March 22–26, 2021
Virtual, UK

Association for
Computing Machinery

Advancing Computing as a Science & Profession

‹Programming› 2021 Companion

Companion Proceedings of the 5th International Conference on

the Art, Science, and Engineering of Programming

Edited by:
Luke Church, Shigeru Chiba, and Elisa Gonzalez Boix

Sponsored by:
in-coop with ACM SIGPLAN, in-coop with ACM SIGSOFT

Supported by:
AOSA

Association for Computing Machinery, Inc.
1601 Broadway, 10th Floor
New York, NY 10019-7434
USA

ACM ISBN: 978-1-4503-8986-0

Additional copies may be ordered prepaid from:

	Phone: 1-800-342-6626
ACM	(USA and Canada)
General Post Office	+1-212-626-0500
P.O. Box 30777	(All other countries)
New York, NY 10087-0777	Fax: +1-212-944-1318
	E-mail: acmhelp@acm.org

Cover photo “Woods of Girton”, © 2020, by Luke Church, permission released to ACM, cropped version.

Production: Conference Publishing Consulting
D-94034 Passau, Germany, info@conference-publishing.com

Welcome from the Chairs

Welcome to the Companion Proceedings for the 5th International Conference on the Art, Science, and Engineering of Programming, named <Programming> 2021.

The International Conference on the Art, Science, and Engineering of Programming is a conference focused on programming topics including the experience of programming. We have named it ‹Programming› for short. ‹Programming› seeks for papers that advance knowledge of programming on any relevant topic, including programming practice and experience.

To build a community and to foster an environment where participants can exchange ideas and experiences related to practical software development, ‹Programming› hosts several co-hosted events, including workshops, posters and the student research competition. This companion gathers all the papers for these events.

This fifth edition was planned to be held in Cambridge, United Kingdom, March 22-26, 2021. Due to the COVID-19 outbreak, it was held online. All workshop events thus ran virtually including the the Salon Littéraire 2021, the International Workshop on Modern Language Runtimes, Ecosystems, and VMs (MoreVMs'21), the 5th edition of the International Workshop on Programming Technology for the Future Web (ProWeb21), the 7th edition of the Programming Experience Workshop (PX/21), the Compiler Coding Dojo (CoCoDo21), and the Open Applications Development (OAF21) tutorial.

We are grateful to AOSA for sponsoring <Programming> 2021, to ACM SIGPLAN and ACM SIGLOG for conferring the "in-cooperation-with" status, and to all from the <Programming> 2021 Organizing and Steering Committee members for the preparation of the event online.

We want to thank also the Program Committee members and the reviewers of all co-hosted events for their efforts in evaluating the submissions. We especially wish to thank the authors of submitted papers for their support in such a special edition affected by the COVID-19 outbreak.

Elisa Gonzalez Boix and Shigeru Chiba
Workshops Co-Chairs

Luke Church
General Chair

<Programming> 2021 Organization Committee

General Chair

Luke Church	University of Cambridge, UK

Program Chair

Jeremy Gibbons	University of Oxford, UK

Workshops Co-Chairs

Shigeru Chiba	The University of Tokyo, Japan
Elisa Gonzalez Boix	Vrije Universiteit Brussel, Belgium

Student Research Competition Co-Chairs

Philipp Haller	KTH Royal Institutue of Technology, Sweden
Hidehiko Masuhara	Tokyo Institute of Technology, Japan

Poster and Demonstrations Co-Chairs

Patrick Rein	Hasso Plattner Institute, Germany
Emma Söderberg	Lund University, Sweden

Student Volunteers Chair

Tom Beckmann	Hasso Plattner Institute, University of Potsdam, Germany

Virtualization Co-Chairs

Toni Mattis	Hasso Plattner Institute, University of Potsdam, Germany
Alan McCabe	Lund University, Sweeden
Fabio Niephaus	Hasso Plattner Institute, University of Potsdam, Germany
Patrick Rein	Hasso Plattner Institute, Germany

Publicity Chair

Fabio Niephaus	Hasso Plattner Institute, University of Potsdam, Germany

Web Chair

Tobias Pape	Hasso Plattner Institute, University of Potsdam, Germany

Steering Committee

Theo D'Hondt	Vrije Universiteit Brussel, Belgium (chair)
Ademar Aguiar	Universidade do Porto, Portugal
Davide Ancona	University of Genova, Italy
Jane Cleland-Huang	University of Notre Dame, USA
Krzysztof Czarnecki	University of Waterloo, Canada
Wolfgang De Meuter	Vrije Universiteit Brussel, Belgium
Matthew Flatt	University of Utah, USA
Lidia Fuentes	Universidad de Málaga, Spain
Richard P. Gabriel	Dream Songs, Inc. & HPI, California
Robert Hirschfeld	Hasso Plattner Institute Potsdam, Germany
Ranjit Jhala	University of California at San Diego, USA
Cristina Videira Lopes	University of California at Irvine, USA
Stefan Marr	University of Kent, UK
Patrick Rein	Hasso Plattner Institute, Germany
Guido Salvaneschi	Technische Universität Darmstadt, Germany
Manuel Serrano	INRIA, France
Emma Söderberg	Lund University, Sweden
Mario Südholt	IMT Atlantique, Nantes, France

Sponsors

AOSA

In-cooperation

acm SIGPLAN

5th International Workshop on Programming Technology for the Future Web (ProWeb21)

Full-fledged web applications have become ubiquitous on desktop and mobile devices alike. Whereas "responsive" web applications already offered a more desktop-like experience, there is an increasing demand for "rich" web applications (RIAs) that offer collaborative and even offline functionality—Google Docs being the prototypical example. Long gone are the days when web servers merely had to answer incoming HTTP requests with a block of static HTML. Today's servers react to a continuous stream of events coming from JavaScript applications that have been pushed to clients. As a result, application logic and data is increasingly distributed. Traditional dichotomies such as "client vs. server" and "offline vs. online" are fading.

The ProWeb workshop series is a forum for researchers and practitioners to share and discuss new technology for programming these and future evolutions of the web. It was our pleasure to host the 5th edition, ProWeb21, alongside <Programming> 2021. Due to the continued disruption due to the COVID-19 pandemic, the workshop was held online.

ProWeb21 received 5 submissions, and the submissions went through a rigorous reviewing process. Every submission received three reviews by the PC members, and was carefully discussed until a consensus was reached. All decisions were based solely on the quality of the submissions and on the outcome of the discussion; we did not target any minimum or maximum number of papers to be accepted. The program committee accepted two full papers to be included in these proceedings, and one presentation abstract available on the website of the workshop.

We were immensely fortunate to host Guido Salvaneschi of the University of St. Gallen as the workshop's keynote speaker, who gave an excellent talk entitled "Why Programming Languages for Distributed Systems are Inevitable". Furthermore, since the 2020 edition was called off due to the start of the pandemic, we were also fortunate to host two ProWeb20 talks along with the workshop.

In spite of the lack of a physical meeting, we greatly enjoyed the high-quality talks, audience participation, and lively discussions. We would like to thank all authors for submitting a set of high-quality submissions, and the program committee for their careful reviews.

Simon Fowler and Andrea Stocco
PC Co-Chairs

Program Committee

- Simon Fowler (University of Glasgow)
- Andrea Gallidabino (Università della Svizzera italiana)
- Daniel Hillerström (University of Edinburgh)
- Magnus Madsen (Aarhus University)
- Jens Nicolay (Vrije Universiteit Brussel)
- Gabriela Sampaio (Imperial College London)
- Andrea Stocco (Università della Svizzera italiana)

7th International Workshop on Programming Experience (PX/21)

Message From the Chairs

Some programming feels fun, other programming feels annoying. Why?

For a while now the study of programming has forced improvements to be described through the Fordist lens of usability and productivity, where the thing that matters is how much software can get built, how quickly.

But along the way, something has gone missing. What makes programmers feel the way they do when they're programming? It's not usually fun to spend an age doing something that could have been done easily, so efficiency and usability still matter, but they're not the end of the story.

Some environments, activities, contexts, languages, infrastructures make programming feel alive, others feel like working in a bureaucracy. This is not purely technologically determined, writing Lisp to do your taxes probably still isn't fun, but it's also not technologically neutral, writing XML to produce performance art is still likely to be <bureaucratic></bureaucratic>.

Whilst we can probably mostly agree about what isn't fun, what is remains more personal and without a space within the academy to describe it.

In its past editions, PX set its focus on questions like: Do programmers create *text* that is transformed into running behavior (the old way), or do they operate on *behavior* directly (*"liveness"*); are they exploring the *live domain* to understand the true nature of the requirements; are they *like authors creating new worlds*; does *visualization* matter; is the experience *immediate, immersive, vivid and continuous*; do *fluency, literacy, and learning* matter; do they build *tools, meta-tools*; are they creating *languages* to express new concepts quickly and easily; and curiously, is *joy* relevant to the experience?

In this PX, we expand its focus to also cover the *experience that programmers have.* What makes it and what breaks it? For whom? What can we build to share the joy of programming with others?

PX/21 was the seventh edition of PX and the second online-version of the workshop: Participants met virtually, authors presented their work in sessions following the Writers' Workshop structure, and everyone engaged in lively discussions that extended beyond the end of the scheduled time.

Our post-workshop proceedings allowed authors to reflect on the feedback they got from both the program committee and the workshop participants and improve their submission.

We would like to thank our program committee, all workshop attendees, and most importantly our authors for their contributions, constructive criticism, hard work, and willingness to share their ideas.

—Luke Church, Richard P. Gabriel, Hidehiko Masuhara, and Robert Hirschfeld

Papers

Web

http://programming-experience.org/px21/
https://2021.programming-conference.org/home/px-2021/

Program Committee

Organizers

4th Raincode Labs Compiler Coding Dojo (CoCoDo 2021)

Welcome from the Chairs

CoCoDo — the Raincode Labs Compiler Coding Dojo — was founded in 2017, not as a workshop or a mini-conference, but intentionally as a *coding dojo*. The intent is that participants enjoy an entire day of hands-on programming, instead of preparing a paper beforehand and giving a presentation about it on the day of the event. The topic of CoCoDo is compiler construction: arguably the oldest branch of computer science, which has been accumulating useful techniques since at least the 1950s. Compiler construction comprises, but is not limited to, lexical analysis, syntactic analysis, preprocessing, context handling, code generation, code optimisation, virtual machines, interpreters, smell detection, clone management, portability, migration, refactoring, domain-specific language design, linking and loading, assembling and disassembling, generics and reflection, and so much more.

The idea of CoCoDo came into existence within Raincode Labs, the largest independent compiler company in the world, freely (but not always for free) providing compiler services to anyone interested. Raincode analysts and developers know better than anyone that compiler construction in practice is far from the boring by-the-book activity it might seem to the uninitiated. This is because it covers a wide range of software language engineering activities, from language design [17] to grammar inference [18], from quality assurance [16, 19] to bad smell elimination [9, 15], from pattern mining [4, 13] to log diffing [3], from model-based code generation [1, 21] to new parsing techniques [18, 20], from live programming [5] to abstract interpretation [6]. The main objective of CoCoDo was to expose participants of ‹Programming›, practitioners and researchers alike, to such realities and practices of compiler construction. Its intended audience was mixed, only supposing some passion for programming languages.

Over the years we have observed having three kinds of participants:

1. compiler experts, carefully handpicked and invited by the organisers; some were serving as overseers of the dojo while others led by example;
2. metagrammarware developers, who were bringing their own language workbenches, languages, tools and techniques to CoCoDo to demonstrate their capabilities and to get feedback;
3. software engineers ranging from students to long time hobbyists, all sharing a casual interest in compiler techniques.

In particular, the first instance, CoCoDo 2017, had the following sessions:

- *Attribute Grammars for DSLs for Music and 3D Graphics*, overseen by **Elizabeth Scott** and **Adrian Johnstone** with ART [7];
- *Metaprogramming in Late Phases of Compilation*, overseen by **Anya Helene Bagge** with Rascal[1];
- *Experimenting with Racket as a Language Workbench*, overseen by **Robby Findler** with Racket[2];
- *The Future of Compilers*: a session consisting of a number of short presentations and pitches, including **Rik Arends** pitching his MakePad[3], **Ralf Lämmel** introducing his upcoming *Software Languages* book [11], **Adrian Johnstone** again with his work on FunCons [8], and **Nicolas Laurent** with his compiler framework Whimsy[4] and parsing framework Autumn[5] [12].

In 2018 we have gone for the similar experience with a different lineup of sessions:

- **Peter D. Mosses** presented his well-known work on FunCons[6] — in fact, this was the first historical public reveal of the newly redesigned CBS[7], a framework and meta-language for component-based specification of programming languages;

[1]Rascal metaprogramming language: `https://www.rascal-mpl.org`
[2]Racket (a Scheme dialect): `https://racket-lang.org`
[3]Makepad: `https://makepad.nl`
[4]Whimsy compiler framework: `https://github.com/ncellar/whimsy`
[5]Autumn parser combinator library: `https://github.com/norswap/autumn`
[6]PLanCompS: `https://plancomps.csle.cs.rhul.ac.uk`
[7]CBS framework: `https://plancomps.github.io/CBS-beta/`

- **Jesper Öqvist** demonstrated and taught JastAdd[8], a metacompilation system for attribute grammars;
- **Anya Helene Bagge** returned with Rascal, this time walking the audience through the language itself instead of showing libraries written in it;
- **Johan Fabry** gave a pitch on a live programming language for robotics [2], **Friedrich Steimann** on dependency grammars [14] and **Vadim Zaytsev** on compiler testing [19].

In 2019, we had the following:

- *Scrap your DSL Boilerplate with a Universe of Syntaxes, their Programs and Proofs*, a highly interactive session where **Guillaume Allais** demonstrated how one can use Agda[9] to prove correctness of program properties;
- *Language Engineering with Rascal*, where **Tijs van der Storm**, one of the core designers of the language, gave a full-fledged introductory course into it;
- A very special *Tool Battle* sessions where two similar tools were presented by members of their development team: SmaCC[10] by **Jason Lecerf** and PetitParser [10] by **Andrei Chiş**.

The battle of the parsers was a close one, where both parties recognised each others respective strengths and weaknesses, so it was called a draw.

After the unfortunate decision to cancel ("postpone") ‹Programming› 2020, we also had to comply and cancel the already fully prepared dojo. We are still grateful to **Walter Cazzola**, **Jeff Smits** and **Dimitri Racordon** for the effort they have put into the preparation, as well as for their understanding when we were hit by the first conference cancellation in what turned out to be a persistent series of event cancellations and virtualisations.

[8]JastAdd: `https://jastadd.cs.lth.se/`

[9]Agda: `https://agda.readthedocs.io/en/v2.6.0.1/`

[10]SmaCC compiler compiler: `https://books.pharo.org/booklet-Smacc/html/`

In 2021, we have gone for the online setup right away, even though it seemed like a poor fit for such a highly interactive event as CoCoDo. Following the online conference format, we could fit three sessions:

- *A Tutorial on the Spoofax Language Workbench* by **Eelco Visser**, the main designer of both the workbench and many of the (meta)languages it comprises;
- *An Interactive Exploration of a Simple Compiler* by **Marcus Denker**, amazing us with the internals of Pharo;
- *From Abstract Syntax Trees to Machine Code with LLVM* with **Dimitri Racordon** providing a comprehensive primer introduction into LLVM[11] and Swift[12] with a specially designed Cocodol[13] language as a running example.

After the event, we also collected submissions from all interested tutorialists of 2021 and provided them with detailed reviews by the members of our programme committee (see next page), resulting in one final tutorial paper suitable for these post-proceedings. We hope this serves as a good self-archiving point, and hereby invite all the readers to participate in CoCoDo 2022 and beyond!

May 2021

Johan Fabry, johan@raincode.com
Vadim Zaytsev, vadim@grammarware.net

https://cocodo.github.io

[11]LLVM compiler infrastructure: https://llvm.org
[12]Swift programming language https://developer.apple.com/swift/
[13]Cocodol: https://github.com/kyouko-taiga/Cocodol

References

[1] V. Blagodarov, Y. Jaradin, and V. Zaytsev. Tool Demo: Raincode Assembler Compiler. In T. van der Storm, E. Balland, and D. Varró, editors, *Proceedings of the Ninth International Conference on Software Language Engineering (SLE)*, pages 221–225, 2016. doi: `10.1145/2997364.2997387`.

[2] M. Campusano and J. Fabry. Live Robot Programming: The Language, Its Implementation, and Robot API Independence. *Science of Computer Programming*, 133:1–19, 2017. doi: `10.1016/j.scico.2016.06.002`.

[3] C. Deknop, J. Fabry, K. Mens, and V. Zaytsev. Improving Software Modernisation Process by Differencing Migration Logs. In M. Morisio, M. Torchiano, and A. Jedlitschka, editors, *Proceedings of the 21st International Conference on Product-Focused Software Process Improvement (PROFES)*, pages 270–286. Springer, 2020. doi: `10.1007/978-3-030-64148-1_17`.

[4] D. Di Nucci, H. S. Pham, J. Fabry, C. De Roover, K. Mens, T. Molderez, S. Nijssen, and V. Zaytsev. A Language-Parametric Modular Framework for Mining Idiomatic Code Patterns. In A. Etien, editor, *Post-proceedings of the 12th Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE)*, volume 2510 of *CEUR Workshop Proceedings*, pages 38–44. CEUR-WS.org, 2019. `http://ceur-ws.org/Vol-2510/sattose2019_paper_3.pdf`.

[5] J. Fabry. The Meager Validation of Live Programming. In *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*, ‹Programming›'19. ACM, 2019. doi: `10.1145/3328433.3328457`.

[6] J. Fabry, Y. Jaradin, and A. Gül. Engineering a Converter Between Two Domain-Specific Languages for Sorting. In *Proceedings of the 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 221–226, 2020. doi: `10.1109/SCAM51674.2020.00030`.

[7] A. Johnstone and E. Scott. Translator Generation Using ART. In B. A. Malloy, S. Staab, and M. van den Brand, editors, *Revised Selected Papers of the Third International Conference on Software Language Engineering*, volume 6563 of *LNCS*, pages 306–315. Springer, 2010. doi: `10.1007/978-3-642-19440-5_20`.

[8] A. Johnstone and E. Scott. Principled and Pragmatic Specification of Programming Languages. In B. Dongol, L. Petre, and G. Smith, editors, *Formal*

Methods Teaching Workshop (FMTea), pages 165–180. Springer, 2019. doi: 10.1007/978-3-030-32441-4_11.

[9] J. Kennedy van Dam and V. Zaytsev. Software Language Identification with Natural Language Classifiers. In K. Inoue, Y. Kamei, M. Lanza, and N. Yoshida, editors, *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering: the Early Research Achievements track (SANER ERA)*, pages 624–628. IEEE, 2016. doi: 10.1109/SANER.2016.92.

[10] J. Kurš, G. Larcheveque, L. Renggli, A. Bergel, D. Cassou, S. Ducasse, and J. Laval. *Deep into Pharo*, chapter 18 PetitParser: Building Modular Parsers, pages 375–410. Square Bracket Associates, 2013. doi: 10.7892/boris.47152.

[11] R. Lämmel. *Software languages: Syntax, Semantics and Metaprogramming*. Springer, 2018. Book's website: http://www.softlang.org/book, doi: 10.1007/978-3-319-90800-7.

[12] N. Laurent and K. Mens. Taming Context-Sensitive Languages with Principled Stateful Parsing. In *Proceedings of the Ninth International Conference on Software Language Engineering*, SLE, page 15–27. ACM, 2016. doi: 10.1145/2997364.2997370.

[13] H. S. Pham, S. Nijssen, K. Mens, D. Di Nucci, T. Molderez, C. De Roover, J. Fabry, and V. Zaytsev. Mining Patterns in Source Code using Tree Mining Algorithms. In P. K. Novak, T. Šmuc, and S. Džeroski, editors, *Proceedings of the 22nd International Conference on Discovery Science (DS)*. Springer, 2019. doi: 10.1007/978-3-030-33778-0_35.

[14] F. Steimann. Replacing Phrase Structure Grammar with Dependency Grammar in the Design and Implementation of Programming Languages. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward!, page 30–43. ACM, 2017. doi: 10.1145/3133850.3133859.

[15] M. Stijlaart and V. Zaytsev. Towards a Taxonomy of Grammar Smells. In B. Combemale, M. Mernik, and B. Rumpe, editors, *Proceedings of the 10th International Conference on Software Language Engineering (SLE)*, pages 43–54. ACM, 2017. doi: 10.1145/3136014.3136035.

[16] L. Włodarski, B. Pereira, I. Povazan, J. Fabry, and V. Zaytsev. Quality First! A Large Scale Modernisation Report. In X. Wang, Z. Chen, and J. Hu, editors, *Proceedings of the 26th IEEE International Conference on Software Analysis,*

Evolution and Reengineering — Industry Track (SANER IT), pages 569–573, 2019. doi: `10.1109/SANER.2019.8668006`.

[17] V. Zaytsev. Language Design with Intent. In D. Batory, J. Gray, and V. Kulkarni, editors, *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, pages 45–52. IEEE, 2017. doi: `10.1109/MODELS.2017.16`.

[18] V. Zaytsev. Parser Generation by Example for Legacy Pattern Languages. In M. Flatt and S. Erdweg, editors, *Proceedings of the 16th International Conference on Generative Programming: Concepts and Experience (GPCE)*, pages 212–218. ACM, 2017. doi: `10.1145/3136040.3136058`.

[19] V. Zaytsev. An Industrial Case Study in Compiler Testing. In D. J. Pearce, T. Mayerhofer, and F. Steimann, editors, *Proceedings of the 11th International Conference on Software Language Engineering (SLE)*, pages 97–102. ACM, 2018. doi: `10.1145/3276604.3276619`.

[20] V. Zaytsev. Event-Based Parsing. In T. Kamina and H. Masuhara, editors, *Proceedings of the Sixth Workshop on Reactive and Event-based Languages and Systems (REBLS)*, 2019. doi: `10.1145/3358503.3361275`.

[21] V. Zaytsev. Modelling of Language Syntax and Semantics: The Case of the Assembler Compiler. *Journal of Object Technology (ECMFA@JOT)*, 19, July 2020. doi: `10.5381/jot.2020.19.2.a5`.

Programme Committee

To process papers for the post-proceedings, we are grateful to have relied on the following programme committee members:

- **Andrei Chiş** (feenk, Switzerland)
- **Johan Fabry** (Raincode Labs, Belgium)
- **Adrian Johnstone** (Royal Holloway, UK)
- **Stefan Marr** (University of Kent, UK)
- **Fabio Niephaus** (Hasso Platner Institut, Germany)
- **Elizabeth Scott** (Royal Holloway, UK)
- **Anthony Sloane** (Australia)
- **Tijs van der Storm** (CWI & University of Groningen, The Netherlands)
- **Vadim Zaytsev** (University of Twente, The Netherlands)

Contents