Last Updated on 22nd June 2022

Nata Stulova

contact

nata@stulova.me

links

web:// **stulova.me**
LinkedIn:// **nata-stulova**
GitHub:// **s0nata**

education

PhD in Software, Systems and Computing cum laude
2014–2018
Technical University of Madrid (UPM)

MSc in Artificial Intelligence
2012–2013
Technical University of Madrid (UPM)

BSc in Systems Analysis
2008–2012
National Technical University of Ukraine "Kyiv Polytechnic Institute" (NTUU "KPI")

developement

Java, C++, Prolog
♥ bash, git
GitLab, Phabricator
Trello, Notion
LaTeX
WordPress

languages

Ukrainian **native**

English, Spanish **proficient**

German **intermediate**

French, Hebrew **beginner**

summary

I am a research software engineer developing tools that keep software and its specifications aligned. I work with: requirement documents • source code comments • formal specifications.

experience

Senior Researcher | University of Bern | 2020-2021 remote / Bern, Switzerland

requirements and documentation engineering > In a team of four researchers I have managed the work on tool support for direct in-IDE integration of the source code and non-code software artifacts. Using Glamorous toolkit as a model development environment our team has:

- published a report *[10]* on design and implementation of functionality for low-code creation of Gherkin-style scenarios from the source code directly in an IDE, for use in BDD workflows;
- published a further generalization of this work *[11]* implementing IDE extensions to create and manage code-linked mind maps, Kanban boards, user stories, and interactive tutorial docs;

documentation analysis > As a team leader, project manager, and engineer working with several distributed R&D teams (4-6 people each) on source comments quality analysis, I have:

- established a collaboration between four research institutions to conduct a systematic analysis of the comment quality research trends within the last 10 years. My data filtering heuristic allowed to pre-select 2/3 of the relevant publications pool. Our analysis methodology identified 14 newly appeared quality attributes, and confirmed several persistent biases in academic studies *[9]*;
- contributed to an empirical study *[8]* on Java and Python developer adherence to coding style guidelines when writing comments, guiding data analysis and visualization;
- contributed to development of a comment clone detection tool that found 1300+ API documentation issues in 10 major Java libraries and systems *[7]*. My cross-validation of our experimental results using NiCad code clone analyzer showed reported methods match in <2% cases.

Scientist | Swiss Federal Institute of Technology in Lausanne (EPFL) | 2019-2020 Lausanne, Switzerland

code and documentation analysis> I have worked on natural language processing (NLP) use for augmenting software analyses, establishing and leading a collaboration between two research institutions on a project for static detection of code-comment inconsistencies during code change. Our prototype accurately pinpointed discrepancies in 9 out of 10 inconsistent co-changes *[6]*.

Research Assistant, Engineering | IMDEA Software Institute | 2014-2018 Madrid, Spain

static and dynamic code analysis> I have worked on program specification languages design, and on tools and techniques for specification-based source code analysis and verification. Joining a team working on Ciao, a dynamic Prolog-based language, its formal specification language of assertions, and its static and dynamic verification frameworks, I have:

- formalized and developed a specification language extension to enable dynamic analysis for higher-order calls *[1]*;
- formalized and developed several optimizations for source-to-source translation of formal specifications into runnable checks to minimize the run-time overhead introduced. One of my translation techniques combined with a mechanism of check result caching added to the dynamic verification framework led to 1-2 orders of magnitude check costs reduction *[2]*. I have also explored the possibilities of using static analysis information during the check code generation, achieving in some cases constant run-time overhead without correctness compromises *[3-4]*.
- collaborated on developing a static cost analysis technique to infer bounds on the overhead that run-time checking introduces in programs *[5]*

other qualifications

Lecturer | University of Bern | 2020-2021 remote/ Bern, Switzerland

teaching > at BSc and MSc levels, in person and fully remote:

- developed from zero a series of practical algorithms and data structures lectures within the Software Skills Lab course (lecture slides and videos, practical assignments, exams)
- co-supervised MSc and BSc theses
- gave lectures on programming languages, software verification, and UI design

Business analyst, Project manager, Web Developer | Ksi Prostir | 2020-2021 remote/ Dnipro, Ukraine

digital transformation > developing a website for a Dnipro-based cultural space KsiProstir. I have worked on the initial requirements analysis, after which I had collaborated in the no-code web development and maintenance.

service

conference organization > co-organized and ran CICLOPS'17, taking care of conference promotion, submissions review, guest speaker invitation, and overall web presence.

reviewing > for journals: EMSE, JOSS, Fundamenta Informaticae, and conferences: LOPSTR, ICLP.

publications

static and dynamic code analysis >

[1] Assertion-based Debugging of Higher-Order (C)LP Programs
N. Stulova, J. F. Morales, M. V. Hermenegildo [PPDP'14]

[2] Practical Run-time Checking via Unobtrusive Property Caching
N. Stulova, J. F. Morales, M. V. Hermenegildo [ICLP'15]

[3] Some Trade-offs in Reducing the Overhead of Assertion Run-time Checks via Static Analysis
N. Stulova, J. F. Morales, M. V. Hermenegildo [SCP volume 155]

[4] Exploiting Term Hiding to Reduce Run-time Checking Overhead
N. Stulova, J. F. Morales, M. V. Hermenegildo [PADL'18]

[5] Static Performance Guarantees for Programs with Run-time Checks
M. Klemen, N. Stulova, P. López-García, J. F. Morales, M. V. Hermenegildo [PPDP'18]

code and documentation analysis >

[6] Towards Detecting Inconsistent Comments in Java Source Code Automatically
N. Stulova, A. Blasi, A. Gorla, O. Nierstrasz [SCAM'20]

[7] RepliComment: Identifying Clones in Code Comments
A. Blasi, N. Stulova, A. Gorla, O. Nierstrasz [JSS volume 182]

documentation quality >

[8] Do Comments follow Commenting Conventions? A Case Study in Java and Python
P. Rani, S. Abukar, N. Stulova, A. Bergel, O. Nierstrasz [SCAM'21]

[9] A Decade of Code Comment Quality Assessment: A Systematic Literature Review
P. Rani, A. Blasi, N. Stulova, S. Panichella, A. Gorla, O. Nierstrasz [JSS, under review]

requirements and documentation engineering >

[10] Interactive Behavior-driven Development: a Low-code Perspective
N. Patkar, A. Chiş, N. Stulova, O. Nierstrasz [LowCode'21]

[11] First-class Artifacts as Building Blocks for Live in-IDE Documentation
N. Patkar, A. Chiş, N. Stulova, O. Nierstrasz [SANER'22]