Eating Your Own Dog Food: WebDSL Case Studies to Improve Academic Workflows

Danny M. Groenewegen
Delft University of Technology, The Netherlands

Elmer van Chastelet
Delft University of Technology, The Netherlands

Max M. de Krieger
Delft University of Technology, The Netherlands

Daniel A. A. Pelsmaeker
Delft University of Technology, The Netherlands

Abstract

SDF, Stratego and Spoofax provide a platform for development of domain-specific programming languages. On this platform, the WebDSL project started out as a case study in language engineering, and grew into a reliable tool for rapid prototyping and continuous development of web applications. Our team led by Eelco Visser develops and operates several web applications to support academic workflows. EvaTool governs the process of course quality control, importing questionnaire data, and providing lecturers and education directors with a platform to discuss and agree on improvements. WebLab is an online learning management system with a focus on programming education, with support for lab work and digital exams, used by over 40 courses. Conf Researchr is a domain-specific content management system for creating and hosting integrated websites for conferences with multiple co-located events, used by all ACM SIGPLAN and SIGSOFT conferences. MyStudyPlanning is an application for composition of individual study programs by students and verification of those programs by the exam board, used by multiple faculties at the Delft University of Technology. These tools served as practical case studies for applying the research, and ensure the continued development of the underlying platform.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases WebDSL, WebLab, Spoofax, Eelco Visser, education

Digital Object Identifier 10.4230/OASIcs.EVCS.2023.13

Acknowledgements We want to thank Eelco Visser for his guidance and vision.

1 Introduction

Software that is produced as part of academic research often remains in a prototype state. There is tension between investing time in improving the quality of the software and reporting on the software through publications and presentations. By applying the software to practical real-world case studies, both the quality of the software improves by discovering and solving issues, and the reporting becomes stronger through meaningful evaluation. Applying research software in practice also generates new questions to explore. Eelco Visser's vision for language design and engineering involves a recurring pattern of building upon software contributions from previous work, and applying the research ideas to new practical case studies. This research expanded from the fields of language engineering into build and deployment tools and web application engineering. Eelco brought together the people that worked on the projects, primarily as part of Master and PhD thesis projects. Figure 1 shows a timeline of notable projects by Eelco and his students.

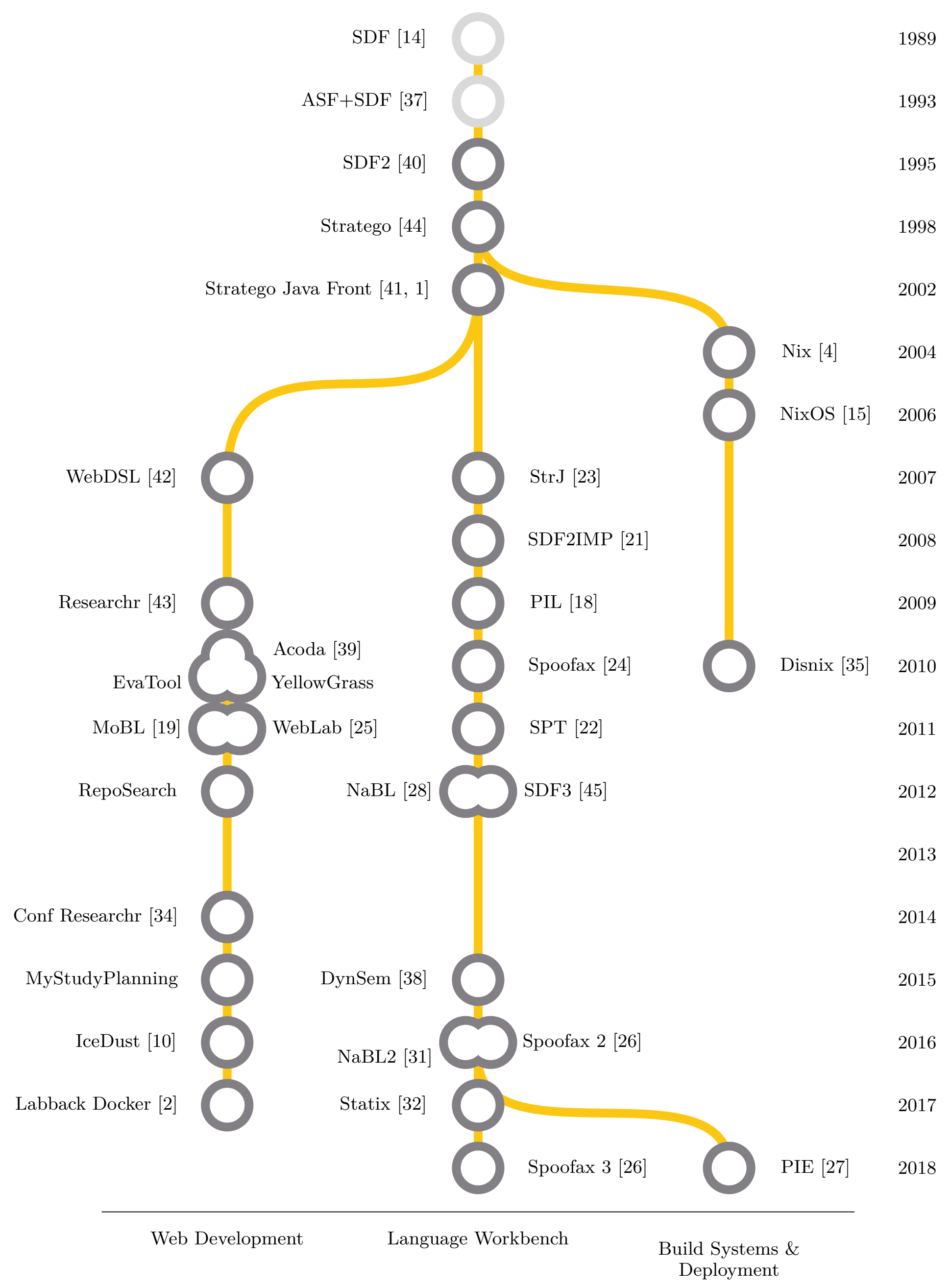

Figure 1 A timeline of notable projects by Eelco Visser and his students. Eelco improved the Syntax Definition Formalism [14] (SDF) as part of his PhD on SDF2 [40]. The Stratego language [44] was designed as an improvement on ASF+SDF [37], allowing programmers complete control of traversal strategies for term rewriting. Bootstrapping the Stratego compiler initiated research into better build systems with Nix [4], which was incorporated as the package manager for the Linux distribution NixOS [15]. WebDSL [42] grew from the wish to improve processes in academia, and served as the largest case study in research that led to the full-fledged language workbench Spoofax [24] to generate IDEs. At the time of writing, several real-world web applications with thousands of users, built using WebDSL and Spoofax, are running on NixOS servers.

Eelco had a strong wish for software to automate the mundane and repetitive parts of working in academia, and therefore Eelco advocated to form a dedicated development team within the department, tasked with writing and maintaining such tools for managing the academic workflow. This was a tough sell, since the university management wanted to avoid depending on in-house developed tools, rather outsourcing both the effort and the responsibility to third-party software companies. When a top-down approach did not pan out, Eelco decided to improve the academic workflow from the bottom-up, starting with automations that would help him in his own courses and his dealings with the students he was supervising, and one approach was to create web applications. Eelco's initial prototype, used to gather requirements for the design of WebDSL, was a web application for tracking master student projects and progress, to replace a previous generic wiki application that lacked any domain knowledge. However, it is difficult to find the resources for maintaining the WebDSL language and applications merely through PhD projects. As a result, the starting point for more widely used applications came when Eelco formed the Academic Workflow Engineering team around 2013, initially consisting of Danny Groenewegen and Elmer van Chastelet, and later joined by Daniel Pelsmaeker and Max de Krieger. We became a permanent team of software developers to continue WebDSL development, and develop new web applications based on real case studies with clients. Initially, most web applications were used just within the Programming Languages group that Eelco led, but thanks to Eelco promoting the applications, they gradually started being adopted in other groups within the Computer Science faculty, then outside the faculty and even outside the university.

Section 2 highlights the main benefits of WebDSL for creating web information systems, and explains how it was built upon previous research by Eelco Visser. The Researchr application (Section 3), an early WebDSL application, assists paper writing by indexing computer science papers, managing bibliographies, and generating required BibTeX files. EvaTool (Section 4) governs the process of course quality control, importing questionnaire data, and providing lecturers and education directors with a platform to discuss and agree on improvements. The WebLab online learning management system (Section 5) has a focus on programming education (students make programming assignments in the browser), with support for lab work and digital exams. WebLab is currently used by over 40 courses, some even outside Delft University of Technology. Conf Researchr (Section 6) is a domain-specific content management system for creating and hosting integrated websites for conferences with multiple co-located events, used by all ACM SIGPLAN and SIGSOFT conferences. Conf Researchr has been used to serve thousands of attendees at over a hundred editions of dozens of (co-located) conferences and workshops. MyStudyPlanning (Section 7) is an application for composition of individual study plans by students and verification of those plans by the exam board, used by multiple faculties at Delft University of Technology. Finally, we wrap up with the conclusion in Section 8.

2 WebDSL

Web programming suffers from abstraction issues, as web frameworks are developed on top of general-purpose programming languages that were not specifically designed for web programming. As a result, static verification of web programming concepts is limited by the strength of the type checker of the programming language, sometimes augmented with separate linting tools built into IDEs [16]. Security should also be taken into account in the design of web programming languages, otherwise regular programming language features become pitfalls that result in web applications with vulnerabilities. For example, using a simple string interpolation feature for concatenation of SQL queries easily leads to injection vulnerabilities.

The WebDSL project [42, 5] takes a language engineering approach to improving web programming. WebDSL[1] is a linguistically integrated domain-specific language (DSL) for web programming that combines abstractions for web programming concerns covering transparent persistence, user interfaces, data validation [7], access control [6], and internal site search [33]. Sublanguages for the various concerns are integrated into WebDSL through static verification to prevent inconsistencies, with immediate feedback in the IDE and error messages in terms of domain concepts. The user interface sublanguage of WebDSL integrates automatic data binding, provides safety from data tampering, prevents input identifier mismatch in action handlers, enables safe composition of input components, enforces cross-site request forgery protection, allows expressive data validation, and supports partial page updates without explicit JavaScript or DOM manipulation. The access control sublanguage allows various policies to be expressed with simple constraints. Enforcement of access control is done by automatically weaving checks into the application code. The explicit declaration of access control though language elements allow assumptions to be made about the related parts in the application, such as hiding inaccessible navigation links. WebDSL is designed and most suitable for building full-stack web information systems, applications with a rich data model, data entry with validation rules, and a multi-user access control policy.

WebDSL development was started by Eelco Visser as a case study for using SDF and Stratego to build real-world programming language compilers [42]. His earlier research on language embedding, together with Martin Bravenboer, resulted in libraries for embedding Java syntax in Stratego code [1], which provided convenient programming tools for code generation. Eelco, together with PhD students Zef Hemel, Lennart Kats, and Danny Groenewegen, used the WebDSL project to discover how to structure compilers and which programming patterns to use in a system that provides rewriting rules with concrete syntax embedding [17]. In parallel, Lennart Kats worked on Spoofax, which consists of Eclipse IDE support for language definitions written in SDF and Stratego [24], and the Stratego Java compiler backend [23] which expanded on initial work on a Stratego Java interpreter by Eelco's former PhD student Karl Trygve Kalleberg [20]. Eelco observed that name binding rules generated complexity in the rewriting rules of the WebDSL compiler implementation, which inspired research into better abstractions for specifying name binding and static semantics, in particular NaBL [29], NaBL2 [31], and ultimately Statix [32]. WebDSL is still the largest case study for these tools, as shown in the encoding of WebDSL static semantics in SDF3 and Statix by Max de Krieger [3].

Another development aspect of compilers is build systems. The bootstrapping required to build Stratego required better tool support, which was initial inspiration for the Nix project by PhD student Eelco Dolstra [4]. This project grew into the full Linux distribution NixOS[2] based on the Nix principles for reproducible, declarative and reliable system specification. WebDSL application deployment is still done on NixOS servers. One specification describes a complete system configuration, which, for example, contains multiple MySQL and Tomcat installations and an Nginx reverse proxy server.

[1] `https://webdsl.org/`
[2] `https://nixos.org/`

3 Researchr

Researchr[3] [43] was one of the first case studies with WebDSL. It is a web application that can search and index computer science publications, and perform bibliography management. At the time, this project was motivated by Eelco's frustration with the DBLP Computer Science Bibliography, a large computer science bibliography website, that would not list papers that were not in the curated list of conferences. Eelco was also convinced that much of the time and effort required to search and use literature was a result of poor tool support and the fact that different independent tools were used for indexing, managing, and classification of publications. Therefore, Eelco wanted Researchr to be the one-stop-shop for finding and categorizing publications, literature review, and managing bibliographies. This requires a big catalog of publications, so Researchr automatically imports all the publications indexed by DBLP, but also allows users to contribute and improve publication metadata.

Although the initial prototype of WebDSL was lacking features for user management, at this time we had developed essential language additions for making complete applications through the exploration of access control [6] and data validation [8]. Researchr was used as a test bed for new developments in the WebDSL compiler, in particular for the renewed implementation of the WebDSL runtime [5]. As the Researchr project grew, so did the demands on WebDSL. For example, the large set of publications in Researchr required an effective and efficient search feature. Implementing such a system with searching entities in the database became too slow and this inspired the development of a dedicated WebDSL language feature for search using Lucene [33]. The project was also a great way to find bugs, memory leaks, and performance problems in WebDSL, and as WebDSL became more stable Eelco worked on Researchr more as a continuous side project, occupying his mind even on holidays.

4 EvaTool

For the performance of courses and evaluating our education at the Delft University of Technology, various statistics are gathered about the courses, as well as feedback from students. These statistics are collected in various different systems across the university, such as EvaSys for surveys, Osiris for grades, and summaries from student feedback groups. Additionally, there is a need to record the instructor response to the course performance, and to record future plans for the course. All this information used to be sent around using ad-hoc emails and was monitored manually in spreadsheets by the quality assurance department. As a result, EvaTool[4] was developed to collect and analyze this information in a single place, and gain better insight into the educational system. The project was initiated by Eelco in 2010 after he had several discussions with colleagues from the quality assurance department. A student who took Eelco's course on Model-Driven Software Development was hired to work on the first prototype of EvaTool. A prototype was created with a basic data model supporting the different steps in the course evaluation process. Various iterations of this prototype were used to tweak functionality and to search for the right workflow and terminology to be valuable in a broader setting with multiple faculties.

EvaTool transformed the ad-hoc nature of course quality control into a directed process of education evaluation, covering the collection and filtering of data, informing relevant people, asking for response and intended improvements, and projecting the evaluation data in the

[3] `https://researchr.org/`
[4] `https://evatool.tudelft.nl/`

various formats suitable for management or students. Over the years, EvaTool gained traction, resulting in other faculties now using the tool to manage their education evaluation process. As these processes differ slightly between faculties, EvaTool was extended with faculty-specific customizations, including customizable roles and optional steps in the evaluation process. The tabular and textual reporting capabilities were extended with a template designer, enabling more visual fact sheets to be generated from the course evaluations.

5 WebLab

WebLab[5] is an online learning environment, which is currently used by courses both inside and outside Delft University of Technology for practice materials, lab assignments, programming exercises, and exams. It started around 2011, while Eelco was teaching Model-Driven Software Development and concluded that it would be much easier to have the students fill out multiple-choice and open questions online than to try to read the various answers on paper. Additionally, if the students had to submit code, Eelco argued that there should be a platform to which they can submit instead of students emailing the course instructor with a ZIP archive of the code. Finally, once the questions and submissions are in an online system, it could also be used to hold the grades for the individual submissions and automatically calculate the overall grade for the student. This inspired the creation of WebLab, which thanks to previous positive experiences was also written in WebDSL.

While Eelco was teaching Concepts of Programming Languages, it became apparent that providing an online editor for student code submissions would not only solve the problem of students having to install development software locally, but also provide an opportunity to perform automatic grading of the code submissions. The initial version started with an email to Danny in December 2011, where Eelco asked how to integrate the Cloud9's Ace editor component in WebDSL code. This simple code editor provided syntax highlighting for the languages in which the course was taught, such as Java and Scala [36]. For programming assignments and automated grading, Vlad Vergu implemented a WebLab *backend* that runs the student's code on the Java Virtual Machine (JVM) in a severely restricted environment. By running a hidden suite of specification tests against the student's code, the test results could be used to determine the grade: the more tests succeed, the better the student is doing. JavaScript support was later added through the JVM's built-in Javascript functionality known as Nashorn, and eventually WebLab also supported C code, which was run using Emscripten and JavaScript on the JVM.

Another course that gave direction to the development of WebLab was Algorithms and Data Structures, which was in essence a stress test for WebLab: previous editions of this course had over 300 students making programming assignments, sometimes all at the same time during an exam (current edition has 600 students). Therefore, it was imperative that WebLab would scale sufficiently to be able to handle all these requests in a timely manner. From this new requirement, the idea of having multiple backends emerged. Each backend would handle and run part of the submissions of the students, and if a backend ended up crashing (for example, due to a student error), a fresh new backend could take its place without impacting the availability of WebLab or the other backends.

As WebLab was growing as a platform and as a case study for WebDSL, it was also useful as a topic for various master and PhD projects. This allowed these projects to extend and improve WebLab and WebDSL, and see their results used in a large application that is actually used in practice. There were bachelor projects that added features such as user groups and SQL support [30].

[5] `https://weblab.tudelft.nl/`

One notable project that generated multiple papers was IceDust [10, 11, 12, 13], part of the PhD work of Daco Harkes [9]. This project resulted directly from the need to streamline the grade calculation and grade dependencies in WebLab, as grade calculations are modelled as *derived values* with complex interactions and dependencies between values (grades) in the data model. In this research he explored the problems associated with derived values and their transitive dependencies, and the performance characteristics of three different strategies for calculating them (on read, on write, or *eventually consistent*).

Eventually, WebLab was used by so many courses that the original JVM-based backends became too restrictive, and were therefore replaced by backends that run isolated Docker containers, as part of another research project [2]. This allows programming assignments for any language for which a Docker image can be created, and it is currently used to provide support for Python, Haskell, Rust, and Agda, among others.

WebLab has been in development over the past decade and gradually gained more and more features, including file submissions, collaborative assignments, and question variants. New features are still being added, such as support for *language servers* using the Language Server Protocol[6] (LSP), learning paths, parameterized questions, and personalized feedback to improve formative testing.

6 Conf Researchr

As an academic, Eelco attended many conferences but noticed that these conferences often had to build a new website for every edition of a conference. He saw an opportunity to help the conference organizers by building a domain-specific content management system (CMS) that would support the needs of conference websites. The conference and many different sub-conferences, workshops, symposia, talks, and other events are coordinated and managed by various organizers and committees, yet should ultimately produce a single conference website that attendees can quickly navigate. With help from Jan Vitek, Eelco secured a contract for funding the development of this system, known as Conf Researchr[7] [34], and the first conference to adopt the system was SPLASH 2014, a large conference with co-hosted subconferences and workshops. Initially, Conf Researchr was meant to be part of Researchr, however, due to time constraints Conf Researchr remained a separate application sharing only the name and the WebDSL implementation language.

Over time, Conf Researchr gained many features for conference organizers, such as support for plenary and blended sessions and timezones. During the COVID-19 pandemic, support was added for virtual conferences with mirrored sessions, such that virtual attendees in other timezones can still catch a talk at an acceptable time. Additionally, conference attendees can create personalized calendars of the talks and events they plan to visit, and export or synchronize this to their personal calendar. As the number of conferences grew, so did the requirements. Integrations with other applications such as HotCRP[8], Conference Publishing Consultancy[9], and EasyChair[10] were added, as well as a simple CMS-mode for workshops and single-track conference websites that do not need all the advanced features. Ultimately, Conf Researchr has supported all SIGPLAN conferences since 2014, ICSE since 2018, and all SIGSOFT conferences since 2020.

[6] `https://microsoft.github.io/language-server-protocol/`
[7] `https://conf.researchr.org/`
[8] `https://hotcrp.com/`
[9] `https://www.conference-publishing.com/`
[10] `https://easychair.org/`

7 MyStudyPlanning

At Delft University of Technology, first year Master students are required to plan ahead and select a number of courses to follow as part of their Individual Exam Program (IEP) in order to obtain the required amount of course credits (ECTS). Previously, students filled out a paper form containing this information, which was sent around multiple departments to gather approvals and signatures, and if the student's program changed, the forms had to be adjusted and sent around again. This process could be more streamlined and less error-prone if it were automated, which resulted in MyStudyPlanning.

MyStudyPlanning[11] is a web application, written in WebDSL, that guides the student through the course selection procedure, and even allows early feedback through the use of categories with constraints, automatically selecting mandatory courses, and showing warnings when the student selects invalid combinations. Additionally, both employees registering the program and employees mandated to approve the submitted course selections benefit from MyStudyPlanning by having clear overviews of pending requests, reminders and notifications, and a traceable history of a course selection. Thanks to WebDSL's ability for rapid prototyping, features of MyStudyPlanning could be implemented and demonstrated quickly, which sparked enthusiasm with employees and causes the application to be adopted across multiple faculties within the university. Since its inception in 2016, the MyStudyPlanning application rapidly grew and is now used by Master students in Computer Science, Electrical Engineering, Mathematics, Civil Engineering, Geosciences, and Aerospace Engineering.

8 Conclusion

As part of the research of Eelco and his Programming Languages group, there was a need for thorough case studies to show the advantages of the programming languages and techniques that were being developed. At the same time, Eelco encountered limitations in his work as an educator and researcher that could be mitigated by applying the developed DSLs and tools in these practical settings. These case studies therefore served two purposes: as practical case studies, and to help solve actual problems that he experienced in education and academia.

For many of his projects Eelco had a bottom-up approach: first apply the research to a single, ideally practical, case study and gradually expanding into other case studies from there. By identifying common patterns in the applications, it became more clear which boilerplate code to abstract into a DSL and on which areas to focus next. This was especially apparent in the WebDSL language [42], whose case studies grew into several usable applications that ended up being used in universities and academic conferences to solve real-world problems, including Researchr, EvaTool, WebLab, Conf Researchr, and MyStudyPlanning. Additionally, Eelco was excellent at promoting these applications both within and outside Delft University of Technology, which provided funding for the continued development of these applications.

Eelco's vision and method resulted in real-world impact of research in language design and engineering: MyStudyPlanning and EvaTool are used by many faculties across the Delft University of Technology. WebLab is currently used by more than 40 courses, both from Delft and other universities. Conf Researchr has been used to serve thousands of attendees at over a hundred editions of dozens of (co-located) conferences and workshops. The impact of these applications, some already over a decade old, keeps growing every year.

[11] `https://mystudyplanning.tudelft.nl/`

References

1 Martin Bravenboer and Eelco Visser. Concrete syntax for objects: domain-specific language embedding and assimilation without restrictions. In *OOPSLA*, pages 365–383, 2004. doi:10.1145/1028976.1029007.

2 Bram Crielaard, Chiel Bruin, and Taico Aerts. Native WebLab: Safe execution of native code in WebLab. Bachelor's thesis, Delft University of Technology, 2017.

3 Max M. de Krieger. Modernizing the WebDSL front-end: A case study in SDF3 and Statix. master's thesis. http://resolver.tudelft.nl/uuid:564b8471-631f-4831-a049-58b187425aed, 2022.

4 Eelco Dolstra. *The Purely Functional Software Deployment Model.* PhD thesis, Utrecht University, Utrecht, The Netherlands, January 2006.

5 Danny M. Groenewegen, Elmer van Chastelet, and Eelco Visser. Evolution of the WebDSL runtime: reliability engineering of the WebDSL web programming language. In *Programming*, pages 77–83, 2020. doi:10.1145/3397537.3397553.

6 Danny M. Groenewegen and Eelco Visser. Declarative access control for WebDSL: Combining language integration and separation of concerns. In *ICWE*, pages 175–188, 2008. doi:10.1109/ICWE.2008.15.

7 Danny M. Groenewegen and Eelco Visser. Integration of data validation and user interface concerns in a DSL for web applications. In *SLE*, pages 164–173, 2009. doi:10.1007/978-3-642-12107-4_13.

8 Danny M. Groenewegen and Eelco Visser. Integration of data validation and user interface concerns in a DSL for web applications. *SoSyM*, 12(1):35–52, 2013. doi:10.1007/s10270-010-0173-9.

9 Daco Harkes. *Declarative Specification of Information System Data Models and Business Logic.* PhD thesis, Delft University of Technology, Netherlands, 2019. doi:10.4233/uuid:5e9805ca-95d0-451e-a8f0-55decb26c94a.

10 Daco Harkes, Danny M. Groenewegen, and Eelco Visser. IceDust: Incremental and eventual computation of derived values in persistent object graphs. In *ECOOP*, 2016. doi:10.4230/LIPIcs.ECOOP.2016.11.

11 Daco Harkes, Elmer van Chastelet, and Eelco Visser. Migrating business logic to an incremental computing DSL: a case study. In *SLE*, pages 83–96, 2018. doi:10.1145/3276604.3276617.

12 Daco Harkes and Eelco Visser. Unifying and generalizing relations in role-based data modeling and navigation. In *SLE*, pages 241–260, 2014. doi:10.1007/978-3-319-11245-9_14.

13 Daco Harkes and Eelco Visser. IceDust 2: Derived bidirectional relations and calculation strategy composition. In *ECOOP*, 2017. doi:10.4230/LIPIcs.ECOOP.2017.14.

14 Jan Heering, P. R. H. Hendriks, Paul Klint, and Jan Rekers. The syntax definition formalism SDF - reference manual. *SIGPLAN*, 24(11):43–75, 1989. doi:10.1145/71605.71607.

15 Armijn Hemel. NixOS: the Nix based operating system INF/SCR-05-91. Master's thesis, University of Utrecht, 2006.

16 Zef Hemel, Danny M. Groenewegen, Lennart C. L. Kats, and Eelco Visser. Static consistency checking of web applications with WebDSL. *JSC*, 46(2):150–182, 2011. doi:10.1016/j.jsc.2010.08.006.

17 Zef Hemel, Lennart C. L. Kats, Danny M. Groenewegen, and Eelco Visser. Code generation by model transformation: a case study in transformation modularity. *Software and Systems Modeling*, 9(3):375–402, 2010. doi:10.1007/s10270-009-0136-1.

18 Zef Hemel and Eelco Visser. PIL: A platform independent language for retargetable DSLs. In *SLE*, pages 224–243, 2009. doi:10.1007/978-3-642-12107-4_17.

19 Zef Hemel and Eelco Visser. Declaratively programming the mobile web with Mobl. In *OOPSLA*, pages 695–712, 2011. doi:10.1145/2048066.2048121.

20 Karl Trygve Kalleberg and Eelco Visser. Fusing a transformation language with an open compiler. *Electronic Notes in Theoretical Computer Science*, 203(2):21–36, 2008. doi:10.1016/j.entcs.2008.03.042.

21 Lennart C. L. Kats, Karl Trygve Kalleberg, and Eelco Visser. Generating editors for embedded languages. integrating SGLR into IMP. In *LDTA*, April 2008.

22 Lennart C. L. Kats, Rob Vermaas, and Eelco Visser. Testing domain-specific languages. In *OOPSLA*, pages 25–26, 2011. doi:10.1145/2048147.2048160.

23 Lennart C. L. Kats and Eelco Visser. Encapsulating software platform logic by aspect-oriented programming: A case study in using aspects for language portability. In *SCAM*, pages 147–156, 2010. doi:10.1109/SCAM.2010.11.

24 Lennart C. L. Kats and Eelco Visser. The Spoofax language workbench: rules for declarative specification of languages and IDEs. In *OOPSLA*, pages 444–463, 2010. doi:10.1145/1869459.1869497.

25 Lennart C. L. Kats, Richard Vogelij, Karl Trygve Kalleberg, and Eelco Visser. Software development environments on the web: a research agenda. In *OOPSLA*, pages 99–116, 2012. doi:10.1145/2384592.2384603.

26 Gabriël Konat. *Language-Parametric Methods for Developing Interactive Programming Systems*. PhD thesis, Delft University of Technology, Netherlands, 2019. doi:10.4233/uuid:03d70c5d-596d-4c8c-92da-0398dd8221cb.

27 Gabriël Konat, Sebastian Erdweg, and Eelco Visser. Scalable incremental building with dynamic task dependencies. In *ASE*, pages 76–86, 2018. doi:10.1145/3238147.3238196.

28 Gabriël Konat, Lennart C. L. Kats, Guido Wachsmuth, and Eelco Visser. Declarative name binding and scope rules. In *SLE*, pages 311–331, 2012. doi:10.1007/978-3-642-36089-3_18.

29 Gabriël Konat, Vlad A. Vergu, Lennart C. L. Kats, Guido Wachsmuth, and Eelco Visser. The Spoofax name binding language. In *Companion to the 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2011, part of SPLASH 2012, Tucson, AR, USA, October 19 - 26, 2012*. ACM, 2012. doi:10.1145/2384716.2384748.

30 Marieke van der Tuin, A. Bastiaan Reijm, Tim K. de Jong, and Jeff Smits. WebLab project. Bachelor's thesis, Delft University of Technology, July 2013. URL: http://resolver.tudelft.nl/uuid:bb2d7a13-1bef-4545-bca0-f2b084a04240.

31 Hendrik van Antwerpen, Pierre Néron, Andrew P. Tolmach, Eelco Visser, and Guido Wachsmuth. A constraint language for static semantic analysis based on scope graphs. In *PEPM*, pages 49–60, 2016. doi:10.1145/2847538.2847543.

32 Hendrik van Antwerpen, Casper Bach Poulsen, Arjen Rouvoet, and Eelco Visser. Scopes as types. *PACMPL*, 2(OOPSLA), 2018. doi:10.1145/3276484.

33 Elmer van Chastelet. A domain-specific language for internal site search. Master's thesis, Delft University of Technology, 2013.

34 Elmer van Chastelet, Eelco Visser, and Craig Anslow. Conf.Researchr.Org: towards a domain-specific content management system for managing large conference websites. In *OOPSLA*, pages 50–51, 2015. doi:10.1145/2814189.2817270.

35 Sander van der Burg and Eelco Dolstra. Disnix: A toolset for distributed deployment. In *Third International Workshop on Academic Software Development Tools and Techniques (WASDeTT-3)*, September 2010.

36 Tim van der Lippe, Thomas Smith, Daniël A. A. Pelsmaeker, and Eelco Visser. A scalable infrastructure for teaching concepts of programming languages in Scala with WebLab: an experience report. In *SCALA*, pages 65–74, 2016. doi:10.1145/2998392.2998402.

37 Arie van Deursen, T. B. Dinesh, and Emma van der Meulen. The ASF+SDF meta-environment. In *amast*, pages 411–412, 1993.

38 Vlad A. Vergu, Pierre Néron, and Eelco Visser. DynSem: A DSL for dynamic semantics specification. In *RTA*, pages 365–378, 2015. doi:10.4230/LIPIcs.RTA.2015.365.

39 Sander Vermolen. *Software Language Evolution*. PhD thesis, Delft University of Technology, Delft, The Netherlands, October 2012.

40 Eelco Visser. A family of syntax definition formalisms. Technical Report P9706, Programming Research Group, University of Amsterdam, August 1997.

41 Eelco Visser. Meta-programming with concrete object syntax. In *GPCE*, pages 299–315, 2002. doi:10.1007/3-540-45821-2_19.

42 Eelco Visser. WebDSL: A case study in domain-specific language engineering. In *GTTSE*, pages 291–373, 2007. doi:10.1007/978-3-540-88643-3_7.

43 Eelco Visser. Performing systematic literature reviews with Researchr: Tool demonstration. Technical Report TUD-SERG-2010-010, Software Engineering Research Group, Delft University of Technology, Delft, The Netherlands, May 2010. URL: http://resolver.tudelft.nl/uuid:22b480a7-d09e-4ae6-abe7-9a5769e03c2b.

44 Eelco Visser, Zine-El-Abidine Benaissa, and Andrew P. Tolmach. Building program optimizers with rewriting strategies. In *ICFP*, pages 13–26, 1998. doi:10.1145/289423.289425.

45 Tobi Vollebregt, Lennart C. L. Kats, and Eelco Visser. Declarative specification of template-based textual editors. In *LDTA*, pages 1–7, 2012. doi:10.1145/2427048.2427056.