232,821 posts to examine what mobile developers ask about. We employ Latent Dirichlet allocation-based topic models to help us summarize the mobile-related questions. Our findings show that developers are asking about app distribution, mobile APIs, data management, sensors and context, mobile tools, and user interface development. We also determine what popular mobile-related issues are the most difficult, explore platform specific issues, and investigate the types (e.g., what, how, or why) of questions mobile developers ask. Our findings help highlight the challenges facing mobile developers that require more attention from the software engineering research and development communities in the future and establish a novel approach for analyzing questions asked on Q&A forums.

2016/06/01/

2016

10.1007/s10664-015-9379-3

Springer Link

Curating GitHub for engineered software projects

Munaiah, Nuthan

Kroh, Steven

Cabrey, Craig

Nagappan, Meiyappan

Empirical Software Engineering

Software forges like GitHub host millions of repositories. Software engineering researchers have been able to take advantage of such a large corpora of potential study subjects with the help of tools like GHTorrent and Boa. However, the simplicity in querying comes with a caveat: there are limited means of separating the signal (e.g. repositories containing engineered software projects) from the noise (e.g. repositories containing home work assignments). The proportion of noise in a random sample of repositories could skew the study and may lead to researchers reaching unrealistic, potentially inaccurate, conclusions. We argue that it is imperative to have the ability to sieve out the noise in such large repository forges. We propose a framework, and present a reference implementation of the framework as a tool called reaper, to enable researchers to select GitHub repositories that contain evidence of an engineered software project. We identify software engineering practices

(called dimensions) and propose means for validating their existence in a GitHub repository. We used reaper to measure the dimensions of 1,857,423 GitHub repositories. We then used manually classified data sets of repositories to train classifiers capable of predicting if a given GitHub repository contains an engineered software project. The performance of the classifiers was evaluated using a set of 200 repositories with known ground truth classification. We also compared the performance of the classifiers to other approaches to classification (e.g. number of GitHub Stargazers) and found our classifiers to outperform existing approaches. We found stargazers-based classifier (with 10 as the threshold for number of stargazers) to exhibit high precision (97%) but an inversely proportional recall (32%). On the other hand, our best classifier exhibited a high precision (82%) and a high recall (86%). The stargazer-based criteria offers precision but fails to recall a significant portion of the population.

2017/12/01/

2017

10.1007/s10664-017-9512-6

Springer Link

An empirical study of the impact of modern code review practices on software quality

McIntosh, Shane

Kamei, Yasutaka

Adams, Bram

Hassan, Ahmed E.

Empirical Software Engineering

Software code review, i.e., the practice of having other team members critique changes to a software system, is a well-established best practice in both open source and proprietary software domains. Prior work has shown that formal code inspections tend to improve the quality of delivered software. However, the formal code inspection process mandates strict review criteria (e.g., in-person meetings and reviewer checklists) to ensure a base level of review quality, while the modern, lightweight code reviewing process does not. Although recent work explores the modern code review process, little is known about the relationship between modern code review practices and long-term software quality. Hence, in this paper, we study the relationship between post-release defects (a popular proxy for long-term software quality) and: (1) code review coverage, i.e., the proportion of changes that have been code reviewed, (2) code review participation, i.e., the degree of reviewer involvement in the code review process, and (3) code reviewer expertise, i.e., the level of domain-specific expertise of the code reviewers. Through a case study of the Qt, VTK, and ITK projects, we find that code review coverage, participation, and expertise share a

significant link with software quality. Hence, our results empirically confirm the intuition that poorly-reviewed code has a negative impact on software quality in large systems using modern reviewing tools.

2016/10/01/

2016

10.1007/s10664-015-9381-9

Springer Link

Comparing and experimenting machine learning techniques for code smell detection

Arcelli Fontana, Francesca

Mäntylä, Mika V.

Zanoni, Marco

Marino, Alessandro

Empirical Software Engineering

Several code smell detection tools have been developed providing different results, because smells can be subjectively interpreted, and hence detected, in different ways. In this paper, we perform the largest experiment of applying machine learning algorithms to code smells to the best of our knowledge. We experiment 16 different machine-learning algorithms on four code smells (Data Class, Large Class, Feature Envy, Long Method) and 74 software systems, with 1986 manually validated code smell samples. We found that all algorithms achieved high performances in the cross-validation data set, yet the highest performances were obtained by J48 and Random Forest, while the worst performance were achieved by support vector machines. However, the lower prevalence of code smells, i.e., imbalanced data, in the entire data set caused varying performances that need to be addressed in the future studies. We conclude that the application of machine learning to the detection of these code smells can provide high accuracy (>96 %), and only a hundred training examples are needed to reach at least 95 % accuracy.

2016/06/01/

2016

10.1007/s10664-015-9378-4

Why we refactor? confessions of GitHub contributors

Silva, Danilo

Tsantalis, Nikolaos

Valente, Marco Tulio

FSE 2016

Refactoring is a widespread practice that helps developers to improve the maintainability and readability of their code. However, there is a limited number of studies empirically investigating the actual motivations behind specific refactoring operations applied by developers. To fill this gap, we monitored Java projects hosted on GitHub to detect recently applied refactorings, and asked the developers to explain the reasons behind their decision to refactor the code. By applying thematic analysis on the collected responses, we compiled a catalogue of 44 distinct motivations for 12 well-known refactoring types. We found that refactoring activity is mainly driven by changes in the requirements and much less by code smells. Extract Method is the most versatile refactoring operation serving 11 different purposes. Finally, we found evidence that the IDE used by the developers affects the adoption of automated refactoring tools.

Seattle, WA, USA

Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering

2016/11/01/

2016

10.1145/2950290.2950305

ACM Digital Library

What would users change in my app? summarizing app reviews for recommending software changes

Di Sorbo, Andrea

Panichella, Sebastiano

Alexandru, Carol V.

Shimagaki, Junji

Visaggio, Corrado A.

Canfora, Gerardo

Gall, Harald C.

FSE 2016

Mobile app developers constantly monitor feedback in user reviews with the goal of improving their mobile apps and better meeting user expectations. Thus, automated approaches have been proposed in literature with the aim of reducing the effort required for analyzing feedback contained in user reviews via automatic

classification/prioritization according to specific topics. In this paper, we introduce SURF (Summarizer of User Reviews Feedback), a novel approach to condense the enormous amount of information that developers of popular apps have to manage due to user feedback received on a daily basis. SURF relies on a conceptual model for capturing user needs useful for developers performing maintenance and evolution tasks. Then it uses sophisticated summarisation techniques for summarizing thousands of reviews and generating an interactive, structured and condensed agenda of recommended software changes. We performed an end-to-end evaluation of SURF on user reviews of 17 mobile apps (5 of them developed by Sony Mobile), involving 23 developers and researchers in total. Results demonstrate high accuracy of SURF in summarizing reviews and the usefulness of the recommended changes. In evaluating our approach we found that SURF helps developers in better understanding user needs, substantially reducing the time required by developers compared to manually analyzing user (change) requests and planning future software changes.

Seattle, WA, USA

Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering

2016/11/01/

2016

10.1145/2950290.2950299

ACM Digital Library

Deep API learning

Gu, Xiaodong

Zhang, Hongyu

Zhang, Dongmei

Kim, Sunghun

FSE 2016

Developers often wonder how to implement a certain functionality (e.g., how to parse XML files) using APIs. Obtaining an API usage sequence based on an API-related natural language query is very helpful in this regard. Given a query, existing approaches utilize information retrieval models to search for matching API sequences. These approaches treat queries and APIs as bags-of-words and lack a deep understanding of the semantics of the query. We propose DeepAPI, a deep learning based approach to generate API usage sequences for a given natural language query. Instead of a bag-of-words assumption, it learns the sequence of words in a query and the sequence of associated APIs. DeepAPI adapts a neural language model named RNN Encoder-Decoder. It encodes a word sequence (user query) into a fixed-length context vector, and generates

an API sequence based on the context vector. We also augment the RNN Encoder-Decoder by considering the importance of individual APIs. We empirically evaluate our approach with more than 7 million annotated code snippets collected from GitHub. The results show that our approach generates largely accurate API sequences and outperforms the related approaches.

Seattle, WA, USA

Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering

2016/11/01/

2016

10.1145/2950290.2950334

ACM Digital Library

Identification and management of technical debt: A systematic mapping study

Alves, Nicolli S. R.

Mendes, Thiago S.

de Mendonça, Manoel G.

Spínola, Rodrigo O.

Shull, Forrest

Seaman, Carolyn

Information and Software Technology

Context

The technical debt metaphor describes the effect of immature artifacts on software maintenance that bring a short-term benefit to the project in terms of increased productivity and lower cost, but that may have to be paid off with interest later. Much research has been performed to propose mechanisms to identify debt and decide the most appropriate moment to pay it off. It is important to investigate the current state of the art in order to provide both researchers and practitioners with information that enables further research activities as well as technical debt management in practice.

Objective

This paper has the following goals: to characterize the types of technical debt, identify indicators that can be used to find technical debt, identify management strategies, understand the maturity level of each proposal, and identify what visualization techniques have been

proposed to support technical debt identification and management activities.

Method

A systematic mapping study was performed based on a set of three research questions. In total, 100 studies, dated from 2010 to 2014, were evaluated.

Results

We proposed an initial taxonomy of technical debt types, created a list of indicators that have been proposed to identify technical debt, identified the existing management strategies, and analyzed the current state of art on technical debt, identifying topics where new research efforts can be invested.

Conclusion

The results of this mapping study can help to identify points that still require further investigation in technical debt research.

2016/02/01/

2016

10.1016/j.infsof.2015.10.008

ScienceDirect

Nopol: Automatic Repair of Conditional Statement Bugs in Java Programs

Xuan, Jifeng

Martinez, Matias

DeMarco, Favio

Clément, Maxime

Marcote, Sebastian Lamelas

Durieux, Thomas

Le Berre, Daniel

Monperrus, Martin

IEEE Transactions on Software Engineering

We propose Nopol, an approach to automatic repair of buggy conditional statements (i.e., if-then-else statements). This approach takes a buggy program as well as a test suite as input and generates a patch with a conditional expression as output. The test suite is required to contain passing test cases to model the expected behavior of the program and at least one failing test case that reveals the bug to be repaired. The process of Nopol consists of three major phases. First, Nopol employs

angelic fix localization to identify expected values of a condition during the test execution. Second, runtime trace collection is used to collect variables and their actual values, including primitive data types and objected-oriented features (e.g., nullness checks), to serve as building blocks for patch generation. Third, Nopol encodes these collected data into an instance of a Satisfiability Modulo Theory (SMT) problem; then a feasible solution to the SMT instance is translated back into a code patch. We evaluate Nopol on 22 real-world bugs (16 bugs with buggy if conditions and six bugs with missing preconditions) on two large open-source projects, namely Apache Commons Math and Apache Commons Lang. Empirical analysis on these bugs shows that our approach can effectively fix bugs with buggy if conditions and missing preconditions. We illustrate the capabilities and limitations of Nopol using case studies of real bug fixes.

2017/01//

2017

10.1109/TSE.2016.2560811

IEEE Xplore

Coverage-Based Greybox Fuzzing as Markov Chain

Böhme, Marcel

Pham, Van-Thuan

Roychoudhury, Abhik

IEEE Transactions on Software Engineering

Coverage-based Greybox Fuzzing (CGF) is a random testing approach that requires no program analysis. A new test is generated by slightly mutating a seed input. If the test exercises a new and interesting path, it is added to the set of seeds; otherwise, it is discarded. We observe that most tests exercise the same few “high-frequency” paths and develop strategies to explore significantly more paths with the same number of tests by gravitating towards low-frequency paths. We explain the challenges and opportunities of CGF using a Markov chain model which specifies the probability that fuzzing the seed that exercises path i generates an input that exercises path j. Each state (i.e., seed) has an energy that specifies the number of inputs to be generated from that seed. We show that CGF is considerably more efficient if energy is inversely proportional to the density of the stationary distribution and increases monotonically every time that seed is chosen. Energy is controlled with a power schedule. We implemented several schedules by extending AFL. In 24 hours, AFLFast exposes 3 previously unreported CVEs that are not exposed by AFL and exposes 6 previously unreported CVEs 7x faster than AFL. AFLFast produces at least an order of magnitude more unique crashes than AFL. We compared AFLFast to the symbolic executor Klee. In terms of vulnerability detection, AFLFast is significantly more effective than Klee on the same subject programs that were discussed in

the original Klee paper. In terms of code coverage, AFLFast only slightly outperforms Klee while a combination of both tools achieves best results by mitigating the individual weaknesses.

2019/05//

2019

10.1109/TSE.2017.2785841

IEEE Xplore

An Empirical Comparison of Model Validation Techniques for Defect Prediction Models

Defect prediction models help software quality assurance teams to allocate their limited resources to the most defect-prone modules. Model validation techniques, such as -fold cross-validation, use historical data to estimate how well a model will perform in the future. However, little is known about how accurate the estimates of model validation techniques tend to be. In this paper, we investigate the bias and variance of model validation techniques in the domain of defect prediction. Analysis of 101 public defect datasets suggests that 77 percent of them are highly susceptible to producing unstable results– -selecting an appropriate model validation technique is a critical experimental design choice. Based on an analysis of 256 studies in the defect prediction literature, we select the 12 most commonly adopted model validation techniques for evaluation. Through a case study of 18 systems, we find that single-repetition holdout validation tends to produce estimates with 46-229 percent more bias and 53-863 percent more variance than the top-ranked model validation techniques. On the other hand, out-of-sample bootstrap validation yields the best balance between the bias and variance of estimates in the context of our study. Therefore, we recommend that future defect prediction studies avoid single-repetition holdout validation, and instead, use out-of-sample bootstrap validation.

A Survey of App Store Analysis for Software Engineering

Martin, William

Sarro, Federica

Jia, Yue

Zhang, Yuanyuan

Harman, Mark

IEEE Transactions on Software Engineering

App Store Analysis studies information about applications obtained from app stores. App stores provide a wealth of information derived from

users that would not exist had the applications been distributed via previous software deployment methods. App Store Analysis combines this non-technical information with technical information to learn trends and behaviours within these forms of software repositories. Findings from App Store Analysis have a direct and actionable impact on the software teams that develop software for app stores, and have led to techniques for requirements engineering, release planning, software design, security and testing. This survey describes and compares the areas of research that have been explored thus far, drawing out common aspects, trends and directions future research should take to address open problems and challenges.

2017/09//

2017

10.1109/TSE.2016.2630689

IEEE Xplore

A Survey on Software Fault Localization

Wong, W. Eric

Gao, Ruizhi

Li, Yihao

Abreu, Rui

Wotawa, Franz

IEEE Transactions on Software Engineering

Software fault localization, the act of identifying the locations of faults in a program, is widely recognized to be one of the most tedious, time consuming, and expensive - yet equally critical - activities in program debugging. Due to the increasing scale and complexity of software today, manually locating faults when failures occur is rapidly becoming infeasible, and consequently, there is a strong demand for techniques that can guide software developers to the locations of faults in a program with minimal human intervention. This demand in turn has fueled the proposal and development of a broad spectrum of fault localization techniques, each of which aims to streamline the fault localization process and make it more effective by attacking the problem in a unique way. In this article, we catalog and provide a comprehensive overview of such techniques and discuss key issues and concerns that are pertinent to software fault localization as a whole.

2016/08//

2016

10.1109/TSE.2016.2521368

IEEE Xplore

Grounded theory in software engineering research: a critical review and guidelines

Stol, Klaas-Jan

Ralph, Paul

Fitzgerald, Brian

ICSE '16

Grounded Theory (GT) has proved an extremely useful research approach in several fields including medical sociology, nursing, education and management theory. However, GT is a complex method based on an inductive paradigm that is fundamentally different from the traditional hypothetico-deductive research model. As there are at least three variants of GT, some ostensibly GT research suffers from method slurring, where researchers adopt an arbitrary subset of GT practices that are not recognizable as GT. In this paper, we describe the variants of GT and identify the core set of GT practices. We then analyze the use of grounded theory in software engineering. We carefully and systematically selected 98 articles that mention GT, of which 52 explicitly claim to use GT, with the other 46 using GT techniques only. Only 16 articles provide detailed accounts of their research procedures. We offer guidelines to improve the quality of both conducting and reporting GT studies. The latter is an important extension since current GT guidelines in software engineering do not cover the reporting process, despite good reporting being necessary for evaluating a study and informing subsequent research.

Austin, Texas

Proceedings of the 38th International Conference on Software Engineering

2016/05/14/

2016

10.1145/2884781.2884833

ACM Digital Library

SourcererCC: scaling code clone detection to big-code

Sajnani, Hitesh

Saini, Vaibhav

Svajlenko, Jeffrey

Roy, Chanchal K.

Lopes, Cristina V.

ICSE '16

Despite a decade of active research, there has been a marked lack in clone detection techniques that scale to large repositories for detecting near-miss clones. In this paper, we present a token-based clone detector, SourcererCC, that can detect both exact and near-miss clones from large inter-project repositories using a standard workstation. It exploits an optimized inverted-index to quickly query the potential clones of a given code block. Filtering heuristics based on token ordering are used to significantly reduce the size of the index, the number of code-block comparisons needed to detect the clones, as well as the number of required token-comparisons needed to judge a potential clone. We evaluate the scalability, execution time, recall and precision of SourcererCC, and compare it to four publicly available and state-of-the-art tools. To measure recall, we use two recent benchmarks: (1) a big benchmark of real clones, BigCloneBench, and (2) a Mutation/Injection-based framework of thousands of fine-grained artificial clones. We find SourcererCC has both high recall and precision, and is able to scale to a large inter-project repository (25K projects, 250MLOC) using a standard workstation.

Austin, Texas

Proceedings of the 38th International Conference on Software Engineering

2016/05/14/

2016

10.1145/2884781.2884877

ACM Digital Library

Angelix: scalable multiline program patch synthesis via symbolic analysis

Mechtaev, Sergey

Yi, Jooyong

Roychoudhury, Abhik

ICSE '16

Since debugging is a time-consuming activity, automated program repair tools such as GenProg have garnered interest. A recent study revealed that the majority of GenProg repairs avoid bugs simply by deleting functionality. We found that SPR, a state-of-the-art repair tool proposed in 2015, still deletes functionality in their many "plausible" repairs. Unlike generate-and-validate systems such as GenProg and SPR, semantic analysis based repair techniques synthesize a repair based on semantic information of the program. While such semantics-based repair methods show promise in terms of quality of generated repairs, their scalability has been a concern so far. In this paper, we present Angelix, a novel semantics-based repair method that scales up to programs of

similar size as are handled by search-based repair tools such as GenProg and SPR. This shows that Angelix is more scalable than previously proposed semantics based repair methods such as SemFix and DirectFix. Furthermore, our repair method can repair multiple buggy locations that are dependent on each other. Such repairs are hard to achieve using SPR and GenProg. In our experiments, Angelix generated repairs from large-scale real-world software such as wireshark and php, and these generated repairs include multi-location repairs. We also report our experience in automatically repairing the well-known Heartbleed vulnerability.

Austin, Texas

Proceedings of the 38th International Conference on Software Engineering

2016/05/14/

2016

10.1145/2884781.2884807

ACM Digital Library

Automatically Learning Semantic Features for Defect Prediction

Wang, Song

Liu, Taiyue

Tan, Lin

2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)

Software defect prediction, which predicts defective code regions, can help developers find bugs and prioritize their testing efforts. To build accurate prediction models, previous studies focus on manually designing features that encode the characteristics of programs and exploring different machine learning algorithms. Existing traditional features often fail to capture the semantic differences of programs, and such a capability is needed for building accurate prediction models. To bridge the gap between programs' semantics and defect prediction features, this paper proposes to leverage a powerful representation-learning algorithm, deep learning, to learn semantic representation of programs automatically from source code. Specifically, we leverage Deep Belief Network (DBN) to automatically learn semantic features from token vectors extracted from programs' Abstract Syntax Trees (ASTs). Our evaluation on ten open source projects shows that our automatically learned semantic features significantly improve both within-project defect prediction (WPDP) and cross-project defect prediction (CPDP) compared to traditional features. Our semantic features improve WPDP on average by 14.7% in precision, 11.5% in recall, and 14.2% in F1. For CPDP, our semantic features based approach outperforms the state-of-the-art technique TCA+ with traditional features by 8.9% in F1.

2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)

2016/05//

2016

10.1145/2884781.2884804

IEEE Xplore

Microservices: The Journey So Far and Challenges Ahead

Jamshidi, Pooyan

Pahl, Claus

Mendonça, Nabor C.

Lewis, James

Tilkov, Stefan

IEEE Software

Microservices are an architectural approach emerging out of service-oriented architecture, emphasizing self-management and lightweightness as the means to improve software agility, scalability, and autonomy. This article examines microservice evolution from the technological and architectural perspectives and discusses key challenges facing future microservice developments.

2018/05//

2018

10.1109/MS.2018.2141039

IEEE Xplore

Ivy: safety verification by interactive generalization

Padon, Oded

McMillan, Kenneth L.

Panda, Aurojit

Sagiv, Mooly

Shoham, Sharon

PLDI '16

Despite several decades of research, the problem of formal verification of infinite-state systems has resisted effective automation. We describe

a system --- Ivy --- for interactively verifying safety of infinite-state systems. Ivy's key principle is that whenever verification fails, Ivy graphically displays a concrete counterexample to induction. The user then interactively guides generalization from this counterexample. This process continues until an inductive invariant is found. Ivy searches for universally quantified invariants, and uses a restricted modeling language. This ensures that all verification conditions can be checked algorithmically. All user interactions are performed using graphical models, easing the user's task. We describe our initial experience with verifying several distributed protocols.

Santa Barbara, CA, USA

Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation

2016/06/02/

2016

10.1145/2908080.2908118

ACM Digital Library

Program synthesis from polymorphic refinement types

Polikarpova, Nadia

Kuraj, Ivan

Solar-Lezama, Armando

ACM SIGPLAN Notices

We present a method for synthesizing recursive functions that provably satisfy a given specification in the form of a polymorphic refinement type. We observe that such specifications are particularly suitable for program synthesis for two reasons. First, they offer a unique combination of expressive power and decidability, which enables automatic verification—and hence synthesis—of nontrivial programs. Second, a type-based specification for a program can often be effectively decomposed into independent specifications for its components, causing the synthesizer to consider fewer component combinations and leading to a combinatorial reduction in the size of the search space. At the core of our synthesis procedure is a newalgorithm for refinement type checking, which supports specification decomposition. We have evaluated our prototype implementation on a large set of synthesis problems and found that it exceeds the state of the art in terms of both scalability and usability. The tool was able to synthesize more complex programs than those reported in prior work (several sorting algorithms and operations on balanced search trees), as well as most of the benchmarks tackled by existing synthesizers, often starting from a more concise and intuitive user input.

2016/08//

2016

10.1145/2980983.2908093

DOI.org (Crossref)

Repairing sequential consistency in C/C++11

Lahav, Ori

Vafeiadis, Viktor

Kang, Jeehoon

Hur, Chung-Kil

Dreyer, Derek

ACM SIGPLAN Notices

The C/C++11 memory model defines the semantics of concurrent memory accesses in C/C++, and in particular supports racy "atomic" accesses at a range of different consistency levels, from very weak consistency ("relaxed") to strong, sequential consistency ("SC"). Unfortunately, as we observe in this paper, the semantics of SC atomic accesses in C/C++11, as well as in all proposed strengthenings of the semantics, is flawed, in that (contrary to previously published results) both suggested compilation schemes to the Power architecture are unsound. We propose a model, called RC11 (for Repaired C11), with a better semantics for SC accesses that restores the soundness of the compilation schemes to Power, maintains the DRF-SC guarantee, and provides stronger, more useful, guarantees to SC fences. In addition, we formally prove, for the first time, the correctness of the proposed stronger compilation schemes to Power that preserve load-to-store ordering and avoid "out-of-thin-air" reads.

2017/06/14/

2017

10.1145/3140587.3062352

June 2017

Studying the Effectiveness of Application Performance Management (APM) Tools for Detecting Performance Regressions for Web Applications: An Experience Report

Ahmed, Tarek M.

Bezemer, Cor-Paul

Chen, Tse-Hsun

Hassan, Ahmed E.

Shang, Weiyi

2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)

Performance regressions, such as a higher CPU utilization than in the previous version of an application, are caused by software application updates that negatively affect the performance of an application.Although a plethora of mining software repository research has been done to detect such regressions, research tools are generally not readily available to practitioners. Application Performance Management (APM) tools are commonly used in practice for detecting performance issues in the field by mining operational data.In contrast to performance regression detection tools that assume a changing code base and a stable workload, APM tools mine operational data to detect performance anomalies caused by a changing workload in an otherwise stable code base. Although APM tools are widely used in practice, no research has been done to understand 1) whether APM tools can identify performance regressions caused by code changes and 2) how well these APM tools support diagnosing the root-cause of these regressions.In this paper, we explore if the readily accessible APM tools can help practitioners detect performance regressions. We perform a case study using three commercial (AppDynamics, New Relic and Dynatrace) and one open source (Pinpoint) APM tools. In particular, we examine the effectiveness of leveraging these APM tools in detecting and diagnosing injected performance regressions (excessive memory usage, high CPU utilization and inefficient database queries) in three open source applications. We find that APM tools can detect most of the injected performance regressions, making them good candidates to detect performance regressions in practice. However, there is a gap between mining approaches that are proposed in state-of-the-art performance regression detection research and the ones used by APM tools. In addition, APM tools lack the ability to be extended, which makes it hard to enhance them when exploring novel mining approaches for detecting performance regressions.

2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)

2016/05//

2016

IEEE Xplore

AndroZoo: Collecting Millions of Android Apps for the Research Community

Allix, Kevin

Bissyandé, Tegawendé F.

Klein, Jacques

Traon, Yves Le

2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)

We present a growing collection of Android Applications col-lected from several sources, including the official GooglePlay app market. Our dataset, AndroZoo, currently contains more than three million apps, each of which has beenanalysed by tens of different AntiVirus products to knowwhich applications are detected as Malware. We provide thisdataset to contribute to ongoing research efforts, as well asto enable new potential research topics on Android Apps.By releasing our dataset to the research community, we alsoaim at encouraging our fellow researchers to engage in reproducible experiments.

2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)

2016/05//

2016

IEEE Xplore

DeepGauge: multi-granularity testing criteria for deep learning systems

Ma, Lei

Juefei-Xu, Felix

Zhang, Fuyuan

Sun, Jiyuan

Xue, Minhui

Li, Bo

Chen, Chunyang

Su, Ting

Li, Li

Liu, Yang

Zhao, Jianjun

Wang, Yadong

ASE 2018

Deep learning (DL) defines a new data-driven programming paradigm that constructs the internal system logic of a crafted neuron network through a set of training data. We have seen wide adoption of DL in many safety-critical scenarios. However, a plethora of studies have shown that the state-of-the-art DL systems suffer from various vulnerabilities which can lead to severe consequences when applied to real-world applications. Currently, the testing adequacy of a DL system is usually measured by the accuracy of test data. Considering the limitation of accessible high

quality test data, good accuracy performance on test data can hardly provide confidence to the testing adequacy and generality of DL systems. Unlike traditional software systems that have clear and controllable logic and functionality, the lack of interpretability in a DL system makes system analysis and defect detection difficult, which could potentially hinder its real-world deployment. In this paper, we propose DeepGauge, a set of multi-granularity testing criteria for DL systems, which aims at rendering a multi-faceted portrayal of the testbed. The in-depth evaluation of our proposed testing criteria is demonstrated on two well-known datasets, five DL systems, and with four state-of-the-art adversarial attack techniques against DL. The potential usefulness of DeepGauge sheds light on the construction of more generic and robust DL systems.

Montpellier, France

Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering

2018/09/03/

2018

10.1145/3238147.3238202

ACM Digital Library

DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems

Zhang, Mengshi

Zhang, Yuqun

Zhang, Lingming

Liu, Cong

Khurshid, Sarfraz

2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)

While Deep Neural Networks (DNNs) have established the fundamentals of image-based autonomous driving systems, they may exhibit erroneous behaviors and cause fatal accidents. To address the safety issues in autonomous driving systems, a recent set of testing techniques have been designed to automatically generate artificial driving scenes to enrich test suite, e.g., generating new input images transformed from the original ones. However, these techniques are insufficient due to two limitations: first, many such synthetic images often lack diversity of driving scenes, and hence compromise the resulting efficacy and reliability. Second, for machine-learning-based systems, a mismatch between training and application domain can dramatically degrade system accuracy, such that it is necessary to validate inputs for improving

system robustness. In this paper, we propose DeepRoad, an unsupervised DNN-based framework for automatically testing the consistency of DNN-based autonomous driving systems and online validation. First, DeepRoad automatically synthesizes large amounts of diverse driving scenes without using image transformation rules (e.g. scale, shear and rotation). In particular, DeepRoad is able to produce driving scenes with various weather conditions (including those with rather extreme conditions) by applying Generative Adversarial Networks (GANs) along with the corresponding real-world weather scenes. Second, DeepRoad utilizes metamorphic testing techniques to check the consistency of such systems using synthetic images. Third, DeepRoad validates input images for DNN-based systems by measuring the distance of the input and training images using their VGGNet features. We implement DeepRoad to test three well-recognized DNN-based autonomous driving systems in Udacity self-driving car challenge. The experimental results demonstrate that DeepRoad can detect thousands of inconsistent behaviors for these systems, and effectively validate input images to potentially enhance the system robustness as well.

2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)

2018/09//

2018

10.1145/3238147.3238187

IEEE Xplore

Deep learning code fragments for code clone detection

White, Martin

Tufano, Michele

Vendome, Christopher

Poshyvanyk, Denys

2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)

Code clone detection is an important problem for software maintenance and evolution. Many approaches consider either structure or identifiers, but none of the existing detection techniques model both sources of information. These techniques also depend on generic, handcrafted features to represent code fragments. We introduce learning-based detection techniques where everything for representing terms and fragments in source code is mined from the repository. Our code analysis supports a framework, which relies on deep learning, for automatically linking patterns mined at the lexical level with patterns mined at the syntactic level. We evaluated our novel learning-based approach for code clone detection with respect to feasibility from the point of view of software maintainers. We sampled and manually evaluated 398 file- and

480 method-level pairs across eight real-world Java systems; 93% of the file- and method-level samples were evaluated to be true positives. Among the true positives, we found pairs mapping to all four clone types. We compared our approach to a traditional structure-oriented technique and found that our learning-based approach detected clones that were either undetected or suboptimally reported by the prominent tool Deckard. Our results affirm that our learning-based approach is suitable for clone detection and a tenable technique for researchers.

2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)

2016/09//

2016

IEEE Xplore

Learn amp;Fuzz: Machine learning for input fuzzing

Godefroid, Patrice

Peleg, Hila

Singh, Rishabh

2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)

Fuzzing consists of repeatedly testing an application with modified, or fuzzed, inputs with the goal of finding security vulnerabilities in input-parsing code. In this paper, we show how to automate the generation of an input grammar suitable for input fuzzing using sample inputs and neural-network-based statistical machine-learning techniques. We present a detailed case study with a complex input format, namely PDF, and a large complex security-critical parser for this format, namely, the PDF parser embedded in Microsoft's new Edge browser. We discuss and measure the tension between conflicting learning and fuzzing goals: learning wants to capture the structure of well-formed inputs, while fuzzing wants to break that structure in order to cover unexpected code paths and find bugs. We also present a new algorithm for this learn&fuzz challenge which uses a learnt input probability distribution to intelligently guide where to fuzz inputs.

2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)

2017/10//

2017

10.1109/ASE.2017.8115618

IEEE Xplore

Usage, costs, and benefits of continuous integration in open-source projects

Hilton, Michael

Tunnell, Timothy

Huang, Kai

Marinov, Darko

Dig, Danny

2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)

Continuous integration (CI) systems automate the compilation, building, and testing of software. Despite CI rising as a big success story in automated software engineering, it has received almost no attention from the research community. For example, how widely is CI used in practice, and what are some costs and benefits associated with CI? Without answering such questions, developers, tool builders, and researchers make decisions based on folklore instead of data. In this paper, we use three complementary methods to study the usage of CI in open-source projects. To understand which CI systems developers use, we analyzed 34,544 open-source projects from GitHub. To understand how developers use CI, we analyzed 1,529,291 builds from the most commonly used CI system. To understand why projects use or do not use CI, we surveyed 442 developers. With this data, we answered several key questions related to the usage, costs, and benefits of CI. Among our results, we show evidence that supports the claim that CI helps projects release more often, that CI is widely adopted by the most popular projects, as well as finding that the overall percentage of projects using CI continues to grow, making it important and timely to focus more research on CI.

2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)

2016/09//

2016

IEEE Xplore

Automatic patch generation by learning correct code

Long, Fan

Rinard, Martin

POPL '16

We present Prophet, a novel patch generation system that works with a set of successful human patches obtained from open- source software repositories to learn a probabilistic, application-independent model of

correct code. It generates a space of candidate patches, uses the model to rank the candidate patches in order of likely correctness, and validates the ranked patches against a suite of test cases to find correct patches. Experimental results show that, on a benchmark set of 69 real-world defects drawn from eight open-source projects, Prophet significantly outperforms the previous state-of-the-art patch generation system.

St. Petersburg, FL, USA

Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages

2016/01/11/

2016

10.1145/2837614.2837617

ACM Digital Library

Synthesizing highly expressive SQL queries from input-output examples

Wang, Chenglong

Cheung, Alvin

Bodik, Rastislav

PLDI 2017

SQL is the de facto language for manipulating relational data. Though powerful, many users find it difficult to write SQL queries due to highly expressive constructs. While using the programming-by-example paradigm to help users write SQL queries is an attractive proposition, as evidenced by online help forums such as Stack Overflow, developing techniques for synthesizing SQL queries from given input-output (I/O) examples has been difficult, due to the large space of SQL queries as a result of its rich set of operators. In this paper, we present a new scalable and efficient algorithm for synthesizing SQL queries based on I/O examples. The key innovation of our algorithm is development of a language for abstract queries, i.e., queries with uninstantiated operators, that can be used to express a large space of SQL queries efficiently. Using abstract queries to represent the search space nicely decomposes the synthesis problem into two tasks: 1) searching for abstract queries that can potentially satisfy the given I/O examples, and 2) instantiating the found abstract queries and ranking the results. We have implemented this algorithm in a new tool called Scythe and evaluated it using 193 benchmarks collected from Stack Overflow. Our evaluation shows that Scythe can efficiently solve 74% of the benchmarks, most in just a few seconds, and the queries range from simple ones involving a single selection to complex queries with 6 nested subqueires.

Barcelona, Spain

Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation

2017/06/14/

2017

10.1145/3062341.3062365

ACM Digital Library

Bringing the web up to speed with WebAssembly

Haas, Andreas

Rossberg, Andreas

Schuff, Derek L.

Titzer, Ben L.

Holman, Michael

Gohman, Dan

Wagner, Luke

Zakai, Alon

Bastien, JF

PLDI 2017

The maturation of the Web platform has given rise to sophisticated and demanding Web applications such as interactive 3D visualization, audio and video software, and games. With that, efficiency and security of code on the Web has become more important than ever. Yet JavaScript as the only built-in language of the Web is not well-equipped to meet these requirements, especially as a compilation target. Engineers from the four major browser vendors have risen to the challenge and collaboratively designed a portable low-level bytecode called WebAssembly. It offers compact representation, efficient validation and compilation, and safe low to no-overhead execution. Rather than committing to a specific programming model, WebAssembly is an abstraction over modern hardware, making it language-, hardware-, and platform-independent, with use cases beyond just the Web. WebAssembly has been designed with a formal semantics from the start. We describe the motivation, design and formal semantics of WebAssembly and provide some preliminary experience with implementations.

Barcelona, Spain

Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation

2017/06/14/

2017

10.1145/3062341.3062363

ACM Digital Library

Fairness testing: testing software for discrimination

Galhotra, Sainyam

Brun, Yuriy

Meliou, Alexandra

ESEC/FSE 2017

This paper defines software fairness and discrimination and develops a testing-based method for measuring if and how much software discriminates, focusing on causality in discriminatory behavior. Evidence of software discrimination has been found in modern software systems that recommend criminal sentences, grant access to financial products, and determine who is allowed to participate in promotions. Our approach, Themis, generates efficient test suites to measure discrimination. Given a schema describing valid system inputs, Themis generates discrimination tests automatically and does not require an oracle. We evaluate Themis on 20 software systems, 12 of which come from prior work with explicit focus on avoiding discrimination. We find that (1) Themis is effective at discovering software discrimination, (2) state-of-the-art techniques for removing discrimination from algorithms fail in many situations, at times discriminating against as much as 98% of an input subdomain, (3) Themis optimizations are effective at producing efficient test suites for measuring discrimination, and (4) Themis is more efficient on systems that exhibit more discrimination. We thus demonstrate that fairness testing is a critical aspect of the software development cycle in domains with possible discrimination and provide initial tools for measuring software discrimination.

Paderborn, Germany

Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering

2017/08/21/

2017

10.1145/3106237.3106277

ACM Digital Library

Are deep neural networks the best choice for modeling source code?

Hellendoorn, Vincent J.

Devanbu, Premkumar

ESEC/FSE 2017

Current statistical language modeling techniques, including deep-learning based models, have proven to be quite effective for source code. We argue here that the special properties of source code can be exploited for further improvements. In this work, we enhance established language modeling approaches to handle the special challenges of modeling source code, such as: frequent changes, larger, changing vocabularies, deeply nested scopes, etc. We present a fast, nested language modeling toolkit specifically designed for software, with the ability to add & remove text, and mix & swap out many models. Specifically, we improve upon prior cache-modeling work and present a model with a much more expansive, multi-level notion of locality that we show to be well-suited for modeling software. We present results on varying corpora in comparison with traditional N-gram, as well as RNN, and LSTM deep-learning language models, and release all our source code for public use. Our evaluations suggest that carefully adapting N-gram models for source code can yield performance that surpasses even RNN and LSTM based deep-learning models.

Paderborn, Germany

Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering

2017/08/21/

2017

10.1145/3106237.3106290

ACM Digital Library

How to design gamification? A method for engineering gamified software

Morschheuser, Benedikt

Hassan, Lobna

Werder, Karl

Hamari, Juho

Information and Software Technology

Context

Since its inception around 2010, gamification has become one of the top technology and software trends. However, gamification has also been regarded as one of the most challenging areas of software engineering. Beyond traditional software design requirements, designing gamification requires the command of disciplines such as (motivational/behavioral) psychology, game design, and narratology, making the development of gamified software a challenge for traditional software developers.

Gamification software inhabits a finely tuned niche of software engineering that seeks for both high functionality and engagement; beyond technical flawlessness, gamification has to motivate and affect users. Consequently, it has also been projected that most gamified software is doomed to fail.

Objective

This paper seeks to advance the understanding of designing gamification and to provide a comprehensive method for developing gamified software.

Method

We approach the research problem via a design science research approach; firstly, by synthesizing the current body of literature on gamification design methods and by interviewing 25 gamification experts, producing a comprehensive list of design principles for developing gamified software. Secondly, and more importantly, we develop a detailed method for engineering of gamified software based on the gathered knowledge and design principles. Finally, we conduct an evaluation of the artifacts via interviews of ten gamification experts and implementation of the engineering method in a gamification project.

Results

As results of the study, we present the method and key design principles for engineering gamified software. Based on the empirical and expert evaluation, the developed method was deemed as comprehensive, implementable, complete, and useful. We deliver a comprehensive overview of gamification guidelines and shed novel insights into the nature of gamification development and design discourse.

Conclusion

This paper takes first steps towards a comprehensive method for gamified software engineering.

2018/03/01/

2018

10.1016/j.infsof.2017.10.015

ScienceDirect

Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?

Yu, Yue

Wang, Huaimin

Yin, Gang

Wang, Tao

Information and Software Technology

Context: The pull-based model, widely used in distributed software development, offers an extremely low barrier to entry for potential contributors (anyone can submit of contributions to any project, through pull-requests). Meanwhile, the project's core team must act as guardians of code quality, ensuring that pull-requests are carefully inspected before being merged into the main development line. However, with pull-requests becoming increasingly popular, the need for qualified reviewers also increases. GitHub facilitates this, by enabling the crowd-sourcing of pull-request reviews to a larger community of coders than just the project's core team, as a part of their social coding philosophy. However, having access to more potential reviewers does not necessarily mean that it's easier to find the right ones (the "needle in a haystack" problem). If left unsupervised, this process may result in communication overhead and delayed pull-request processing. Objective: This study aims to investigate whether and how previous approaches used in bug triaging and code review can be adapted to recommending reviewers for pull-requests, and how to improve the recommendation performance. Method: First, we extend three typical approaches used in bug triaging and code review for the new challenge of assigning reviewers to pull-requests. Second, we analyze social relations between contributors and reviewers, and propose a novel approach by mining each project's comment networks (CNs). Finally, we combine the CNs with traditional approaches, and evaluate the effectiveness of all these methods on 84 GitHub projects through both quantitative and qualitative analysis. Results: We find that CN-based recommendation can achieve, by itself, similar performance as the traditional approaches. However, the mixed approaches can achieve significant improvements compared to using either of them independently. Conclusion: Our study confirms that traditional approaches to bug triaging and code review are feasible for pull-request reviewer recommendations on GitHub. Furthermore, their performance can be improved significantly by combining them with information extracted from prior social interactions between developers on GitHub. These results prompt for novel tools to support process automation in social coding platforms, that combine social (e.g., common interests among developers) and technical factors (e.g., developers' expertise).

2016/06/01/

2016

10.1016/j.infsof.2016.01.004

ScienceDirect

Business process maturity models: A systematic literature review

Tarhan, Ayca

Turetken, Oktay

Reijers, Hajo A.

Information and Software Technology

Context

The number of maturity models proposed in the area of Business Process Management (BPM) has increased considerably in the last decade. However, there are a number of challenges, such as the limited empirical studies on their validation and a limited extent of actionable properties of these models in guiding their application. These challenges hinder the widespread usage of the maturity models in the BPM field.

Objective

In order to better understand the state of the research on business process maturity models (BPMMs) and identify opportunities for future research, we conducted a systematic literature review.

Method

We searched the studies between the years 1990 and 2014 in established digital libraries to identify empirical studies of BPMMs by focusing on their development, validation, and application. We targeted studies on generic models proposed for business process maturity, business process management or orientation maturity, and selected 61 studies out of 2899 retrieved initially.

Results

We found that despite that many BPMMs were proposed in the last decade, the level of empirical evidence that reveals the validity and usefulness of these models is scarce.

Conclusion

The current state of research on BPM maturity is in its early phases, and academic literature lacks methodical applications of many mainstream BPMMs that have been proposed. Future research should be directed towards: (1) reconciling existing models with a strong emphasis on prescriptive properties, (2) conducting empirical studies to demonstrate the validity and usefulness of BPMMs, and (3) separating the assessment method used to evaluate the maturity level from the maturity model which acts as the reference framework for the assessment.

2016/07/01/

2016

10.1016/j.infsof.2016.01.010

ScienceDirect

Guidelines for including grey literature and conducting multivocal literature reviews in software engineering

Garousi, Vahid

Felderer, Michael

Mäntylä, Mika V.

Information and Software Technology

Context

A Multivocal Literature Review (MLR) is a form of a Systematic Literature Review (SLR) which includes the grey literature (e.g., blog posts, videos and white papers) in addition to the published (formal) literature (e.g., journal and conference papers). MLRs are useful for both researchers and practitioners since they provide summaries both the state-of-the art and –practice in a given area. MLRs are popular in other fields and have recently started to appear in software engineering (SE). As more MLR studies are conducted and reported, it is important to have a set of guidelines to ensure high quality of MLR processes and their results.

Objective

There are several guidelines to conduct SLR studies in SE. However, several phases of MLRs differ from those of traditional SLRs, for instance with respect to the search process and source quality assessment. Therefore, SLR guidelines are only partially useful for conducting MLR studies. Our goal in this paper is to present guidelines on how to conduct MLR studies in SE.

Method

To develop the MLR guidelines, we benefit from several inputs: (1) existing SLR guidelines in SE, (2), a literature survey of MLR guidelines and experience papers in other fields, and (3) our own experiences in conducting several MLRs in SE. We took the popular SLR guidelines of Kitchenham and Charters as the baseline and extended/adopted them to conduct MLR studies in SE. All derived guidelines are discussed in the context of an already-published MLR in SE as the running example.

Results

The resulting guidelines cover all phases of conducting and reporting MLRs in SE from the planning phase, over conducting the review to the final reporting of the review. In particular, we believe that incorporating and adopting a vast set of experience-based recommendations from MLR guidelines and experience papers in other fields have enabled us to propose a set of guidelines with solid foundations.

Conclusion

Having been developed on the basis of several types of experience and evidence, the provided MLR guidelines will support researchers to effectively and efficiently conduct new MLRs in any area of SE. The authors recommend the researchers to utilize these guidelines in their MLR studies and then share their lessons learned and experiences.

2019/02/01/

2019

10.1016/j.infsof.2018.09.006

ScienceDirect

Continuous deployment of software intensive products and services: A systematic mapping study

Rodríguez, Pilar

Haghighatkhah, Alireza

Lwakatare, Lucy Ellen

Teppola, Susanna

Suomalainen, Tanja

Eskeli, Juho

Karvonen, Teemu

Kuvaja, Pasi

Verner, June M.

Oivo, Markku

Journal of Systems and Software

The software intensive industry is moving towards the adoption of a value-driven and adaptive real-time business paradigm. The traditional view of software as an item that evolves through releases every few months is being replaced by the continuous evolution of software functionality. This study aims to classify and analyse the literature related to continuous deployment in the software domain in order to scope the phenomenon, provide an overview of the state-of-the-art, investigate the scientific evidence in the reported results and identify areas suitable for further research. We conducted a systematic mapping study and classified the continuous deployment literature. The benefits and challenges related to continuous deployment were also analysed. RESULTS: The systematic mapping study includes 50 primary studies published between 2001 and 2014. An in-depth analysis of the primary studies revealed ten recurrent themes that characterize continuous deployment and provide researchers with directions for future work. In addition, a set of benefits and challenges of which practitioners may take advantage were identified. CONCLUSION: Overall, although the topic area is very promising, it is still in its infancy, thus offering a plethora of new opportunities for both researchers and software intensive companies.

2017/01/01/

2017

10.1016/j.jss.2015.12.015

ScienceDirect

Teamwork quality and project success in software development: A survey of agile development teams

Lindsjørn, Yngve

Sjøberg, Dag I. K.

Dingsøyr, Torgeir

Bergersen, Gunnar R.

Dybå, Tore

Journal of Systems and Software

Small, self-directed teams are central in agile development. This article investigates the effect of teamwork quality on team performance, learning and work satisfaction in agile software teams, and whether this effect differs from that of traditional software teams. A survey was administered to 477 respondents from 71 agile software teams in 26 companies and analyzed using structural equation modeling. A positive effect of teamwork quality on team performance was found when team members and team leaders rated team performance. In contrast, a negligible effect was found when product owners rated team performance. The effect of teamwork quality on team members´ learning and work satisfaction was strongly positive, but was only rated by the team members. Despite claims of the importance of teamwork in agile teams, this study did not find teamwork quality to be higher than in a similar survey on traditional teams. The effect of teamwork quality on team performance was only marginally greater for the agile teams than for the traditional teams.

2016/12/01/

2016

10.1016/j.jss.2016.09.028

ScienceDirect

A survey of the use of crowdsourcing in software engineering

Mao, Ke

Capra, Licia

Harman, Mark

Jia, Yue

Journal of Systems and Software

The term ‘crowdsourcing’ was initially introduced in 2006 to describe an emerging distributed problem-solving model by online workers. Since then it has been widely studied and practiced to support software engineering. In this paper we provide a comprehensive survey of the use of crowdsourcing in software engineering, seeking to cover all literature on this topic. We first review the definitions of crowdsourcing and derive our definition of Crowdsourcing Software Engineering together with its taxonomy. Then we summarise industrial crowdsourcing practice in software engineering and corresponding case studies. We further analyse the software engineering domains, tasks and applications for crowdsourcing and the platforms and stakeholders involved in realising Crowdsourced Software Engineering solutions. We conclude by exposing trends, open issues and opportunities for future research on Crowdsourced Software Engineering.

2017/04/01/

2017

10.1016/j.jss.2016.09.015

ScienceDirect

Continuous software engineering: A roadmap and agenda

Fitzgerald, Brian

Stol, Klaas-Jan

Journal of Systems and Software

Throughout its short history, software development has been characterized by harmful disconnects between important activities such as planning, development and implementation. The problem is further exacerbated by the episodic and infrequent performance of activities such as planning, testing, integration and releases. Several emerging phenomena reflect attempts to address these problems. For example, Continuous Integration is a practice which has emerged to eliminate discontinuities between development and deployment. In a similar vein, the recent emphasis on DevOps recognizes that the integration between software development and its operational deployment needs to be a continuous one. We argue a similar continuity is required between business strategy and development, BizDev being the term we coin for this. These disconnects are even more problematic given the need for reliability and resilience in the complex and data-intensive systems being developed today. We identify a number of continuous activities which together we label as ‘Continuous *’ (i.e. Continuous Star) which we present as part of an overall roadmap for Continuous Software engineering. We argue for a continuous (but not necessarily rapid) software engineering delivery pipeline. We conclude the paper with a research agenda.

2017/01/01/

2017

10.1016/j.jss.2015.06.063

ScienceDirect

Challenges and success factors for large-scale agile transformations: A systematic literature review

Dikert, Kim

Paasivaara, Maria

Lassenius, Casper

Journal of Systems and Software

Agile methods have become an appealing alternative for companies striving to improve their performance, but the methods were originally designed for small and individual teams. This creates unique challenges when introducing agile at scale, when development teams must synchronize their activities, and there might be a need to interface with other organizational units. In this paper we present a systematic literature review on how agile methods and lean software development has been adopted at scale, focusing on reported challenges and success factors in the transformation. We conducted a systematic literature review of industrial large-scale agile transformations. Our keyword search found 1875 papers. We included 52 publications describing 42 industrial cases presenting the process of taking large-scale agile development into use. Almost 90% of the included papers were experience reports, indicating a lack of sound academic research on the topic. We identified 35 reported challenges grouped into nine categories, and 29 success factors, grouped into eleven categories. The most salient success factor categories were management support, choosing and customizing the agile model, training and coaching, and mindset and alignment.

2016/09/01/

2016

10.1016/j.jss.2016.06.013

ScienceDirect

DeepTest: automated testing of deep-neural-network-driven autonomous cars

Tian, Yuchi

Pei, Kexin

Jana, Suman

Ray, Baishakhi

ICSE '18

Recent advances in Deep Neural Networks (DNNs) have led to the development of DNN-driven autonomous cars that, using sensors like camera, LiDAR, etc., can drive without any human intervention. Most major manufacturers including Tesla, GM, Ford, BMW, and Waymo/Google are working on building and testing different types of autonomous vehicles. The lawmakers of several US states including California, Texas, and New York have passed new legislation to fast-track the process of testing and deployment of autonomous vehicles on their roads. However, despite their spectacular progress, DNNs, just like traditional software, often demonstrate incorrect or unexpected corner-case behaviors that can lead to potentially fatal collisions. Several such real-world accidents involving autonomous cars have already happened including one which resulted in a fatality. Most existing testing techniques for DNN-driven vehicles are heavily dependent on the manual collection of test data under different driving conditions which become prohibitively expensive as the number of test conditions increases. In this paper, we design, implement, and evaluate DeepTest, a systematic testing tool for automatically detecting erroneous behaviors of DNN-driven vehicles that can potentially lead to fatal crashes. First, our tool is designed to automatically generated test cases leveraging real-world changes in driving conditions like rain, fog, lighting conditions, etc. DeepTest systematically explore different parts of the DNN logic by generating test inputs that maximize the numbers of activated neurons. DeepTest found thousands of erroneous behaviors under different realistic driving conditions (e.g., blurring, rain, fog, etc.) many of which lead to potentially fatal crashes in three top performing DNNs in the Udacity self-driving car challenge.

Gothenburg, Sweden

Proceedings of the 40th International Conference on Software Engineering

2018/05/27/

2018

10.1145/3180155.3180220

ACM Digital Library

Groute: An Asynchronous Multi-GPU Programming Model for Irregular Computations

Ben-Nun, Tal

Sutton, Michael

Pai, Sreepathi

Pingali, Keshav

ACM SIGPLAN Notices

Nodes with multiple GPUs are becoming the platform of choice for high-performance computing. However, most applications are written using

bulk-synchronous programming models, which may not be optimal for irregular algorithms that benefit from low-latency, asynchronous communication. This paper proposes constructs for asynchronous multi-GPU programming, and describes their implementation in a thin runtime environment called Groute. Groute also implements common collective operations and distributed work-lists, enabling the development of irregular applications without substantial programming effort. We demonstrate that this approach achieves state-of-the-art performance and exhibits strong scaling for a suite of irregular applications on 8-GPU and heterogeneous systems, yielding over 7x speedup for some algorithms.

2017/01/26/

2017

10.1145/3155284.3018756

August 2017

Gunrock: a high-performance graph processing library on the GPU

Wang, Yangzihao

Davidson, Andrew

Pan, Yuechao

Wu, Yuduo

Riffel, Andy

Owens, John D.

PPoPP '16

For large-scale graph analytics on the GPU, the irregularity of data access/control flow and the complexity of programming GPUs have been two significant challenges for developing a programmable high-performance graph library. "Gunrock," our high-level bulk-synchronous graph-processing system targeting the GPU, takes a new approach to abstracting GPU graph analytics: rather than designing an abstraction around computation, Gunrock instead implements a novel data-centric abstraction centered on operations on a vertex or edge frontier. Gunrock achieves a balance between performance and expressiveness by coupling high-performance GPU computing primitives and optimization strategies with a high-level programming model that allows programmers to quickly develop new graph primitives with small code size and minimal GPU programming knowledge. We evaluate Gunrock on five graph primitives (BFS, BC, SSSP, CC, and PageRank) and show that Gunrock has on average at least an order of magnitude speedup over Boost and PowerGraph, comparable performance to the fastest GPU hardwired primitives, and better performance than any other GPU high-level graph library.

Barcelona, Spain

Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming

2016/02/27/

2016

10.1145/2851141.2851145

ACM Digital Library

Superneurons: dynamic GPU memory management for training deep neural networks

Wang, Linnan

Ye, Jinmian

Zhao, Yiyang

Wu, Wei

Li, Ang

Song, Shuaiwen Leon

Xu, Zenglin

Kraska, Tim

PPoPP '18

Going deeper and wider in neural architectures improves their accuracy, while the limited GPU DRAM places an undesired restriction on the network design domain. Deep Learning (DL) practitioners either need to change to less desired network architectures, or nontrivially dissect a network across multiGPUs. These distract DL practitioners from concentrating on their original machine learning tasks. We present SuperNeurons: a dynamic GPU memory scheduling runtime to enable the network training far beyond the GPU DRAM capacity. SuperNeurons features 3 memory optimizations, Liveness Analysis, Unified Tensor Pool, and Cost-Aware Recomputation; together they effectively reduce the network-wide peak memory usage down to the maximal memory usage among layers. We also address the performance issues in these memory-saving techniques. Given the limited GPU DRAM, SuperNeurons not only provisions the necessary memory for the training, but also dynamically allocates the memory for convolution workspaces to achieve the high performance. Evaluations against Caffe, Torch, MXNet and TensorFlow have demonstrated that SuperNeurons trains at least 3.2432 deeper network than current ones with the leading performance. Particularly, SuperNeurons can train ResNet2500 that has 104 basic network layers on a 12GB K40c.

Vienna, Austria

Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming

2018/02/10/

2018

10.1145/3178487.3178491

ACM Digital Library

Keep calm and react with foresight: strategies for low-latency and energy-efficient elastic data stream processing

De Matteis, Tiziano

Mencagli, Gabriele

ACM SIGPLAN Notices

This paper addresses the problem of designing scaling strategies for elastic data stream processing. Elasticity allows applications to rapidly change their configuration on-the-fly (e.g., the amount of used resources) in response to dynamic workload fluctuations. In this work we face this problem by adopting the Model Predictive Control technique, a control-theoretic method aimed at finding the optimal application configuration along a limited prediction horizon in the future by solving an online optimization problem. Our control strategies are designed to address latency constraints, using Queueing Theory models, and energy consumption by changing the number of used cores and the CPU frequency through the Dynamic Voltage and Frequency Scaling (DVFS) support available in the modern multicore CPUs. The proactive capabilities, in addition to the latency- and energy-awareness, represent the novel features of our approach. To validate our methodology, we develop a thorough set of experiments on a high-frequency trading application. The results demonstrate the high-degree of flexibility and configurability of our approach, and show the effectiveness of our elastic scaling strategies compared with existing state-of-the-art techniques used in similar scenarios.

2016/02/27/

2016

10.1145/3016078.2851148

August 2016

S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters

Awan, Ammar Ahmad

Hamidouche, Khaled

Hashmi, Jahanzeb Maqbool

Panda, Dhabaleswar K.

PPoPP '17

Availability of large data sets like ImageNet and massively parallel computation support in modern HPC devices like NVIDIA GPUs have fueled a renewed interest in Deep Learning (DL) algorithms. This has triggered the development of DL frameworks like Caffe, Torch, TensorFlow, and CNTK. However, most DL frameworks have been limited to a single node. In order to scale out DL frameworks and bring HPC capabilities to the DL arena, we propose, S-Caffe; a scalable and distributed Caffe adaptation for modern multi-GPU clusters. With an in-depth analysis of new requirements brought forward by the DL frameworks and limitations of current communication runtimes, we present a co-design of the Caffe framework and the MVAPICH2-GDR MPI runtime. Using the co-design methodology, we modify Caffe's workflow to maximize the overlap of computation and communication with multi-stage data propagation and gradient aggregation schemes. We bring DL-Awareness to the MPI runtime by proposing a hierarchical reduction design that benefits from CUDA-Aware features and provides up to a massive 133x speedup over OpenMPI and 2.6x speedup over MVAPICH2 for 160 GPUs. S-Caffe successfully scales up to 160 K-80 GPUs for GoogLeNet (ImageNet) with a speedup of 2.5x over 32 GPUs. To the best of our knowledge, this is the first framework that scales up to 160 GPUs. Furthermore, even for single node training, S-Caffe shows an improvement of 14\% and 9\% over Nvidia's optimized Caffe for 8 and 16 GPUs, respectively. In addition, S-Caffe achieves up to 1395 samples per second for the AlexNet model, which is comparable to the performance of Microsoft CNTK.

Austin, Texas, USA

Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming

2017/01/26/

2017

10.1145/3018743.3018769

ACM Digital Library

Automated Extraction of Conceptual Models from User Stories via NLP

Robeer, Marcel

Lucassen, Garm

van der Werf, Jan Martijn E. M.

Dalpiaz, Fabiano

Brinkkemper, Sjaak

2016 IEEE 24th International Requirements Engineering Conference (RE)

Natural language (NL) is still the predominant notation that practitioners use to represent software requirements. Albeit easy to read, NL does not readily highlight key concepts and relationships such as dependencies and conflicts. This contrasts with the inherent capability of graphical conceptual models to visualize a given domain in a holistic fashion. In this paper, we propose to automatically derive conceptual models from a concise and widely adopted NL notation for requirements: user stories. Due to their simplicity, we hypothesize that our approach can improve on the low accuracy of previous works. We present an algorithm that combines state-of-the-art heuristics and that is implemented in our Visual Narrator tool. We evaluate our work on two case studies wherein we obtained promising precision and recall results (between 80% and 92%). The creators of the user stories perceived the generated models as a useful artifact to communicate and discuss the requirements, especially for team members who are not yet familiar with the project.

2016 IEEE 24th International Requirements Engineering Conference (RE)

2016/09//

2016

10.1109/RE.2016.40

IEEE Xplore

A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution

Guzman, Emitza

Ibrahim, Mohamed

Glinz, Martin

2017 IEEE 25th International Requirements Engineering Conference (RE)

Twitter is one of the most popular social networks. Previous research found that users employ Twitter to communicate about software applications via short messages, commonly referred to as tweets, and that these tweets can be useful for requirements engineering and software evolution. However, due to their large number—in the range of thousands per day for popular applications—a manual analysis is unfeasible.In this work we present ALERTme, an approach to automatically classify, group and rank tweets about software applications. We apply machine learning techniques for automatically classifying tweets

requesting improvements, topic modeling for grouping semantically related tweets and a weighted function for ranking tweets according to specific attributes, such as content category, sentiment and number of retweets. We ran our approach on 68,108 collected tweets from three software applications and compared its results against software practitioners' judgement. Our results show that ALERTme is an effective approach for filtering, summarizing and ranking tweets about software applications. ALERTme enables the exploitation of Twitter as a feedback channel for information relevant to software evolution, including end-user requirements.

2017 IEEE 25th International Requirements Engineering Conference (RE)

2017/09//

2017

10.1109/RE.2017.88

IEEE Xplore

SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews

Johann, Timo

Stanik, Christoph

Alizadeh B., Alireza M.

Maalej, Walid

2017 IEEE 25th International Requirements Engineering Conference (RE)

A main advantage of app stores is that they aggregate important information created by both developers and users. In the app store product pages, developers usually describe and maintain the features of their apps. In the app reviews, users comment these features. Recent studies focused on mining app features either as described by developers or as reviewed by users. However, extracting and matching the features from the app descriptions and the reviews is essential to bear the app store advantages, e.g. allowing analysts to identify which app features are actually being reviewed and which are not. In this paper, we propose SAFE, a novel uniform approach to extract app features from the single app pages, the single reviews and to match them. We manually build 18 part-of-speech patterns and 5 sentence patterns that are frequently used in text referring to app features. We then apply these patterns with several text pre-and post-processing steps. A major advantage of our approach is that it does not require large training and configuration data. To evaluate its accuracy, we manually extracted the features mentioned in the pages and reviews of 10 apps. The extraction precision and recall outperformed two state-of-the-art approaches. For well-maintained app pages such as for Google Drive our approach has a precision of 87% and on average 56% for 10 evaluated apps. SAFE also

matches 87% of the features extracted from user reviews to those extracted from the app descriptions.

2017 IEEE 25th International Requirements Engineering Conference (RE)

2017/09//

2017

10.1109/RE.2017.71

IEEE Xplore

A Needle in a Haystack: What Do Twitter Users Say about Software?

Guzman, Emitza

Alkadhi, Rana

Seyff, Norbert

2016 IEEE 24th International Requirements Engineering Conference (RE)

Users of the Twitter microblogging platform share a vast amount of information about various topics through short messages on a daily basis. Some of these so called tweets include information that is relevant for software companies and could, for example, help requirements engineers to identify user needs. Therefore, tweets have the potential to aid in the continuous evolution of software applications. Despite the existence of such relevant tweets, little is known about their number and content. In this paper we report on the results of an exploratory study in which we analyzed the usage characteristics, content and automatic classification potential of tweets about software applications by using descriptive statistics, content analysis and machine learning techniques. Although the manual search of relevant information within the vast stream of tweets can be compared to looking for a needle in a haystack, our analysis shows that tweets provide a valuable input for software companies. Furthermore, our results demonstrate that machine learning techniques have the capacity to identify and harvest relevant information automatically.

2016 IEEE 24th International Requirements Engineering Conference (RE)

2016/09//

2016

10.1109/RE.2016.67

IEEE Xplore

Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning

Kurtanović, Zijad

Maalej, Walid

2017 IEEE 25th International Requirements Engineering Conference (RE)

In this paper, we take up the second RE17 data challenge: the identification of requirements types using the "Quality attributes (NFR)" dataset provided. We studied how accurately we can automatically classify requirements as functional (FR) and non-functional (NFR) in the dataset with supervised machine learning. Furthermore, we assessed how accurately we can identify various types of NFRs, in particular usability, security, operational, and performance requirements. We developed and evaluated a supervised machine learning approach employing meta-data, lexical, and syntactical features. We employed under-and over-sampling strategies to handle the imbalanced classes in the dataset and cross-validated the classifiers using precision, recall, and F1 metrics in a series of experiments based on the Support Vector Machine classifier algorithm. We achieve a precision and recall up to 92% for automatically identifying FRs and NFRs. For the identification of specific NFRs, we achieve the highest precision and recall for security and performance NFRs with 92% precision and 90% recall. We discuss the most discriminating features of FRs and NFRs as well as the sampling strategies used with an additional dataset and their impact on the classification accuracy.

2017 IEEE 25th International Requirements Engineering Conference (RE)

2017/09//

2017

10.1109/RE.2017.82

IEEE Xplore

Automatic Verification of C and Java Programs: SV-COMP 2019

Beyer, Dirk

Beyer, Dirk

Huisman, Marieke

Kordon, Fabrice

Steffen, Bernhard

Lecture Notes in Computer Science

This report describes the 2019 Competition on Software Verification (SV-COMP), the 8\(^{\text {th}}\) edition of a series of comparative evaluations of fully automatic software verifiers for C programs, and now also for Java programs. The competition provides a snapshot of the current state of the art in the area, and has a strong focus on replicability of its results. The repository of benchmark verification

tasks now supports a new, more flexible format for task definitions (based on YAML), which was a precondition for conveniently benchmarking Java programs in the same controlled competition setting that was successfully applied in the previous years. The competition was based on 10 522 verification tasks for C programs and 368 verification tasks for Java programs. Each verification task consisted of a program and a property (reachability, memory safety, overflows, termination). SV-COMP 2019 had 31 participating verification systems from 14 countries.

Cham

Tools and Algorithms for the Construction and Analysis of Systems

2019///

2019

10.1007/978-3-030-17502-3_9

Springer Link

Software Verification with Validation of Results

Beyer, Dirk

Legay, Axel

Margaria, Tiziana

Lecture Notes in Computer Science

This report describes the 2017 Competition on Software Verification (SV-COMP), the 6\(^{\text {th}}\) edition of the annual thorough comparative evaluation of fully-automatic software verifiers. The goal is to reflect the current state of the art in software verification in terms of effectiveness and efficiency. The major achievement of the 6\(^{\text {th}}\) edition of SV-COMP is that the verification results were validated in most categories. The verifiers have to produce verification witnesses, which contain hints that a validator can later use to reproduce the verification result. The answer of a verifier counts only if the validator confirms the verification result. SV-COMP uses two independent, publicly available witness validators. For 2017, a new category structure was introduced that now orders the verification tasks according to the property to verify on the top level, and by the type of programs (e.g., which kind of data types are used) on a second level. The categories Overflows and Termination were heavily extended, and the category SoftwareSystems now contains also verification tasks from the software system BusyBox. The competition used 8 908 verification tasks that each consisted of a C program and a property (reachability, memory safety, termination). SV-COMP 2017 had 32 participating verification systems from 12 countries.

Berlin, Heidelberg

Tools and Algorithms for the Construction and Analysis of Systems

2017///

2017

10.1007/978-3-662-54580-5_20

Springer Link

Scaling Enumerative Program Synthesis via Divide and Conquer

Alur, Rajeev

Radhakrishna, Arjun

Udupa, Abhishek

Legay, Axel

Margaria, Tiziana

Lecture Notes in Computer Science

Given a semantic constraint specified by a logical formula, and a syntactic constraint specified by a context-free grammar, the Syntax-Guided Synthesis (SyGuS) problem is to find an expression that satisfies both the syntactic and semantic constraints. An enumerative approach to solve this problem is to systematically generate all expressions from the syntactic space with some pruning, and has proved to be surprisingly competitive in the newly started competition of SyGuS solvers. It performs well on small to medium sized benchmarks, produces succinct expressions, and has the ability to generalize from input-output examples. However, its performance degrades drastically with the size of the smallest solution. To overcome this limitation, in this paper we propose an alternative approach to solve SyGuS instances.The key idea is to employ a divide-and-conquer approach by separately enumerating (a) smaller expressions that are correct on subsets of inputs, and (b) predicates that distinguish these subsets. These expressions and predicates are then combined using decision trees to obtain an expression that is correct on all inputs. We view the problem of combining expressions and predicates as a multi-label decision tree learning problem. We propose a novel technique of associating a probability distribution over the set of labels that a sample can be labeled with. This enables us to use standard information-gain based heuristics to learn compact decision trees.We report a prototype implementation eusolver. Our tool is able to match the running times and the succinctness of solutions of both standard enumerative solver and the latest white-box solvers on most benchmarks from the SyGuS competition. In the 2016 edition of the SyGuS competition, eusolver placed first in the general track and the programming-by-examples track, and placed second in the linear integer arithmetic track.

Berlin, Heidelberg

Tools and Algorithms for the Construction and Analysis of Systems

2017///

2017

10.1007/978-3-662-54577-5_18

Springer Link

Reliable and Reproducible Competition Results with BenchExec and Witnesses (Report on SV-COMP 2016)

Beyer, Dirk

Chechik, Marsha

Raskin, Jean-François

Lecture Notes in Computer Science

The 5\(^{\text {th}}\) Competition on Software Verification (SV-COMP 2016) continues the tradition of a thorough comparative evaluation of fully-automatic software verifiers. This report presents the results of the competition and includes a special section that describes how SV-COMP ensures that the experiments are reliably executed, precisely measured, and organized such that the results can be reproduced later. SV-COMP uses BenchExec for controlling and measuring the verification runs, and requires violation witnesses in an exchangeable format, whenever a verifier reports that a property is violated. Each witness was validated by two independent and publicly-available witness validators. The tables report the state of the art in software verification in terms of effectiveness and efficiency. The competition used 6 661 verification tasks that each consisted of a C program and a property (reachability, memory safety, termination). SV-COMP 2016 had 35 participating verification systems (22 in 2015) from 16 countries.

Berlin, Heidelberg

Tools and Algorithms for the Construction and Analysis of Systems

2016///

2016

10.1007/978-3-662-49674-9_55

Springer Link

Feature-Guided Black-Box Safety Testing of Deep Neural Networks

Wicker, Matthew

Huang, Xiaowei

Kwiatkowska, Marta

Beyer, Dirk

Huisman, Marieke

Lecture Notes in Computer Science

Despite the improved accuracy of deep neural networks, the discovery of adversarial examples has raised serious safety concerns. Most existing approaches for crafting adversarial examples necessitate some knowledge (architecture, parameters, etc) of the network at hand. In this paper, we focus on image classifiers and propose a feature-guided black-box approach to test the safety of deep neural networks that requires no such knowledge. Our algorithm employs object detection techniques such as SIFT (Scale Invariant Feature Transform) to extract features from an image. These features are converted into a mutable saliency distribution, where high probability is assigned to pixels that affect the composition of the image with respect to the human visual system. We formulate the crafting of adversarial examples as a two-player turn-based stochastic game, where the first player's objective is to minimise the distance to an adversarial example by manipulating the features, and the second player can be cooperative, adversarial, or random. We show that, theoretically, the two-player game can converge to the optimal strategy, and that the optimal strategy represents a globally minimal adversarial image. For Lipschitz networks, we also identify conditions that provide safety guarantees that no adversarial examples exist. Using Monte Carlo tree search we gradually explore the game state space to search for adversarial examples. Our experiments show that, despite the black-box setting, manipulations guided by a perception-based saliency distribution are competitive with state-of-the-art methods that rely on white-box saliency matrices or sophisticated optimization procedures. Finally, we show how our method can be used to evaluate robustness of neural networks in safety-critical applications such as traffic sign recognition in self-driving cars.

Cham

Tools and Algorithms for the Construction and Analysis of Systems

2018///

2018

10.1007/978-3-319-89960-2_22

Springer Link

Microservices migration patterns

Balalaie, Armin

Heydarnoori, Abbas

Jamshidi, Pooyan

Tamburri, Damian A.

Lynn, Theo

Software: Practice and Experience

Microservices architectures are becoming the defacto standard for building continuously deployed systems. At the same time, there is a substantial growth in the demand for migrating on-premise legacy applications to the cloud. In this context, organizations tend to migrate their traditional architectures into cloud-native architectures using microservices. This article reports a set of migration and rearchitecting design patterns that we have empirically identified and collected from industrial-scale software migration projects. These migration patterns can help information technology organizations plan their migration projects toward microservices more efficiently and effectively. In addition, the proposed patterns facilitate the definition of migration plans by pattern composition. Qualitative empirical research is used to evaluate the validity of the proposed patterns. Our findings suggest that the proposed patterns are evident in other architectural refactoring and migration projects and strong candidates for effective patterns in system migrations.

2018///

2018

10.1002/spe.2608

Wiley Online Library

ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers

Piraghaj, Sareh Fotuhi

Dastjerdi, Amir Vahid

Calheiros, Rodrigo N.

Buyya, Rajkumar

Software: Practice and Experience

Containers are increasingly gaining popularity and becoming one of the major deployment models in cloud environments. To evaluate the performance of scheduling and allocation policies in containerized cloud data centers, there is a need for evaluation environments that support scalable and repeatable experiments. Simulation techniques provide repeatable and controllable environments, and hence, they serve as a powerful tool for such purpose. This paper introduces ContainerCloudSim, which provides support for modeling and simulation of containerized cloud computing environments. We developed a simulation architecture

for containerized clouds and implemented it as an extension of CloudSim. We described a number of use cases to demonstrate how one can plug in and compare their container scheduling and provisioning policies in terms of energy efficiency and SLA compliance. Our system is highly scalable as it supports simulation of large number of containers, given that there are more containers than virtual machines in a data center. Copyright © 2016 John Wiley & Sons, Ltd.

2017///

2017

10.1002/spe.2422

Wiley Online Library

Better bitmap performance with Roaring bitmaps

Chambi, Samy

Lemire, Daniel

Kaser, Owen

Godin, Robert

Software: Practice and Experience

Bitmap indexes are commonly used in databases and search engines. By exploiting bit-level parallelism, they can significantly accelerate queries. However, they can use much memory, and thus, we might prefer compressed bitmap indexes. Following Oracle's lead, bitmaps are often compressed using run-length encoding (RLE). Building on prior work, we introduce the Roaring compressed bitmap format: it uses packed arrays for compression instead of RLE. We compare it to two high-performance RLE-based bitmap encoding techniques: Word Aligned Hybrid compression scheme and Compressed ‘n’ Composable Integer Set. On synthetic and real data, we find that Roaring bitmaps (1) often compress significantly better (e.g., 2×) and (2) are faster than the compressed alternatives (up to 900× faster for intersections). Our results challenge the view that RLE-based bitmap compression is best. Copyright © 2015 John Wiley & Sons, Ltd.

2016///

2016

10.1002/spe.2325

Wiley Online Library

The anatomy of big data computing

Kune, Raghavendra

Konugurthi, Pramod Kumar

Agarwal, Arun

Chillarige, Raghavendra Rao

Buyya, Rajkumar

Software: Practice and Experience

Advances in information technology and its widespread growth in several areas of business, engineering, medical, and scientific studies are resulting in information/data explosion. Knowledge discovery and decision-making from such rapidly growing voluminous data are a challenging task in terms of data organization and processing, which is an emerging trend known as big data computing, a new paradigm that combines large-scale compute, new data-intensive techniques, and mathematical models to build data analytics. Big data computing demands a huge storage and computing for data curation and processing that could be delivered from on-premise or clouds infrastructures. This paper discusses the evolution of big data computing, differences between traditional data warehousing and big data, taxonomy of big data computing and underpinning technologies, integrated platform of big data and clouds known as big data clouds, layered architecture and components of big data cloud, and finally open-technical challenges and future directions. Copyright © 2015 John Wiley & Sons, Ltd.

2016///

2016

10.1002/spe.2374

Wiley Online Library

iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments

Gupta, Harshit

Vahid Dastjerdi, Amir

Ghosh, Soumya K.

Buyya, Rajkumar

Software: Practice and Experience

Internet of Things (IoT) aims to bring every object (eg, smart cameras, wearable, environmental sensors, home appliances, and vehicles) online, hence generating massive volume of data that can overwhelm storage systems and data analytics applications. Cloud computing offers services at the infrastructure level that can scale to IoT storage and processing requirements. However, there are applications such as health monitoring and emergency response that require low latency, and delay that is

caused by transferring data to the cloud and then back to the application can seriously impact their performances. To overcome this limitation, Fog computing paradigm has been proposed, where cloud services are extended to the edge of the network to decrease the latency and network congestion. To realize the full potential of Fog and IoT paradigms for real-time analytics, several challenges need to be addressed. The first and most critical problem is designing resource management techniques that determine which modules of analytics applications are pushed to each edge device to minimize the latency and maximize the throughput. To this end, we need an evaluation platform that enables the quantification of performance of resource management policies on an IoT or Fog computing infrastructure in a repeatable manner. In this paper we propose a simulator, called iFogSim, to model IoT and Fog environments and measure the impact of resource management techniques in latency, network congestion, energy consumption, and cost. We describe two case studies to demonstrate modeling of an IoT environment and comparison of resource management policies. Moreover, scalability of the simulation toolkit of RAM consumption and execution time is verified under different circumstances.

2017///

2017

10.1002/spe.2509

Wiley Online Library

On-demand Developer Documentation

Robillard, Martin P.

Marcus, Andrian

Treude, Christoph

Bavota, Gabriele

Chaparro, Oscar

Ernst, Neil

Gerosa, Marco Aurélio

Godfrey, Michael

Lanza, Michele

Linares-Vásquez, Mario

Murphy, Gail C.

Moreno, Laura

Shepherd, David

Wong, Edmund

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

We advocate for a paradigm shift in supporting the information needs of developers, centered around the concept of automated on-demand developer documentation. Currently, developer information needs are fulfilled by asking experts or consulting documentation. Unfortunately, traditional documentation practices are inefficient because of, among others, the manual nature of its creation and the gap between the creators and consumers. We discuss the major challenges we face in realizing such a paradigm shift, highlight existing research that can be leveraged to this end, and promote opportunities for increased convergence in research on software documentation.

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

2017/09//

2017

10.1109/ICSME.2017.17

IEEE Xplore

Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing

Linares-Vásquez, Mario

Moran, Kevin

Poshyvanyk, Denys

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Mobile app development involves a unique set of challenges including device fragmentation and rapidly evolving platforms, making testing a difficult task. The design space for a comprehensive mobile testing strategy includes features, inputs, potential contextual app states, and large combinations of devices and underlying platforms. Therefore, automated testing is an essential activity of the development process. However, current state of the art of automated testing tools for mobile apps posses limitations that has driven a preference for manual testing in practice. As of today, there is no comprehensive automated solution for mobile testing that overcomes fundamental issues such as automated oracles, history awareness in test cases, or automated evolution of test cases.In this perspective paper we survey the current state of the art in terms of the frameworks, tools, and services available to developers to aid in mobile testing, highlighting present shortcomings. Next, we provide commentary on current key challenges that restrict the possibility of a comprehensive, effective, and practical automated

testing solution. Finally, we offer our vision of a comprehensive mobile app testing framework, complete with research agenda, that is succinctly summarized along three principles: Continuous, Evolutionary and Large-scale (CEL).

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

2017/09//

2017

10.1109/ICSME.2017.27

IEEE Xplore

CCLearner: A Deep Learning-Based Clone Detection Approach

Li, Liuqing

Feng, He

Zhuang, Wenjie

Meng, Na

Ryder, Barbara

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Programmers produce code clones when developing software. By copying and pasting code with or without modification, developers reuse existing code to improve programming productivity. However, code clones present challenges to software maintenance: they may require consistent application of the same or similar bug fixes or program changes to multiple code locations. To simplify the maintenance process, various tools have been proposed to automatically detect clones [1], [2], [3], [4], [5], [6]. Some tools tokenize source code, and then compare the sequence or frequency of tokens to reveal clones [1], [3], [4], [5]. Some other tools detect clones using tree-matching algorithms to compare the Abstract Syntax Trees (ASTs) of source code [2], [6]. In this paper, we present CCLEARNER, the first solely token-based clone detection approach leveraging deep learning. CCLEARNER extracts tokens from known method-level code clones and nonclones to train a classifier, and then uses the classifier to detect clones in a given codebase. To evaluate CCLEARNER, we reused BigCloneBench [7], an existing large benchmark of real clones. We used part of the benchmark for training and the other part for testing, and observed that CCLEARNER effectively detected clones. With the same data set, we conducted the first systematic comparison experiment between CCLEARNER and three popular clone detection tools. Compared with the approaches not using deep learning, CCLEARNER achieved competitive clone detection effectiveness with low time cost.

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

2017/09//

2017

10.1109/ICSME.2017.46

IEEE Xplore

Understanding the Factors That Impact the Popularity of GitHub Repositories

Borges, Hudson

Hora, Andre

Valente, Marco Tulio

2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Software popularity is a valuable information to modern open source developers, who constantly want to know if their systems are attracting new users, if new releases are gaining acceptance, or if they are meeting user's expectations. In this paper, we describe a study on the popularity of software systems hosted at GitHub, which is the world's largest collection of open source software. GitHub provides an explicit way for users to manifest their satisfaction with a hosted repository: the stargazers button. In our study, we reveal the main factors that impact the number of stars of GitHub projects, including programming language and application domain. We also study the impact of new features on project popularity. Finally, we identify four main patterns of popularity growth, which are derived after clustering the time series representing the number of stars of 2,279 popular GitHub repositories. We hope our results provide valuable insights to developers and maintainers, which could help them on building and evolving systems in a competitive software market.

2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)

2016/10//

2016

10.1109/ICSME.2016.31

IEEE Xplore

DroidRA: taming reflection to support whole-program analysis of Android apps

Li, Li

Bissyandé, Tegawendé F.

Octeau, Damien

Klein, Jacques

ISSTA 2016

Android developers heavily use reflection in their apps for legitimate reasons, but also significantly for hiding malicious actions. Unfortunately, current state-of-the-art static analysis tools for Android are challenged by the presence of reflective calls which they usually ignore. Thus, the results of their security analysis, e.g., for private data leaks, are inconsistent given the measures taken by malware writers to elude static detection. We propose the DroidRA instrumentation-based approach to address this issue in a non-invasive way. With DroidRA, we reduce the resolution of reflective calls to a composite constant propagation problem. We leverage the COAL solver to infer the values of reflection targets and app, and we eventually instrument this app to include the corresponding traditional Java call for each reflective call. Our approach allows to boost an app so that it can be immediately analyzable, including by such static analyzers that were not reflection-aware. We evaluate DroidRA on benchmark apps as well as on real-world apps, and demonstrate that it can allow state-of-the-art tools to provide more sound and complete analysis results.

Saarbrücken, Germany

Proceedings of the 25th International Symposium on Software Testing and Analysis

2016/07/18/

2016

10.1145/2931037.2931044

ACM Digital Library

A learning-to-rank based fault localization approach using likely invariants

B. Le, Tien-Duy

Lo, David

Le Goues, Claire

Grunske, Lars

ISSTA 2016

Debugging is a costly process that consumes much of developer time and energy. To help reduce debugging effort, many studies have proposed various fault localization approaches. These approaches take as input a set of test cases (some failing, some passing) and produce a ranked list of program elements that are likely to be the root cause of the failures (i.e., failing test cases). In this work, we propose Savant, a new fault localization approach that employs a learning-to-rank strategy, using likely invariant diffs and suspiciousness scores as features, to rank methods based on their likelihood to be a root cause of a failure. Savant has four steps: method clustering & test case selection, invariant mining, feature extraction, and method ranking. At the end of these four steps, Savant produces a short ranked list of potentially buggy methods. We have evaluated Savant on 357 real-life bugs from 5 programs from the Defects4J benchmark. Out of these bugs, averaging over 100 repeated trials with different seeds to randomly break ties, we find that on average Savant can identify correct buggy methods for 63.03, 101.72, and 122 bugs at top 1, 3, and 5 positions in the ranked lists that Savant produces. We have compared Savant against several state-of-the-art fault localization baselines that work on program spectra. We show that Savant can successfully locate 57.73%, 56.69%, and 43.13% more bugs at top 1, top 3, and top 5 positions than the best performing baseline, respectively.

Saarbrücken, Germany

Proceedings of the 25th International Symposium on Software Testing and Analysis

2016/07/18/

2016

10.1145/2931037.2931049

ACM Digital Library

ASTOR: a program repair library for Java (demo)

Martinez, Matias

Monperrus, Martin

ISSTA 2016

During the last years, the software engineering research community has proposed approaches for automatically repairing software bugs. Unfortunately, many software artifacts born from this research are not available for repairing Java programs. To-reimplement those approaches from scratch is costly. To facilitate experimental replications and comparative evaluations, we present Astor, a publicly available program repair library that includes the implementation of three notable repair approaches (jGenProg, jKali and jMutRepair). We envision that the research community will use Astor for setting up comparative evaluations and explore the design space of automatic repair for Java. Astor offers researchers ways to implement new repair approaches or to modify

existing ones. Astor repairs in total 33 real bugs from four large open source projects.

Saarbrücken, Germany

Proceedings of the 25th International Symposium on Software Testing and Analysis

2016/07/18/

2016

10.1145/2931037.2948705

ACM Digital Library

Practitioners' expectations on automated fault localization

Kochhar, Pavneet Singh

Xia, Xin

Lo, David

Li, Shanping

ISSTA 2016

Software engineering practitioners often spend significant amount of time and effort to debug. To help practitioners perform this crucial task, hundreds of papers have proposed various fault localization techniques. Fault localization helps practitioners to find the location of a defect given its symptoms (e.g., program failures). These localization techniques have pinpointed the locations of bugs of various systems of diverse sizes, with varying degrees of success, and for various usage scenarios. Unfortunately, it is unclear whether practitioners appreciate this line of research. To fill this gap, we performed an empirical study by surveying 386 practitioners from more than 30 countries across 5 continents about their expectations of research in fault localization. In particular, we investigated a number of factors that impact practitioners' willingness to adopt a fault localization technique. We then compared what practitioners need and the current state-of-research by performing a literature review of papers on fault localization techniques published in ICSE, FSE, ESEC-FSE, ISSTA, TSE, and TOSEM in the last 5 years (2011-2015). From this comparison, we highlight the directions where researchers need to put effort to develop fault localization techniques that matter to practitioners.

Saarbrücken, Germany

Proceedings of the 25th International Symposium on Software Testing and Analysis

2016/07/18/

2016

10.1145/2931037.2931051

ACM Digital Library

Sapienz: multi-objective automated testing for Android applications

Mao, Ke

Harman, Mark

Jia, Yue

ISSTA 2016

We introduce Sapienz, an approach to Android testing that uses multi-objective search-based testing to automatically explore and optimise test sequences, minimising length, while simultaneously maximising coverage and fault revelation. Sapienz combines random fuzzing, systematic and search-based exploration, exploiting seeding and multi-level instrumentation. Sapienz significantly outperforms (with large effect size) both the state-of-the-art technique Dynodroid and the widely-used tool, Android Monkey, in 7/10 experiments for coverage, 7/10 for fault detection and 10/10 for fault-revealing sequence length. When applied to the top 1,000 Google Play apps, Sapienz found 558 unique, previously unknown crashes. So far we have managed to make contact with the developers of 27 crashing apps. Of these, 14 have confirmed that the crashes are caused by real faults. Of those 14, six already have developer-confirmed fixes.

Saarbrücken, Germany

Proceedings of the 25th International Symposium on Software Testing and Analysis

2016/07/18/

2016

10.1145/2931037.2931054

ACM Digital Library

Formalizing and appling compliance patterns for business process compliance

Elgammal, Amal

Turetken, Oktay

van den Heuvel, Willem-Jan

Papazoglou, Mike

Software & Systems Modeling

Today’s enterprises demand a high degree of compliance of business processes to meet diverse regulations and legislations. Several industrial studies have shown that compliance management is a daunting task, and organizations are still struggling and spending billions of dollars annually to ensure and prove their compliance. In this paper, we introduce a comprehensive compliance management framework with a main focus on design-time compliance management as a first step towards a preventive lifetime compliance support. The framework enables the automation of compliance-related activities that are amenable to automation, and therefore can significantly reduce the expenditures spent on compliance. It can help experts to carry out their work more efficiently, cut the time spent on tedious manual activities, and reduce potential human errors. An evident candidate compliance activity for automation is the compliance checking, which can be achieved by utilizing formal reasoning and verification techniques. However, formal languages are well known of their complexity as only versed users in mathematical theories and formal logics are able to use and understand them. However, this is generally not the case with business and compliance practitioners. Therefore, in the heart of the compliance management framework, we introduce the Compliance Request Language (CRL), which is formally grounded on temporal logic and enables the abstract pattern-based specification of compliance requirements. CRL constitutes a series of compliance patterns that spans three structural facets of business processes; control flow, employed resources and temporal perspectives. Furthermore, CRL supports the specification of compensations and non-monotonic requirements, which permit the relaxation of some compliance requirements to handle exceptional situations. An integrated tool suite has been developed as an instantiation artefact, and the validation of the approach is undertaken in several directions, which includes internal validity, controlled experiments, and functional testing.

2016/02/01/

2016

10.1007/s10270-014-0395-3

Springer Link

Clafer: unifying class and feature modeling

Bąk, Kacper

Diskin, Zinovy

Antkiewicz, Michał

Czarnecki, Krzysztof

Wąsowski, Andrzej

Software & Systems Modeling

We present Clafer (class, feature, reference), a class modeling language with first-class support for feature modeling. We designed Clafer as a concise notation for meta-models, feature models, mixtures of meta- and feature models (such as components with options), and models that couple feature models and meta-models via constraints (such as mapping feature configurations to component configurations or model templates). Clafer allows arranging models into multiple specialization and extension layers via constraints and inheritance. We identify several key mechanisms allowing a meta-modeling language to express feature models concisely. Clafer unifies basic modeling constructs, such as class, association, and property, into a single construct, called clafer. We provide the language with a formal semantics built in a structurally explicit way. The resulting semantics explains the meaning of hierarchical models whereby properties can be arbitrarily nested in the presence of inheritance and feature modeling constructs. The semantics also enables building consistent automated reasoning support for the language: To date, we implemented three reasoners for Clafer based on Alloy, Z3 SMT, and Choco3 CSP solvers. We show that Clafer meets its design objectives using examples and by comparing to other languages.

2016/07/01/

2016

10.1007/s10270-014-0441-1

Springer Link

Model transformation intents and their properties

Lúcio, Levi

Amrani, Moussa

Dingel, Juergen

Lambers, Leen

Salay, Rick

Selim, Gehan M. K.

Syriani, Eugene

Wimmer, Manuel

Software & Systems Modeling

The notion of model transformation intent is proposed to capture the purpose of a transformation. In this paper, a framework for the description of model transformation intents is defined, which includes, for instance, a description of properties a model transformation has to satisfy to qualify as a suitable realization of an intent. Several common model transformation intents are identified, and the framework is used to describe six of them in detail. A case study from the

automotive industry is used to demonstrate the usefulness of the proposed framework for identifying crucial properties of model transformations with different intents and to illustrate the wide variety of model transformation intents that an industrial model-driven software development process typically encompasses.

2016/07/01/

2016

10.1007/s10270-014-0429-x

Springer Link

Scalable process discovery and conformance checking

Leemans, Sander J. J.

Fahland, Dirk

van der Aalst, Wil M. P.

Software & Systems Modeling

Considerable amounts of data, including process events, are collected and stored by organisations nowadays. Discovering a process model from such event data and verification of the quality of discovered models are important steps in process mining. Many discovery techniques have been proposed, but none of them combines scalability with strong quality guarantees. We would like such techniques to handle billions of events or thousands of activities, to produce sound models (without deadlocks and other anomalies), and to guarantee that the underlying process can be rediscovered when sufficient information is available. In this paper, we introduce a framework for process discovery that ensures these properties while passing over the log only once and introduce three algorithms using the framework. To measure the quality of discovered models for such large logs, we introduce a model–model and model–log comparison framework that applies a divide-and-conquer strategy to measure recall, fitness, and precision. We experimentally show that these discovery and measuring techniques sacrifice little compared to other algorithms, while gaining the ability to cope with event logs of 100,000,000 traces and processes of 10,000 activities on a standard computer.

2018/05/01/

2018

10.1007/s10270-016-0545-x

Springer Link

Industry 4.0 as a Cyber-Physical System study

Mosterman, Pieter J.

Zander, Justyna

Software & Systems Modeling

Advances in computation and communication are taking shape in the form of the Internet of Things, Machine-to-Machine technology, Industry 4.0, and Cyber-Physical Systems (CPS). The impact on engineering such systems is a new technical systems paradigm based on ensembles of collaborating embedded software systems. To successfully facilitate this paradigm, multiple needs can be identified along three axes: (i) online configuring an ensemble of systems, (ii) achieving a concerted function of collaborating systems, and (iii) providing the enabling infrastructure. This work focuses on the collaborative function dimension and presents a set of concrete examples of CPS challenges. The examples are illustrated based on a pick and place machine that solves a distributed version of the Towers of Hanoi puzzle. The system includes a physical environment, a wireless network, concurrent computing resources, and computational functionality such as, service arbitration, various forms of control, and processing of streaming video. The pick and place machine is of medium-size complexity. It is representative of issues occurring in industrial systems that are coming online. The entire study is provided at a computational model level, with the intent to contribute to the model-based research agenda in terms of design methods and implementation technologies necessary to make the next generation systems a reality.

2016/02/01/

2016

10.1007/s10270-015-0493-x

Springer Link

MadMax: surviving out-of-gas conditions in Ethereum smart contracts

Grech, Neville

Kong, Michael

Jurisevic, Anton

Brent, Lexi

Scholz, Bernhard

Smaragdakis, Yannis

Proceedings of the ACM on Programming Languages

Ethereum is a distributed blockchain platform, serving as an ecosystem for smart contracts: full-fledged inter-communicating programs that capture the transaction logic of an account. Unlike programs in mainstream languages, a gas limit restricts the execution of an Ethereum

smart contract: execution proceeds as long as gas is available. Thus, gas is a valuable resource that can be manipulated by an attacker to provoke unwanted behavior in a victim's smart contract (e.g., wasting or blocking funds of said victim). Gas-focused vulnerabilities exploit undesired behavior when a contract (directly or through other interacting contracts) runs out of gas. Such vulnerabilities are among the hardest for programmers to protect against, as out-of-gas behavior may be uncommon in non-attack scenarios and reasoning about it is far from trivial.

In this paper, we classify and identify gas-focused vulnerabilities, and present MadMax: a static program analysis technique to automatically detect gas-focused vulnerabilities with very high confidence. Our approach combines a control-flow-analysis-based decompiler and declarative program-structure queries. The combined analysis captures high-level domain-specific concepts (such as "dynamic data structure storage" and "safely resumable loops") and achieves high precision and scalability. MadMax analyzes the entirety of smart contracts in the Ethereum blockchain in just 10 hours (with decompilation timeouts in 8% of the cases) and flags contracts with a (highly volatile) monetary value of over $2.8B as vulnerable. Manual inspection of a sample of flagged contracts shows that 81% of the sampled warnings do indeed lead to vulnerabilities, which we report on in our experiment.

2018/10/24/

2018

10.1145/3276486

DOI.org (Crossref)

The tensor algebra compiler

Kjolstad, Fredrik

Kamil, Shoaib

Chou, Stephen

Lugato, David

Amarasinghe, Saman

Proceedings of the ACM on Programming Languages

Tensor algebra is a powerful tool with applications in machine learning, data analytics, engineering and the physical sciences. Tensors are often sparse and compound operations must frequently be computed in a single kernel for performance and to save memory. Programmers are left to write kernels for every operation of interest, with different mixes of dense and sparse tensors in different formats. The combinations are infinite, which makes it impossible to manually implement and optimize them all. This paper introduces the first compiler technique to automatically

generate kernels for any compound tensor algebra operation on dense and sparse tensors. The technique is implemented in a C++ library called taco. Its performance is competitive with best-in-class hand-optimized kernels in popular libraries, while supporting far more tensor operations.

2017/10/12/

2017

10.1145/3133901

DOI.org (Crossref)

RustBelt: securing the foundations of the Rust programming language

Jung, Ralf

Jourdan, Jacques-Henri

Krebbers, Robbert

Dreyer, Derek

Proceedings of the ACM on Programming Languages

Rust is a new systems programming language that promises to overcome the seemingly fundamental tradeoff between high-level safety guarantees and low-level control over resource management. Unfortunately, none of Rust's safety claims have been formally proven, and there is good reason to question whether they actually hold. Specifically, Rust employs a strong, ownership-based type system, but then extends the expressive power of this core type system through libraries that internally use unsafe features. In this paper, we give the first formal (and machine-checked) safety proof for a language representing a realistic subset of Rust. Our proof is extensible in the sense that, for each new Rust library that uses unsafe features, we can say what verification condition it must satisfy in order for it to be deemed a safe extension to the language. We have carried out this verification for some of the most important libraries that are used throughout the Rust ecosystem.

2018/01//

2018

10.1145/3158154

DOI.org (Crossref)

An abstract domain for certifying neural networks

Singh, Gagandeep

Gehr, Timon

Püschel, Markus

Vechev, Martin

Proceedings of the ACM on Programming Languages

We present a novel method for scalable and precise certification of deep neural networks. The key technical insight behind our approach is a new abstract domain which combines floating point polyhedra with intervals and is equipped with abstract transformers specifically tailored to the setting of neural networks. Concretely, we introduce new transformers for affine transforms, the rectified linear unit (ReLU), sigmoid, tanh, and maxpool functions.

We implemented our method in a system called DeepPoly and evaluated it extensively on a range of datasets, neural architectures (including defended networks), and specifications. Our experimental results indicate that DeepPoly is more precise than prior work while scaling to large networks.

We also show how to combine DeepPoly with a form of abstraction refinement based on trace partitioning. This enables us to prove, for the first time, the robustness of the network when the input image is subjected to complex perturbations such as rotations that employ linear interpolation.

2019/01/02/

2019

10.1145/3290354

DOI.org (Crossref)

code2vec: learning distributed representations of code

Alon, Uri

Zilberstein, Meital

Levy, Omer

Yahav, Eran

Proceedings of the ACM on Programming Languages

We present a neural model for representing snippets of code as continuous distributed vectors (``code embeddings''). The main idea is to represent a code snippet as a single fixed-length code vector, which can be used to predict semantic properties of the snippet. To this end, code is first decomposed to a collection of paths in its abstract syntax tree. Then, the network learns the atomic representation of each path while simultaneously learning how to aggregate a set of them.

We demonstrate the effectiveness of our approach by using it to predict a method's name from the vector representation of its

body. We evaluate our approach by training a model on a dataset of 12M methods. We show that code vectors trained on this dataset can predict method names from files that were unobserved during training. Furthermore, we show that our model learns useful method name vectors that capture semantic similarities, combinations, and analogies.

A comparison of our approach to previous techniques over the same dataset shows an improvement of more than 75%, making it the first to successfully predict method names based on a large, cross-project corpus. Our trained model, visualizations and vector similarities are available as an interactive online demo at http://code2vec.org. The code, data and trained models are available at https://github.com/tech-srl/code2vec.

2019/01/02/

2019

10.1145/3290353

DOI.org (Crossref)

Future Trends in Software Engineering Research for Mobile Apps

Nagappan, Meiyappan

Shihab, Emad

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

There has been tremendous growth in the use of mobile devices over the last few years. This growth has fueled the development of millions of software applications for these mobile devices often called as 'apps'. Current estimates indicate that there are hundreds of thousands of mobile app developers. As a result, in recent years, there has been an increasing amount of software engineering research conducted on mobile apps to help such mobile app developers. In this paper, we discuss current and future research trends within the framework of the various stages in the software development life-cycle: requirements (including non-functional), design and development, testing, and maintenance. While there are several non-functional requirements, we focus on the topics of energy and security in our paper, since mobile apps are not necessarily built by large companies that can afford to get experts for solving these two topics. For the same reason we also discuss the monetizing aspects of a mobile app at the end of the paper. For each topic of interest, we first present the recent advances done in these stages and then we present the challenges present in current work, followed by the future opportunities and the risks present in pursuing such research.

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

2016/03//

2016

10.1109/SANER.2016.88

IEEE Xplore

An Investigation into the Use of Common Libraries in Android Apps

Li, Li

Bissyandé, Tegawendé F.

Klein, Jacques

Le Traon, Yves

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

The packaging model of Android apps requires the entire code necessary for the execution of an app to be shipped into one single apk file. Thus, an analysis of Android apps often visits code which is not part of the functionality delivered by the app. Such code is often contributed by the common libraries which are used pervasively by all apps. Unfortunately, Android analyses, e.g., for piggybacking detection and malware detection, can produce inaccurate results if they do not take into account the case of library code, which constitute noise in app features. Despite some efforts on investigating Android libraries, the momentum of Android research has not yet produced a complete set of common libraries to further support in-depth analysis of Android apps. In this paper, we leverage a dataset of about 1.5 million apps from Google Play to harvest potential common libraries, including advertisement libraries. With several steps of refinements, we finally collect by far the largest set of 1,113 libraries supporting common functionality and 240 libraries for advertisement. We use the dataset to investigates several aspects of Android libraries, including their popularity and their proportion in Android app code. Based on these datasets, we have further performed several empirical investigations to confirm the motivations behind our work.

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

2016/03//

2016

10.1109/SANER.2016.52

IEEE Xplore

Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software

Beller, Moritz

Bholanath, Radjino

McIntosh, Shane

Zaidman, Andy

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

The use of automatic static analysis has been a software engineering best practice for decades. However, we still do not know a lot about its use in real-world software projects: How prevalent is the use of Automated Static Analysis Tools (ASATs) such as FindBugs and JSHint? How do developers use these tools, and how does their use evolve over time? We research these questions in two studies on nine different ASATs for Java, JavaScript, Ruby, and Python with a population of 122 and 168,214 open-source projects. To compare warnings across the ASATs, we introduce the General Defect Classification (GDC) and provide a grounded-theory-derived mapping of 1,825 ASAT-specific warnings to 16 top-level GDC classes. Our results show that ASAT use is widespread, but not ubiquitous, and that projects typically do not enforce a strict policy on ASAT use. Most ASAT configurations deviate slightly from the default, but hardly any introduce new custom analyses. Only a very small set of default ASAT analyses is widely changed. Finally, most ASAT configurations, once introduced, never change. If they do, the changes are small and have a tendency to occur within one day of the configuration's initial introduction.

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)

2016/03//

2016

10.1109/SANER.2016.105

IEEE Xplore

Under-optimized smart contracts devour your money

Chen, Ting

Li, Xiaoqi

Luo, Xiapu

Zhang, Xiaosong

2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)

Smart contracts are full-fledged programs that run on blockchains (e.g., Ethereum, one of the most popular blockchains). In Ethereum, gas (in

Ether, a cryptographic currency like Bitcoin) is the execution fee compensating the computing resources of miners for running smart contracts. However, we find that under-optimized smart contracts cost more gas than necessary, and therefore the creators or users will be overcharged. In this work, we conduct the first investigation on Solidity, the recommended compiler, and reveal that it fails to optimize gas-costly programming patterns. In particular, we identify 7 gas-costly patterns and group them to 2 categories. Then, we propose and develop GASPER, a new tool for automatically locating gas-costly patterns by analyzing smart contracts' bytecodes. The preliminary results on discovering 3 representative patterns from 4,240 real smart contracts show that 93.5%, 90.1% and 80% contracts suffer from these 3 patterns, respectively.

2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)

2017/02//

2017

10.1109/SANER.2017.7884650

IEEE Xplore

History Driven Program Repair

Effective automated program repair techniques have great potential to reduce the costs of debugging and maintenance. Previously proposed automated program repair (APR) techniques often follow a generate-and-validate and test-case-driven procedure: They first randomly generate a large pool of fix candidates and then exhaustively validate the quality of the candidates by testing them against existing or provided test suites. Unfortunately, many real-world bugs cannot be repaired by existing techniques even after more than 12 hours of computation in a multi-core cloud environment. More work is needed to advance the capabilities of modern APR techniques. We propose a new technique that utilizes the wealth of bug fixesacross projects in their development history to effectively guide and drive a programrepair process. Our main insight is that recurring bug fixes are common inreal-world applications, and that previously-appearing fix patterns canprovide useful guidance to an automated repair technique. Based on this insight, our technique first automaticallymines bug fix patterns from the history of many projects. We then employ existingmutation operators to generate fix candidates for a given buggy program. Candidates that match frequently occurring historical bug fixes are consideredmore likely to be relevant, and we thus give them priority inthe random search process. Finally, candidates thatpass all the previously failed test cases are recommended as likely fixes. We compare our technique against existinggenerate-and-validate and test-driven APR approaches using 90 bugs from 5 Javaprograms. The experiment results show that our technique can producegood-quality fixes for many more bugs as compared to the baselines, while beingreasonably computationally efficient: it takes less than 20minutes, on average, to correctly fix a bug.

https://ieeexplore.ieee.org/document/7476644/

'Cause I'm strong enough: Reasoning about consistency choices in distributed systems

Gotsman, Alexey

Yang, Hongseok

Ferreira, Carla

Najafzadeh, Mahsa

Shapiro, Marc

POPL '16

Large-scale distributed systems often rely on replicated databases that allow a programmer to request different data consistency guarantees for different operations, and thereby control their performance. Using such databases is far from trivial: requesting stronger consistency in too many places may hurt performance, and requesting it in too few places may violate correctness. To help programmers in this task, we propose the first proof rule for establishing that a particular choice of consistency guarantees for various operations on a replicated database is enough to ensure the preservation of a given data integrity invariant. Our rule is modular: it allows reasoning about the behaviour of every operation separately under some assumption on the behaviour of other operations. This leads to simple reasoning, which we have automated in an SMT-based tool. We present a nontrivial proof of soundness of our rule and illustrate its use on several examples.

St. Petersburg, FL, USA

Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages

2016/01/11/

2016

10.1145/2837614.2837625

ACM Digital Library

Modelling the ARMv8 architecture, operationally: concurrency and ISA

Flur, Shaked

Gray, Kathryn E.

Pulte, Christopher

Sarkar, Susmit

Sezgin, Ali

Maranget, Luc

Deacon, Will

Sewell, Peter

POPL '16

In this paper we develop semantics for key aspects of the ARMv8 multiprocessor architecture: the concurrency model and much of the 64-bit application-level instruction set (ISA). Our goal is to clarify what the range of architecturally allowable behaviour is, and thereby to support future work on formal verification, analysis, and testing of concurrent ARM software and hardware. Establishing such models with high confidence is intrinsically difficult: it involves capturing the vendor's architectural intent, aspects of which (especially for concurrency) have not previously been precisely defined. We therefore first develop a concurrency model with a microarchitectural flavour, abstracting from many hardware implementation concerns but still close to hardware-designer intuition. This means it can be discussed in detail with ARM architects. We then develop a more abstract model, better suited for use as an architectural specification, which we prove sound w.r.t.~the first. The instruction semantics involves further difficulties, handling the mass of detail and the subtle intensional information required to interface to the concurrency model. We have a novel ISA description language, with a lightweight dependent type system, letting us do both with a rather direct representation of the ARM reference manual instruction descriptions. We build a tool from the combined semantics that lets one explore, either interactively or exhaustively, the full range of architecturally allowed behaviour, for litmus tests and (small) ELF executables. We prove correctness of some optimisations needed for tool performance. We validate the models by discussion with ARM staff, and by comparison against ARM hardware behaviour, for ISA single- instruction tests and concurrent litmus tests.

St. Petersburg, FL, USA

Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages

2016/01/11/

2016

10.1145/2837614.2837615

ACM Digital Library

Learning invariants using decision trees and implication counterexamples

Garg, Pranav

Neider, Daniel

Madhusudan, P.

Roth, Dan

ACM SIGPLAN Notices

Inductive invariants can be robustly synthesized using a learning model where the teacher is a program verifier who instructs the learner through concrete program configurations, classified as positive, negative, and implications. We propose the first learning algorithms in this model with implication counter-examples that are based on machine learning techniques. In particular, we extend classical decision-tree learning algorithms in machine learning to handle implication samples, building new scalable ways to construct small decision trees using statistical measures. We also develop a decision-tree learning algorithm in this model that is guaranteed to converge to the right concept (invariant) if one exists. We implement the learners and an appropriate teacher, and show that the resulting invariant synthesis is efficient and convergent for a large suite of programs.

2016/04/08/

2016

10.1145/2914770.2837664

DOI.org (Crossref)

Dependent types and multi-monadic effects in F*

Swamy, Nikhil

Hriţcu, Cătălin

Keller, Chantal

Rastogi, Aseem

Delignat-Lavaud, Antoine

Forest, Simon

Bhargavan, Karthikeyan

Fournet, Cédric

Strub, Pierre-Yves

Kohlweiss, Markulf

Zinzindohoue, Jean-Karim

Zanella-Béguelin, Santiago

POPL '16

We present a new, completely redesigned, version of F*, a language that works both as a proof assistant as well as a general-purpose, verification-oriented, effectful programming language. In support of these complementary roles, F* is a dependently typed, higher-order, call-by-value language with _primitive_ effects including state, exceptions, divergence and IO. Although primitive, programmers choose the granularity at which to specify effects by equipping each effect with a monadic, predicate transformer semantics. F* uses this to efficiently compute weakest preconditions and discharges the resulting proof obligations using a combination of SMT solving and manual proofs. Isolated from the effects, the core of F* is a language of pure functions used to write specifications and proof terms---its consistency is maintained by a semantic termination check based on a well-founded order. We evaluate our design on more than 55,000 lines of F* we have authored in the last year, focusing on three main case studies. Showcasing its use as a general-purpose programming language, F* is programmed (but not verified) in F*, and bootstraps in both OCaml and F#. Our experience confirms F*'s pay-as-you-go cost model: writing idiomatic ML-like code with no finer specifications imposes no user burden. As a verification-oriented language, our most significant evaluation of F* is in verifying several key modules in an implementation of the TLS-1.2 protocol standard. For the modules we considered, we are able to prove more properties, with fewer annotations using F* than in a prior verified implementation of TLS-1.2. Finally, as a proof assistant, we discuss our use of F* in mechanizing the metatheory of a range of lambda calculi, starting from the simply typed lambda calculus to System F-omega and even micro-F*, a sizeable fragment of F* itself---these proofs make essential use of F*'s flexible combination of SMT automation and constructive proofs, enabling a tactic-free style of programming and proving at a relatively large scale.

St. Petersburg, FL, USA

Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages

2016/01/11/

2016

10.1145/2837614.2837655

ACM Digital Library

TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration

Beller, Moritz

Gousios, Georgios

Zaidman, Andy

2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

Continuous Integration (CI) has become a best practice of modern software development. Thanks in part to its tight integration with GitHub, Travis CI has emerged as arguably the most widely used CI platform for Open-Source Software (OSS) development. However, despite its prominent role in Software Engineering in practice, the benefits, costs, and implications of doing CI are all but clear from an academic standpoint. Little research has been done, and even less was of quantitative nature. In order to lay the groundwork for data-driven research on CI, we built TravisTorrent, travistorrent.testroots.org, a freely available data set based on Travis CI and GitHub that provides easy access to hundreds of thousands of analyzed builds from more than 1,000 projects. Unique to TravisTorrent is that each of its 2,640,825 Travis builds is synthesized with meta data from Travis CI's API, the results of analyzing its textual build log, a link to the GitHub commit which triggered the build, and dynamically aggregated project data from the time of commit extracted through GHTorrent.

2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

2017/05//

2017

10.1109/MSR.2017.24

IEEE Xplore

A look at the dynamics of the JavaScript package ecosystem

Wittern, Erik

Suter, Philippe

Rajagopalan, Shriram

MSR '16

The node package manager (npm) serves as the frontend to a large repository of JavaScript-based software packages, which foster the development of currently huge amounts of server-side Node. js and client-side JavaScript applications. In a span of 6 years since its inception, npm has grown to become one of the largest software ecosystems, hosting more than 230, 000 packages, with hundreds of millions of package installations every week. In this paper, we examine the npm ecosystem from two complementary perspectives: 1) we look at package descriptions, the dependencies among them, and download metrics, and 2) we look at the use of npm packages in publicly available applications hosted on GitHub. In both perspectives, we consider historical data, providing us with a unique view on the evolution of the ecosystem. We present analyses that provide insights into the ecosystem's growth and activity, into conflicting measures of package

popularity, and into the adoption of package versions over time. These insights help understand the evolution of npm, design better package recommendation engines, and can help developers understand how their packages are being used.

Austin, Texas

Proceedings of the 13th International Conference on Mining Software Repositories

2016/05/14/

2016

10.1145/2901739.2901743

ACM Digital Library

Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub

Beller, Moritz

Gousios, Georgios

Zaidman, Andy

2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

Continuous Integration (CI) has become a best practice of modern software development. Yet, at present, we have a shortfall of insight into the testing practices that are common in CI-based software development. In particular, we seek quantifiable evidence on how central testing is to the CI process, how strongly the project language influences testing, whether different integration environments are valuable and if testing on the CI can serve as a surrogate to local testing in the IDE. In an analysis of 2,640,825 Java and Ruby builds on Travis CI, we find that testing is the single most important reason why builds fail. Moreover, the programming language has a strong influence on both the number of executed tests, their run time, and proneness to fail. The use of multiple integration environments leads to 10% more failures being caught at build time. However, testing on Travis CI does not seem an adequate surrogate for running tests locally in the IDE. To further research on Travis CI with GitHub, we introduce TravisTorrent.

2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

2017/05//

2017

10.1109/MSR.2017.62

IEEE Xplore

Notice of Retraction: Does Refactoring of Test Smells Induce Fixing Flaky Tests?

Palomba, Fabio

Zaidman, Andy

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Retracted.

2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)

2017/09//

2017

10.1109/ICSME.2017.12

IEEE Xplore

Appendix 2 – Details of the European research activities

The following table lists the RIAs and IAs funded under the previous topics listed, which are of relevance and interest for the research roadmap that SWForum is defining.

Table 4. European funded projects related to the topics of the research roadmaps

Topic	Acronym	Project Details
ICT-16-2018 Software Technologies	FASTEN, Fine-Grained Analysis of Software Ecosystems as Networks	Type, reference: IA, 825328. Description: The aim is to make software ecosystems more robust by making package management more intelligent. This objective relies on the Fine-Grained Call Graph (FGCG) method that serves to analyze i) security vulnerability propagation, ii) licensing compliance, and iii) dependency risk profiles and processing. To facilitate adoption, FASTEN applies the approach to popular package managers for Java, C, and Python programming languages. Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/825328 Web: https://www.fasten-project.eu/
ICT-16-2018 Software Technologies	DECODER, DEveloper COmpanion for Documented and annotatEd code Reference	Type, reference: RIA, 824231. Description: It consists of a methodology and an open source Integrated Development Environment to improve the efficiency of software development and maintenance, and to assure the quality of software in medium criticality systems, such as in IoT, cloud computing and HPC. The IDE combines information from different sources through formal and semi-formal models to deliver software project intelligence that shortens the learning curve of software programmers and maintainers and increases their productivity. With this approach developers deliver high quality code that are more secure and better aligned with requirements and maintainers know immediately what has been done, how and with which tools. Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/824231 Web: https://www.decoder-project.eu/
ICT-16-2018 Software Technologies	UNICORE, A Common Code Base and Toolkit for Deployment of Applications to Secure and Reliable Virtual Execution Environments	Type, reference: IA, 825377. Description: The goal is to create a common code base and toolkit for the deployment of applications to secure and reliable execution environments. This will facilitate software developers to easily build and quickly deploy the smallest lightweight virtual machines (unikernels). Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/824231 Web: https://unicore-project.eu/

Topic	Acronym	Project Details
ICT-16-2018 Software Technologies	RADON, Rational decomposition and orchestration for serverless computing	Type, reference: RIA, 825040. Description: The project aims at developing a DevOps framework to address the creation and management of microservices-based applications that can optimally exploit serverless computing technologies. The final goal is to broaden the adoption of serverless computing technologies within the European software industry. The methodology will strive to tackle complexity, harmonize the abstraction and actuation of action-trigger rules, avoid FaaS lock-in, and optimize decomposition and reuse through model-based FaaS-enabled development and orchestration. Dates: 1/01/2019 – 30/06/2021. Cordis info: https://cordis.europa.eu/project/id/825040 Web: https://radon-h2020.eu/
ICT-16-2018 Software Technologies	SODALITE, SOftware Defined AppLication Infrastructures managemenT and Engineering	Type, reference: RIA, 825480. Description: The project provides tools for developers and infrastructure operators that abstract their applications and infrastructure requirements to facilitate simpler and faster development and operation and execution of diverse applications over diverse software-defined, high-performance cloud infrastructures. Dates: 1/02/2019 – 31/01/2022. Cordis info: https://cordis.europa.eu/project/id/825480 Web: https://www.sodalite.eu/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	DEEPHEALTH, Deep-Learning and HPC to Boost Biomedical Applications for Health	Type, reference: IA, 825111. Description: The project delivers HPC power at the service of biomedical applications and applies deep learning (DL) techniques on vast and compound biomedical data sets, aiming to underpin new and more effective methods of diagnosis, monitoring and treatment of diseases. The resilient and scalable structure for the HPC + Big Data environment relies on two new libraries: the European Distributed Deep Learning Library (EDDLL) and the European Computer Vision Library (ECVL). Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/825111 Web: https://deephealth-project.eu/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	IoTWINS, Distributed Digital Twins for industrial SMEs: a big-data platform	Type, reference: IA, 857191. Description: The project is building testbeds for digital twins in the manufacturing and facility management sectors. The digital models are intended to integrate data from various sources such as data APIs, historical data, embedded sensors, and open data. This will give manufacturers an unprecedented view into how their

Topic	Acronym	Project Details
		products are performing. In facility management, the technology will be instrumental in improving the way buildings and their systems operate and in preventing prospective problems. Dates: 1/09/2019 – 31/08/2022. Cordis info: https://cordis.europa.eu/project/id/857191 Web: https://www.iotwins.eu/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	CYBELE, Fostering precision agriculture and livestock farming through secure access to large-scale HPC-enabled virtual industrial experimentation environment empowering scalable big data analytics	Type, reference: IA, 825355. Description: The project envisages demonstrating that high-performance computing (HPC), Big Data, cloud computing, and the Internet of Things can transform farming, reduce scarcity, increase food supply, and bring social, economic, and environmental benefits. CYBELE develops large-scale HPC-enabled testbeds and brings a distributed Big Data management architecture and a data management strategy that provide integrated access to large-scale data sets, a data and service-driven virtual HPC-enabled environment, and a bouquet of field-specific and generic services. Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/825355 Web: https://www.cybele-project.eu/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	EVOLVE, HPC and Cloud-enhanced Testbed for Extracting Value from Diverse Data at Large Scale	Type, reference: IA, 825061. Description: The project is developing a solution based on three pillars: technologies from high-performance computing, Big Data software and cloud deployment and access capabilities. It is also implementing a hardware platform with heterogeneous processing and storage, including the systems software to take advantage of the hardware value, a Big Data stack with connectors for acceleration and an innovative Cloud-like deployment, and access layer. The testbed will be demonstrated through pilots implemented in seven domains. Dates: 1/12/2018 – 30/11/2021. Cordis info: https://cordis.europa.eu/project/id/825061 Web: https://www.evolve-h2020.eu/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	LEXIS, Large-scale EXecution for Industry & Society	Type, reference: IA, 825532. Description: The project is building an advanced engineering platform at the confluence of HPC, Cloud and Big Data which will leverage large-scale, geographically distributed resources from existing HPC infrastructures, employ Big Data analytics solutions and augment them with Cloud services. It will be validated in the pilots - in the industrial and scientific sectors (Aeronautics, Earthquake and Tsunami, Weather and Climate).

Topic	Acronym	Project Details
		Dates: 1/01/2019 – 31/12/2021. Cordis info: https://cordis.europa.eu/project/id/825532 Web: https://lexis-project.eu/web/
ICT-11-2018-2019 HPC and Big Data enabled Large-scale Test-beds and Applications	INFINITECH, Tailored IoT & BigData Sandboxes and Testbeds for Smart, Autonomous and Personalized Services in the European Finance and Insurance Services Ecosystem	Type, reference: IA, 856632. Description: INFINITECH is developing novel data analytics and cloud computing solutions that will be combined with existing high-performance computing infrastructures and new hardware capabilities to bring computing power to the masses. It provides: Novel BigData/IoT technologies for seamless management and querying of all types of data, interoperable data analytics, blockchain-based data sharing, real-time analytics, as well as libraries of advanced AI algorithms; (b) regulatory tools incorporating various data governance capabilities and facilitating compliance to regulations; and (c) Nine novel and configurable testbeds & sandboxes, each one offering Open APIs and other resources for validating autonomous and personalized solutions, including a unique collection of data assets for finance/insurance. The results will be validated in the scope of 14 high impact pilots for the finance and insurance sectors. Dates: 1/10/2019 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/825480 Web: https://www.infinitech-h2020.eu/
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	UP2DATE, Intelligent software-UPDATE technologies for safe and secure mixed-criticality and high performance cyber physical systems	Type, reference: RIA, 871465. Description: The project aims to elaborate a new paradigm for safety and security (SASE) over-the-air software updates of Mixed-Criticality Cyber-Physical Systems (MCCPS) that is built around composability and modularity as main properties to enable a dynamic (post-deployment) validation of SASE properties. The novel architecture will be tested in the automotive and railway sectors. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871465 Web: https://h2020up2date.eu/
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	TEACHING, A computing toolkit for building efficient autonomous applications leveraging humanistic intelligence	Type, reference: RIA, 871385. Description: The project integrates Artificial Intelligence with fundamental concepts of security and dependability while including feedback from the AI-human-CPSoS interactions. TEACHING aims to develop a human-aware CPSoS for autonomous safety-critical applications, based on a distributed, energy-efficient and dependable Artificial Intelligence. The goal is to design a computing software and system supporting the development and deployment of adaptive and

Topic	Acronym	Project Details
		dependable CPSoS applications, allowing to exploit a sustainable human feedback to drive, optimize and personalize the provisioning of the offered services. TEACHING is expected to fundamentally impact the development of autonomous safety-critical systems, providing means to improve their safety, dependability, and overall acceptability. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871385 Web: https://teaching-h2020.eu/
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	AMPERE, A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation	Type, reference: RIA, 871669. Description: Parallel and heterogeneous platforms are difficult to program and even more to optimize for the multiple conflicting criteria imposed by applications, such as performance, energy efficiency, real-time response, resiliency, and fault tolerance. AMPERE addresses this challenge by incorporating model-driven engineering (MDE) as the key element for the construction of complex software architectures. AMPERE envisages the development of novel methods and tools for constructing accurate models of proposed systems in computing platforms. Thus, system constraints are efficiently dealt with, while ensuring performance targets are met. AMPERE's computing software will also help improve overall system efficiency along with fulfilment of non-functional requirements. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871669 Web: https://ampere-euproject.eu/
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	CPSoSaware, Cross-layer cognitive optimization tools & methods for the lifecycle support of dependable CPSoS	Type, reference: RIA, 871738. Description: For testing in the manufacturing industry and with semi-autonomous connected vehicles, the project is designing new models and software tools to allocate computational power/resources to the CPS end devices of the system by determining and autonomously generating what cyber-physical processes will be handled by a device's heterogeneous component (processor cores, GPUs, FPGA fabric, software stacks). Making maximum use of artificial intelligence to boost reliability, fault tolerance and security at system level, the CPSoSaware system is also intended to interact with human users through extended reality visual and touchable interfaces. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871738 Web: https://cpsosaware.eu/

Topic	Acronym	Project Details
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	ADEPTNESS, Design-Operation Continuum Methods for Testing and Deployment under Unforeseen Conditions for Cyber-Physical Systems of Systems	Type, reference: RIA, 871319. Description: The ADEPTNESS project seeks to investigate and to implement a streamlined and automatic workflow that makes methods and tools to be seamlessly used during design phases as well as in operation. This will be carried out by proposing a novel embedded microservices-based architecture for the context of CPSoS. Automatic and synchronised deployment techniques are also being investigated to improve the agility of the whole workflow that covers the design-operation continuum. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871319 Web: https://adeptness.eu/
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	1-SWARM, Integrated development and operations management framework for cyber-physical systems of systems under the paradigm of swarm intelligence	Type, reference: RIA, 871743. Description: The focus is on increasing the intelligence of cyber-physical systems at the edge of the networks, so that they show cognitive behaviour and have a high degree of autonomy. The project aims to develop a modular framework for designing robust CPSoS networks characterised by swarm intelligence that meet industrially accepted open standards. Dubbed the 'Swarm Intelligence DevOps Framework', it will aid researchers to engineer CPSoS for diverse scenarios: food packaging and material handling operations, automated guided vehicles in dynamic environments, and flocks of aerial drones used to monitor retail shops. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871743 Web:
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	SELENE, Self-monitored Dependable platform for High-Performance Safety-Critical Systems	Type, reference: RIA, 871467. Description: High-performance computing that employs commercial off-the-shelf components offers an alternative path to increasing the computational capability of safety-critical applications. Despite their potential in several domains, the use of these systems is limited due to the lack of certified, reliable hardware platforms. This project proposes a safety-critical cognitive computing platform (CCP) with self-aware and self-adaptive capabilities, which uses artificial intelligence techniques to maximise the efficiency of the safety-critical system and to adapt its behaviour in different domains such as automotive, space, avionics, robotics and factory automation. Dates: 1/12/2019 – 30/11/2022. Cordis info: https://cordis.europa.eu/project/id/871467 Web: https://www.selene-project.eu/

Topic	Acronym	Project Details
ICT-01-2019 Computing Technologies and engineering methods for cyber-physical systems of systems.	ADMORPH, Towards Adaptively Morphing Embedded Systems	Type, reference: RIA, 871259. Description: The project is implementing a holistic approach towards enabling the safety and security of cyber-physical systems to detect and prevent potential attacks. The new fault-tolerant design approaches will enable cyber-physical systems to automatically reconfigure themselves in case of component failures and cyberattacks. Project methods will be tested in three use cases: radar surveillance systems, autonomous operations for aircrafts, and transport management systems. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871259 Web: http://admorph.eu/
ICT-15-2019-2020 Cloud Computing	ACCORDION, Adaptive edge/cloud compute and network continuum over a heterogeneous sparse edge infrastructure to support nextgen applications	Type, reference: RIA, 871793. Description: The project considers that by associating edge computing with advanced technologies such as 5G, the EU will be able to capitalise on its local resource and infrastructure and bring benefit to its SMEs. The project settles a practical approach in connecting edge resources and infrastructures to support next-generation applications. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871793 Web: https://www.accordion-project.eu/
ICT-15-2019-2020 Cloud Computing	RAINBOW, A fog platform for secured IoT services	Type, reference: RIA, 871403. Description: The project is developing an open and secured fog computing platform that will advance the management of extensible, diverse and safe IoT services and cross-cloud applications. The project envisages extending fog computing to its real potential by supplying the development, composition, data, and network management to reach secure end-applications. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871403 Web: https://rainbow-h2020.eu/
ICT-15-2019-2020 Cloud Computing	MORPHEMIC, Cloud computing for multi-cloud deployments	Type, reference: RIA, 871643. Description: MORPHEMIC proposes a unique way of adapting and optimizing Cloud computing applications by introducing the novel concepts of polymorph architecture and proactive adaptation. The former is when a component can run in different technical forms, i.e. in a Virtual Machine (VM), in a container, as a big data job, or as serverless components, etc. The technical form of deployment is chosen during the optimization process to fulfil the user's requirements

Topic	Acronym	Project Details
		and needs. The quality of the deployment is measured by a user defined and application specific utility. Depending on the application's requirements and its current workload, its components could be deployed in various forms in different environments to maximize the utility of the application deployment and the satisfaction of the user. Proactive adaptation is not only based on the current execution context and conditions but aims to forecast future resource needs and possible deployment configurations. This ensures that adaptation can be done effectively and seamlessly for the users of the application. Dates: 1/01/2020 – 31/12/2022. Cordis info: https://cordis.europa.eu/project/id/871643 Web: https://www.morphemic.cloud/
ICT-15-2019-2020 Cloud Computing	PLEDGER, Performance optimization and edge computing orchestration for enhanced experience and Quality of Service	Type, reference: RIA, 871536. Description: Next-generation edge computing infrastructures should confront the new challenges faced today with the power offered by cloud infrastructures. The project aims to provide a new architectural model as well as a set of software tools that will prepare the future development of the next generation of edge computing. The project will allow edge computing providers to secure the stability and effective performance of the edge infrastructures. It will also allow edge computing users to understand the nature of their applications, research understandable quality of service metrics and optimise the competitiveness of their infrastructures. The project intends to introduce the set of tools in the application fields of manufacturing, mixed reality, and smart cities. Dates: 1/12/2019 – 30/11/2022. Cordis info: https://cordis.europa.eu/project/id/871536 Web: http://www.pledger-project.eu/
ICT-15-2019-2020 Cloud Computing	FogProtect, Protecting Sensitive Data in the Computing Continuum	Type, reference: RIA, 871525. Description: Thanks to the new technology, data produced in end tools such as smartphones and IoT devices are stored, developed, elaborated, and transmitted through fog nodes to cloud services. However, the protection of sensitive data in such a decentralised environment is not guaranteed. The project provides advanced design, technologies, and methods to secure end-to-end data protection along the computing continuum. The project integrates four technological innovations to create applicable solutions for data protection in smart cities, smart industries, and smart media. Dates: 1/01/2020 – 31/12/2022.

Topic	Acronym	Project Details
		Cordis info: https://cordis.europa.eu/project/id/871525 Web: https://fogprotect.eu/
ICT-50-2020 Software Technologies	FOCETA, Foundations for continuous engineering of trustworthy autonomy	Type, reference: RIA, 956123. Description: The project is developing the foundation for continuous engineering of trustworthy learning-enabled autonomous systems integrating data-driven and model-based engineering. A new system, grounded on open source tools with open data-exchange standards, will be demonstrated with the most demanding applications such as urban driving automation and intelligent medical devices, to prove viability, scalability, and safety. Dates: 1/10/2020 – 30/09/2023. Cordis info: https://cordis.europa.eu/project/id/956123 Web: http://www.foceta-project.eu/
	ELEGANT, Secure and Seamless Edge-to-Cloud Analytics	Type, reference: RIA, 957286. Description: The project aims to develop a novel software solution that addresses key challenges facing IoT and Big Data: interoperability, reliability, safety, and security. Some of the key innovations of the proposed framework are lightweight virtualisation, automatic code extraction compatible with IoT and Big Data frameworks, intelligent orchestration, dynamic code motion and advanced code verification and cybersecurity mechanisms. These should enable the seamless operation of end-to-end IoT/Big Data systems. Dates: 1/01/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/957286 Web: https://www.elegant-project.eu/
	XANDAR, X-by-Construction Design framework for Engineering Autonomous & Distributed Real-time Embedded Software Systems	Type, reference: RIA, 957210. Description: The next generation of networked embedded systems (ES) requires fast prototyping and high performance in addition to its key properties of reliability and safety. However, the dependence of the current autonomous systems trend on machine learning and artificial intelligence applications in combination with fail-operational requirements makes the verification and validation of ES a challenging endeavour. The project is delivering a mature software toolchain that fulfils the industrial requirements for rapid prototyping of interoperable and autonomous ES. A model-based system architecture to support novel automatic model synthesis and software parallelisation techniques will be used to achieve the objectives of a new real-time, safety- and security-by-construction paradigm.

Topic	Acronym	Project Details
		Dates: 1/01/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/957210 Web: https://xandar-project.eu/
	COSMOS, DevOps for Complex Cyber-physical Systems	Type, reference: RIA, 957254. Description: The objective is to enhance DevOps pipelines for the development of cyber-physical systems software. It will integrate more sophisticated validation and verification methods, which will comprise a mix of static code analysis correlated with issues and bug reports, automated test-case generation, runtime verification, hardware in the loop (HiL) testing and feedback from field devices. The project will also use machine learning, model-based testing, and search-based test generation. Dates: 1/01/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/957254 Web:
	VeriDevOps	Type, reference: RIA, 957212. Description: DevOps helps increase an organisation's ability to deliver applications and services at high velocity. It aims to shorten the systems development life cycle and provide continuous delivery with high-quality software. Current systems development practices are increasingly based on off-the-shelf and legacy components, which make such systems prone to security vulnerabilities. Since DevOps is promoting frequent software deliveries, verification artefacts should be updated in a timely fashion to cope with the pace of the process. The project plans to develop methods and tools that provide a faster feedback loop for verifying the security requirements – confidentiality, integrity, availability, authentication and authorisation – in large-scale cyber-physical systems. Dates: 1/10/2020 – 30/09/2023. Cordis info: https://cordis.europa.eu/project/id/957212 Web:
	PIACERE, Programming trustworthy Infrastructure As Code in a sEcuRE framework	Type, reference: RIA, 101000162. Description: The role of software management in operating infrastructures is increasing; automation, cloud, and software-defined networking boost the speed and security of operations. The DevOps philosophy already offers a set of practices and tools that combine software development and IT operations to shorten the development life cycle and provide continuous delivery. However, there is a need for tools to manage the whole life cycle of infrastructure as code

Topic	Acronym	Project Details
		(IaC), with a special focus on trustworthiness and security aspects throughout the IaC life cycle. The project is developing the tools, techniques and methods to allow organisations to develop and operate IaC through DevSecOps practices as they would do with traditional code. Dates: 1/12/2020 – 30/11/2023. Cordis info: https://cordis.europa.eu/project/id/101000162 Web: https://www.piacere-project.eu/
ICT-56-2020 Next Generation Internet of Things	ASSIST-IoT, Architecture for Scalable, Self-*, human-centric, Intelligent, Secure, and Tactile next generation IoT	Type, reference: RIA, 957258. Description: The project is developing the reference architecture in which intelligence can be distributed among nodes by implementing artificial intelligence and machine learning close to data generation and actuation, and hyperconnecting nodes, in the edge-cloud continuum, over softwarised smart networks. Dates: 1/11/2020 – 31/10/2023. Cordis info: https://cordis.europa.eu/project/id/957258 Web: https://assist-iot.eu/
ICT-56-2020 Next Generation Internet of Things	IntellIoT, Intelligent, distributed, human-centered and trustworthy IoT environments	Type, reference: RIA, 957218. Description: The project is developing a framework for intelligent IoT environments that execute semi-autonomous IoT applications, enabling a suite of novel use cases in which a human expert plays a key role in controlling and teaching the AI-enabled systems. Specifically, the project will focus on agriculture (tractors semi-autonomously operated in conjunction with drones), healthcare (patients monitored by sensors) and manufacturing (automated plants shared by multiple tenants who utilise machinery from third-party vendors). It will establish human-defined autonomy through distributed AI running on intelligent IoT devices. Dates: 1/10/2020 – 30/09/2023. Cordis info: https://cordis.europa.eu/project/id/957218 Web: https://intelliot.eu/
ICT-56-2020 Next Generation Internet of Things	IoT-NGIN, Next Generation IoT as part of Next Generation Internet	Type, reference: RIA, 957246. Description: The project introduces novel research and innovation concepts to establish itself as the 'engine' that will fuel the next generation IoT. It starts by uncovering a pattern based meta-architecture and optimises IoT/machine-to-machine and 5G/machine-cloud-machine communications by extending the edge cloud paradigm. Moreover, it enables user and self-aware autonomous IoT systems through privacy-preserving federated machine learning and ambient intelligence, with augmented reality support. Finally,

Topic	Acronym	Project Details
		IoT-NGIN researches towards distributed IoT cybersecurity and privacy. IoT-NGIN will be validated using dozens of heterogeneous devices, including drones and robots. Dates: 1/10/2020 – 30/09/2023. Cordis info: https://cordis.europa.eu/project/id/957246 Web: https://iot-ngin.eu/
ICT-56-2020 Next Generation Internet of Things	TERMINET, nexT gEneRation sMart INterconnectEd ioT	Type, reference: RIA, 957406. Description: Based on cutting-edge technologies such as software-defined networking (SDN), multiple-access edge computing, and virtualisation for next-generation IoT, the project will develop a novel next-generation reference architecture. Its main aim is to simplify the connection of a vast number of different devices through a flexible SDN-enabled middleware layer. To improve supply chain processes, the project will design an IoT-driven decentralised and distributed blockchain framework within manufacturing. TERMINET's approach will be tested in real-life situations such as energy, smart buildings, smart farming, healthcare, and manufacturing. Dates: 1/11/2020 – 31/11/2023. Cordis info: https://cordis.europa.eu/project/id/957406 Web: https://terminet-h2020.eu/
	iNGENIOUS, Next-GENeration IoT sOlutions for the Universal Supply chain	Type, reference: RIA, 957216. Description: Expanding into huge networks of heterogeneous organisations involved in the manufacture and delivery of products to end users, supply chains of the future are in for a big change, thanks to the many new technologies such as 5G, big data, blockchains, virtual reality and artificial intelligence. The project will design the next generation of internet of things (IoT) technology to add digital value to future supply chains. It will also propose technical and business enablers to build a complete platform for supply chain management. It will create a holistic security architecture for next-generation IoT built on neuromorphic sensors with security governed by Artificial Intelligence (AI) algorithms and tile-based hardware architectures based on security by design and isolation by default. In the application layer, new AI mechanisms will allow more precise predictions than conventional systems. Project outcomes will be validated into 4 large-scale Proof of Concept demonstration, covering 1 factory, 2 ports, and 1 ship, encompassing 6 uses cases. Dates: 1/10/2020 – 31/03/2023.

Topic	Acronym	Project Details
		Cordis info: https://cordis.europa.eu/project/id/957216 Web: https://ingenious-iot.eu/web/
	VEDLIoT, Very Efficient Deep Learning in IOT	Type, reference: RIA, 957197. Description: The project develops an IoT platform that uses deep learning algorithms distributed throughout the IoT continuum. The proposed new platform with innovative IoT architecture is expected to bring significant benefits to many applications, including industrial robots, self-driving cars, and smart homes. Dates: 1/11/2020 – 31/10/2023. Cordis info: https://cordis.europa.eu/project/id/957197 Web: https://vedliot.eu/
ICT-40 Cloud Computing: towards a smart cloud computing continuum	AI-SPRINT, Artificial Intelligence in Secure PRIvacy-preserving computing coNTinuum	Type, reference: RIA, 101016577. Description: AI-SPRINT defines a framework for developing AI applications in computing continua, enabling a finely-tuned trade-off between performance (e.g. in terms of end-to-end latency and throughput) and AI model accuracy, while providing security and privacy guarantees. Dates: 1/1/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/rcn/232728/en Web: https://www.ai-sprint-project.eu/
	CHARITY: Cloud for Holography and Cross Reality	Type, reference: RIA, 101016509. Description: Cloud for Holography and Cross Reality (CHARITY), which is a complete framework that attempts to overcome the challenges and meet the requirements of Advanced media applications. CHARITY leverages an innovative cloud architecture that exploits edge solutions, a computing and network continuum autonomous orchestration, application-driven interfacing, mechanisms for smart, adaptive and efficient resource management, strong community involvement, and overreaching compatibility with all infrastructure vendors Dates: 1/1/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/101016509 Web: https://www.charity-project.eu/en
	DataCloud ENABLING THE BIG DATA PIPELINE LIFECYCLE ON THE COMPUTING CONTINUUM	Type, reference: RIA, 101016509. Description: DataCloud provides a novel paradigm covering the complete lifecycle of managing Big Data pipelines through discovery, design, simulation, provisioning, deployment, and adaptation across the Computing Continuum. DataCloud delivers a toolbox of new languages, methods, infrastructures, and prototypes for discovering, simulating, deploying, and

Topic	Acronym	Project Details
		adapting Big Data pipelines on heterogeneous and untrusted resources. Dates: 1/1/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/101016835 Web: https://datacloudproject.eu/
	PHYSICS Optimized hybrid space-time service continuum in FAAS	Type, reference: RIA, 101017047. Description: PHYSICS applies a unified continuum approach, including functional and operational management across sites and service stacks, performance through the relativity of space (location of execution) and time (of execution), enhanced by semantics of application components and services. Dates: 1/1/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/101017047 Web: https://physics-faas.eu/the-project/
	SERRANO TRANSPARENT APPLICATION DEPLOYMENT IN A SECURE, ACCELERATED AND COGNITIVE CLOUD CONTINUUM	Type, reference: RIA, 101017168. Description: The EU-funded SERRANO project aims to introduce a novel ecosystem of cloud-based technologies, ranging from specialised hardware resources to software tool sets. The project will introduce an abstraction layer that transforms the distributed edge, cloud and high performance computing resources into a single borderless infrastructure, in addition to facilitating their automated and cognitive orchestration. Dates: 1/1/2021 – 31/12/2023. Cordis info: https://cordis.europa.eu/project/id/101017168 Web: https://ict-serrano.eu/