Delft University of Technology

Getting Things Done

The Eelco Way

van Deursen, Arie

DOI
10.4230/OASIcs.EVCS.2023.1

Publication date
2023

Document Version
Final published version

Published in
Eelco Visser Commemorative Symposium, EVCS 2023

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Getting Things Done: The Eelco Way

Arie van Deursen ✉ ⓘD
Delft University of Technology, The Netherlands

Abstract

Eelco Visser (1966–2022) was a leading member of the department of Software Technology (ST) of the faculty of Electrical Engineering Mathematics, and Computer Science (EEMCS) of Delft University of Technology. He had a profound influence on the educational programs in computer science at TU Delft, built a highly successful Programming Languages Group from the ground up, and used his research results to develop widely used tools and services that have been used by thousands of students and researchers for more than a decade. He realized all these successes not just alone, but in close collaboration with a range of people, who he convinced to follow his lead. In this short reflection, I look back at his achievements, and at the way in which he worked with others to bring ambitious ideas to successful reality.

1 Background

A common challenge in empirical software engineering is how to establish the value of a proposed new tool, or, fitting in the context of my dearly missed colleague Eelco Visser[1] in whose memory I write this essay, a domain-specific language. New methods and tools to help developers are proposed continuously, but how can one assess that such tools actually help as intended? Among various (quasi-)experimental approaches to assess this, one is the *removed treatment group design* [11], in which measurements are conducted not only before and while using a tool, but also after taking it away again. If removing the tool makes the developers complain, we can conclude that the tool was of value, and we can make the developers happy by giving the tool back to them.

Since April 2022, I have frequently felt like I was in the middle of such an experiment. Someone, for some reason, had decided they wanted to show how valuable Eelco was for the TU Delft, so they took him away from us. "Can we please stop the experiment and get Eelco back?" is what I think almost every day. Unfortunately, that is not an option.

In this essay, I look back at Eelco's time in Delft, his extra-ordinary achievements for the organization, and, the personal traits that enabled him and his co-workers to realize his successes.

2 Eelco at TU Delft

In 2006, Eelco Visser joined the TU Delft Software Engineering Research Group as an associate professor. I knew Eelco from our earlier collaborations [1, 3] in Amsterdam, and I was very happy he had accepted our offer to join us in Delft. Within the software engineering section he built up more and more programming languages activities, in such areas as parsing [2, 12], package management [6, 7], model-driven software evolution [5], web programming [13, 8], and language work benches [9, 14]. In 2013, he secured a highly

---

[1] https://avandeursen.com/2022/04/15/eelco-visser-1966-2022-en/

prestigious NWO Vici grant that enabled him to attract and employ multiple PhD students and postdocs. In the same year he was also promoted to the rank of full professor.[2] By 2015, Eelco's programming languages activities had grown so much that it was time to spin off a new section, the *Programming Languages Group*. This group officially started on January 1st, 2016, and, under the leadership of Eelco, grew to a total of five (including Eelco) faculty members and numerous postdocs, support staff, and bachelor, master, and PhD students.

January 1st, 2016, was also the day that I started in a new role, as head of the department of Software Technology. At that time, the department consisted of six sections, including the two aforementioned programming languages and software engineering sections. In his role as section leader, Eelco was also part of the *Management Team* of the department. Within this team, we discussed and decided about such issues as opening up our bachelor program to non-Dutch speaking students, handling the enormous increase in student intake in 2018, navigating students and employees safely through COVID-19, the increasingly prominent role of artificial intelligence and its impact on computer science education, and hiring and promotion decisions, effectively doubling the department in size over the years.

3   Lessons Learned from Eelco

Eelco played a leading role in Delft, both through the Programming Languages Group that he founded, and through his active membership of the department's management team. He had his personal way to get things done,[3] which I try to capture below.

EV1: *Play the long game*
Science is for eternity. I don't think I ever heard Eelco say this explicitly, but in all his actions he made it clear he considered this evident. Consequently, it is an obligation and a calling to deliver the best quality research possible, in terms of results and presentation. It also means that we must take as much time as needed to discuss, sharpen, and truly understand each other's research results. It explains why Eelco had the stamina to work on long term projects, such as the (re)design of the Syntax Definition Formalism SDF (25 years, [12, 2]), the WebDSL language and system for web engineering (20 years [8, 13]), or the award winning[4] Spoofax language workbench (15 years, [9, 14]). On me and many others, Eelco's unshakable belief in the longevity of his endeavors had a magnetic effect. In 2000 (this was before Wikipedia existed), I gladly joined his grand ambition to unify all knowledge on *Program Transformation* in a new wiki program-transformation.org [4].

EV2: *Articulate a bold vision*
You can only play the long game if you know what you want the future to look like. Again, I haven't heard Eelco use the word "vision" often, but Eelco was always able to articulate the long term ideal world. He made it clear that he expected computer science professors to be programming; that computer scientists need the best possible equipment; and that researchers need excellent housing enabling them to do concentrated work. He also envisioned a computer science bachelor program with a substantial amount of theory (despite some TU Delft resistance). He explained why programming education needs online learning tools beyond the standard Integrated Development Environment. And that the research community needs a memory of conference activities, and can save valuable time and costs by relying on a single shared conference management system. On all important matters, Eelco would have a vision ready at hand.

---

[2] https://eelcovisser.org/blog/2013/06/14/antoni-van-leeuwenhoek/
[3] I'm pretty sure Eelco also lectured me on David Allen's "Getting Things Done" methodology, but I don't remember whether he followed it.
[4] https://eelcovisser.org/blog/2021/02/08/spoofax-mip/

EV3: *Just do it*

While some may think having a bold vision that is hard to realize is of little use, Eelco was committed to his ideals. He just started doing what was necessary. If the vision involved software, his preferred approach would be to start programming himself. If he needed others, he would use his magnetism to convince them to participate. Short of money, he would try to start making expenses anyway, trusting he would get forgiveness easier than permission. And, most importantly, he would work towards early successes that would get people addicted, such as a first version of the WebLab online programming education system [10]. Once addicted, the department could not say no to requests for resources anymore.

EV4: *Take resistance as encouragement*

Eelco's mission was to bring change, not just to TU Delft, but to computer science at large. If you advocate change, resistance is to be expected. To Eelco, this was inherent to academic life. I believe he considered it a good sign, confirming that he was indeed trying to change the status quo. Eelco was always willing, even eager, to engage in debate and listen to arguments, in his calm and friendly way. But resistance alone would be confirmation that he was onto something, rather than a reason to change course.

EV5: *Embrace education*

A key factor in Eelco's success at TU Delft was his dedication to education. This came from deep within: He was truly devoted to sharing his love for computer science in general and programming in particular. Eelco would use any opportunity to increase the teaching load of the Programming Language Group. Whenever we were searching for a teacher for a course, he would volunteer. This included undergrad courses, such as algorithms and data structures, which he would use to develop the WebLab infrastructure. He would involve his postdocs in the teaching, giving them useful experience for their later academic careers. And, Eelco would use it as an argument to justify growth of the Programming Languages Group: "Our high teaching load forces us to let postdocs teach: we need more faculty members." An argument that we happily subscribed to.

EV6: *Align individual and departmental interests*

Eelco ensured his Programming Languages Group fared well, for example in terms of office space, equipment, traveling, support staff, and starting packages. In the fight for (scarce) resources, this is not as obvious as it may sound. Eelco always made it clear that he wanted improvements for his group based on general principles that should hold for *everyone*. He would volunteer to help realize his vision, trusting that when successful his group would also reap the benefits.

EV7: *Be supportive*

Eelco was well aware that there is a cost to his approach to academic life. As a full professor with plenty of responsibilities, it takes resolve to create time for programming and individual research. Resistance always leaves a mark. Persistence takes energy, as does boldness. I believe this was also why Eelco was always ready to listen to his students, group members, or peers when they needed support. He related to the struggles and doubts, and tried to help where he could. He was there for me, too. Besides formal meetings, we regularly chatted about our own lives and careers, as well as the ups and downs of the department. On departmental matters we did not always agree. But I always knew that no matter what I would do, he would be there to support me. He had my back, and not just mine.

4   Thank You Eelco

There is much more that can be said about Eelco's footprint. Here I attempted to highlight his unique style and personality, through which he made a lasting impact on computer science research and education, at Delft University of Technology as well as in the international research community. Thank you Eelco for all you've done for us – we miss you very much.

── **References** ──

1   Mark van den Brand, Arie van Deursen, Jan Heering, Hayco de Jong, Merijn de Jonge, Tobias Kuipers, Paul Klint, Leon Moonen, Pieter A. Olivier, Jeroen Scheerder, Jurgen J. Vinju, Eelco Visser, and Joost Visser. The ASF+SDF meta-environment: A component-based language development environment. In *Proc. 10th Int. Conf. on Compiler Construction (CC 2001)*, volume 2027 of *LNCS*, pages 365–370. Springer, 2001. `doi:10.1016/S1571-0661(04)80917-4`.

2   Luis Eduardo de Souza Amorim and Eelco Visser. Multi-purpose syntax definition with SDF3. In *18th International Conference on Software Engineering and Formal Methods (SEFM 2020)*, volume 12310 of *LNCS*, pages 1–23. Springer, 2020. `doi:10.1007/978-3-030-58768-0_1`.

3   Arie van Deursen, Jan Heering, and Paul Klint, editors. *Language Prototyping. An Algebraic Specification Approach*, volume 5 of *AMAST Series in Computing*. World Scientific, 1996.

4   Arie van Deursen and Eelco Visser. The Reengineering Wiki. In *6th European Conference on Software Maintenance and Reengineering (CSMR 2002), 11-13 March 2002, Budapest, Hungary, Proceedings*, pages 217–220. IEEE Computer Society, 2002. `doi:10.1109/CSMR.2002.995808`.

5   Arie van Deursen, Eelco Visser, and Jos Warmer. Model-driven software evolution: A research agenda. In *Proc. 1st Int. Workshop on Model-Driven Software Evolution*, pages 41–49, 2007. URL: `http://www.sciences.univ-nantes.fr/MoDSE2007/p19.pdf`.

6   Eelco Dolstra, Merijn de Jonge, and Eelco Visser. Nix: A safe and policy-free system for software deployment. In *Proceedings of the 18th Conference on Systems Administration (LISA 2004), Atlanta, USA, November 14-19, 2004*, pages 79–92. USENIX, 2004. URL: `http://www.usenix.org/publications/library/proceedings/lisa04/tech/dolstra.html`.

7   Eelco Dolstra, Eelco Visser, and Merijn de Jonge. Imposing a memory management discipline on software deployment. In *26th ACM/IEEE Int. Conf. on Software Engineering (ICSE 2004)*, pages 583–592, 2004. `doi:10.1109/ICSE.2004.1317480`.

8   Danny M. Groenewegen, Zef Hemel, Lennart C. L. Kats, and Eelco Visser. WebDSL: a domain-specific language for dynamic web applications. In *Companion to the 23rd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2008)*, pages 779–780. ACM, 2008. `doi:10.1145/1449814.1449858`.

9   Lennart C. L. Kats and Eelco Visser. The Spoofax language workbench: rules for declarative specification of languages and IDEs. In *Proc. 25th Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. ACM, 2010. `doi:10.1145/1869459.1869497`.

10 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 *Proceedings of the 7th ACM SIGPLAN Symposium on Scala, SCALA@SPLASH 2016*, pages 65–74. ACM, 2016. `doi:10.1145/2998392.2998402`.

11 William Shadish, Thomas Cook, and Donald Cambell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, Boston, 2002.

12 Eelco Visser. *Syntax Definition for Language Prototyping*. PhD thesis, University of Amsterdam, September 1997.

13 Eelco Visser. WebDSL: A case study in domain-specific language engineering. In *Generative and Transformational Techniques in Software Engineering II (GTTSE) 2007*, volume 5235 of *LNCS*, pages 291–373. Springer, 2007. `doi:10.1007/978-3-540-88643-3_7`.

14 Guido Wachsmuth, Gabriël Konat, and Eelco Visser. Language design with the Spoofax language workbench. *IEEE Software*, 31(5):35–43, 2014. `doi:10.1109/MS.2014.100`.