Teaching Statement

Ralf Lämmel

December 2020

Abstract

Section 1 summarizes my **courses taught** in the last 10 years.
Section 2 identifies **additional topics** I want to cover in teaching.
Section 3 characterizes my evolving and maturing **teaching approach**.
Section 4 provides evidence that '**research meets teaching**' in my work.

1 Courses Taught in the Last 10 Years

See `http://www.softlang.org/teaching` for some traces, but much of the courses are now conducted via eLearning platforms that are only accessible through the university intranet.

Object-Oriented Programming and Modeling ((OO)PM) Bachelor 1st semester introduction to computer science including basics of programming and modeling with typically 250–400 students; 9-11 ECTS. I run this course every winter semester.

Introduction to Functional Programming (FP) Bachelor-level course with Haskell as the functional programming language with approximately 100 students; 6 ECTS. I run this course every summer semester.

Theory of Programming Languages (PLT) Advanced Bachelor-level, introductory Master-level course with typically 30-40 students; 6 ECTS. I run this course every summer semester.

Software Language Engineering (SLE) Master-level choice subject with coverage of domain-specific languages, compiler construction and others with typically 10 students in a research-oriented format; 6 ETCS. I run this course approximately once per year.

Mining Software Repositories (MSR) Master-level choice subject with typically 10 students in a research-oriented format; 6 ETCS. I run this or the ESE course (see below) approximately once per year.

Empirical Software Engineering (ESE) Master-level choice subject with typically 10 students in a research-oriented format; 6 ETCS. I run this or the MSR course (see above) approximately once per year.

Programming Techniques and Technologies (PTT) Advanced Bachelor-level choice subject with typically 30-50 students; 6 ETCS. I do not currently (plan to) run this course.

2 Additional Course Topics of Interest

I hope to set up courses, for some of the following topics, eventually.

- I my recent research, most notably in R&D on software engineering and infrastructure at Facebook, I leveraged *machine learning, data science, data engineering and process mining*. I expect to also serve these topics in teaching. Currently, I am working on a related textbook: 'Software (Systems) Analytics'.
- I have modest interests in *algorithms and data structures*, but I never covered this topic deeply in teaching – because of existing duties at the Bachelor level including the aforementioned course (OO)PM, which covers a lightweight version of algorithms and data structures.
- Based on my research over the last few years, I have acquired knowledge and interest in the *semantic web*, e.g., in terms of semantic data models (RDF, ontologies, description logics) and graph query languages, schema languages, and applications such as Linked Data or Wikidata. I expect to also serve these topics in teaching. The aforementioned textbook, 'Software (Systems) Analytics', is likely to leverage some of these topics, e.g., the use of knowledge graphs for modeling some aspects of the systems or companies being analyzed.
- An interdisciplinary course which combines elements of modeling and domain-specific languages with a domain such as *IoT* or *robotics* would be very interesting in my opinion; it could, in fact, be an alternative approach to the SLE course that I am running. This kind of combination has been successfully used elsewhere, for example, at Chalmers by Thorsten Berger and collaborators.
- Because of my commitment to Software Language Engineering and due to significant obligations in the Bachelor education, I have not yet covered classical (or modern) *compiler construction* in depth in a lecture, but I continue to be interested in doing so.

3 Approach to Teaching

Social teaching In the time of high-quality online courses (e.g., MOOCs), it is important that university courses are more than just means of delivering knowledge to students and running exams; the modern university hosts communities that are based on a joint learning experience, subject to close interaction between students and faculty. I enjoy being part of this sort of community.

Asynchronous teaching Long before Corona, I have been interested in enabling asynchronous teaching to some degree. For example, the typical Bachelor education in Germany involves quite many lectures and exercise courses. Removal of a presence condition (e.g., by means of recorded lectures or online exercise groups) provides more flexibility to students.

Open teaching I fundamentally believe that teaching material and course designs should be open. Whenever possible, I share course material openly and I also reuse material and designs from elsewhere. I also take the initiative in bringing together teaching efforts; see, for example, SLEBOK. I have (and had) many popular YouTube videos online on subjects of functional programming, software language engineering, parallel programming, programming language theory, and programming technologies.

60 minutes or less The classical length of 90 minutes (in Germany) is too long for an effective learning experience, according to attention spans understood in physiology. I have gradually departed from the classical 90 minutes model, by initially injecting a break and eventually just cutting short lecturing time; the remaining time can be used more effectively for opt-in discussions. Splitting up lectures into smaller units (MOOC style) helps — especially in combination with the asynchronous mode.

Flipped classroom Especially in my course on programming language theory, I wanted the students to engage with the content through reading. The time in the class is then used as follows. i) The lecturer poses questions which may also have been known to the students ahead of the class so that these questions guide preparation. ii) The students can also ask questions — especially in relation to ongoing or checked homework assignments. iii) The lecturer may go through additional examples. As a result, no classical separation between lecture and exercise slot is needed. Instead, there is just one type of meeting that covers all the aspects i)–iii).

Live programming Especially in my course on functional programming, live programming helps with finding a good pace of delivering programming techniques and foundations. The course content is structured and published on a wiki in a manner that the live programming session is essentially a selective walk through the content. (In my experience, it is harder to maintain student

attention with slides that essentially present many programs; one easily ends up showing too much program code too quickly.)

Fewer exams In my Master-level, research-oriented courses, I have stepped away from any use of an oral or written exam. Instead, the courses require several packages of project work from the students. In this manner, I have increased engagement with the course and transparency or predictability regarding grading.

Consultations In courses with the project type of work, I make sure that students get quality feedback from me or other teaching staff at a point in time well ahead of the final deadline, thereby making sure students start early and they get a chance to improve and to learn from the feedback. To this end, I use the time normally scheduled for exercise course elements in a more classical course design.

Clear communication I have to come to understand how important it is to be very clear on communication. In particular, expectations (e.g., for the project type of work) need to be stated very clearly (e.g., by a metamodel). Also, feedback regarding any intermediate or final results of the students must be systematically structured and clearly communicated. Finally, results of course evaluations, when they are performed, must be discussed, also with the objective of resolving any reported problems.

4 Research Meets Teaching

Some of my research effectively concerns teaching or education (or knowledge engineering, in a broad sense). In fact, some of my research is even directly used by my team and elsewhere in teaching.

The Software Languages Book This textbook, published by Springer in 2018, is used in related courses at various universities internationally – also based on chapters I released ahead of the Springer publication. I also published popular appetizer videos for most of the book's topics. The book received the Choice Award as an Outstanding Academic Title for 2019:

> "The book's 12 chapters cover a wealth of information by adopting a formal point of view that supplies the rigor needed for such an extensive treatment while offering plenty of self-learning opportunities through exercises and additional references. [...] This book is also useful as the first step in a more in-depth exploration of specific topics, such as metaprogramming, as it provides the references needed for a comprehensive literature review. [...] Summing Up: Highly recommended. Advanced undergraduates through faculty and professionals." (L. Benedicenti, Choice, Vol. 56 (9), May, 2019)

SLE courses As a co-founder of the Software Language Engineering community, I have also taken a lead in designing courses in this area. This has, for example, also resulted in a corresponding Dagstuhl seminar on the *SLE Body of Knowledge*.

- Benoît Combemale, Ralf Lämmel, and Eric Van Wyk. SLEBOK: the software language engineering body of knowledge (dagstuhl seminar 17342). *Dagstuhl Reports*, 7(8):45–54, 2017
- Anya Helene Bagge, Ralf Lämmel, and Vadim Zaytsev. Reflections on courses for software language engineering. In Birgit Demuth and Dave R. Stikkolorum, editors, *Proceedings of the MODELS Educators Symposium co-located with the ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014), Valencia, Spain, September 29, 2014*, volume 1346 of *CEUR Workshop Proceedings*, pages 54–63. CEUR-WS.org, 2014

Software chrestomathies A chrestomathy is essentially a collection of literary pieces useful for learning. A software or program chrestomathy is about programming languages rather than natural languages. My team has been developing software chrestomathies and they are used in courses on functional programming and software language engineering.

- Jean-Marie Favre, Ralf Lämmel, Thomas Schmorleiz, and Andrei Varanovich. 101companies: A community project on software technologies and software languages. In Carlo A. Furia and Sebastian Nanz, editors, *Objects, Models, Components, Patterns - 50th International Conference, TOOLS 2012, Prague, Czech Republic, May 29-31, 2012. Proceedings*, volume 7304 of *Lecture Notes in Computer Science*, pages 58–74. Springer, 2012
- Ralf Lämmel, Thomas Schmorleiz, and Andrei Varanovich. The 101haskell chrestomathy: A whole bunch of learnable lambdas. In Rinus Plasmeijer, editor, *Proceedings of the 25th Symposium on Implementation and Application of Functional Languages, Nijmegen, The Netherlands, August 28-30, 2013*, page 25. ACM, 2013
- Ralf Lämmel. Software chrestomathies. *Sci. Comput. Program.*, 97:98–104, 2015
- Simon Schauss, Ralf Lämmel, Johannes Härtel, Marcel Heinz, Kevin Klein, Lukas Härtel, and Thorsten Berger. A chrestomathy of DSL implementations. In Benoît Combemale, Marjan Mernik, and Bernhard Rumpe, editors, *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2017, Vancouver, BC, Canada, October 23-24, 2017*, pages 103–114. ACM, 2017

Technology modeling We use such megamodels (or models of linguistic architecture) in teaching, as they are highly structured, actionable, and validated models for programming technologies as opposed to informal descriptions.

- Ralf Lämmel. Megamodels on the catwalk. Keynote at MODELSWARD 2021 `http://www.modelsward.org/`, 2021
- Jean-Marie Favre, Ralf Lämmel, and Andrei Varanovich. Modeling the linguistic architecture of software products. In Robert B. France, Jürgen Kazmeier, Ruth

Breu, and Colin Atkinson, editors, *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*, volume 7590 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2012

- Jean-Marie Favre, Ralf Lämmel, Martin Leinberger, Thomas Schmorleiz, and Andrei Varanovich. Linking documentation and source code in a software chrestomathy. In *19th Working Conference on Reverse Engineering, WCRE 2012, Kingston, ON, Canada, October 15-18, 2012*, pages 335–344. IEEE Computer Society, 2012
- Ralf Lämmel and Andrei Varanovich. Interpretation of linguistic architecture. In Jordi Cabot and Julia Rubin, editors, *Modelling Foundations and Applications - 10th European Conference, ECMFA@STAF 2014, York, UK, July 21-25, 2014. Proceedings*, volume 8569 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2014
- Johannes Härtel, Lukas Härtel, Ralf Lämmel, Andrei Varanovich, and Marcel Heinz. Interconnected linguistic architecture. *Art Sci. Eng. Program.*, 1(1):3, 2017
- Marcel Heinz, Ralf Lämmel, and Andrei Varanovich. Axioms of linguistic architecture. In Luís Ferreira Pires, Slimane Hammoudi, and Bran Selic, editors, *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2017, Porto, Portugal, February 19-21, 2017*, pages 478–486. SciTePress, 2017
- Marcel Heinz, Johannes Härtel, and Ralf Lämmel. Reproducible construction of interconnected technology models for EMF code generation. *J. Object Technol.*, 19(2):8:1–25, 2020

Analyzing Wikipedia Related work by my team has also been motivated by teaching in the sense that we need, for example, reliable categorical data for the purpose of chrestomathies and technology models.

- Ralf Lämmel, Dominik Mosen, and Andrei Varanovich. Method and tool support for classifying software languages with wikipedia. In Martin Erwig, Richard F. Paige, and Eric Van Wyk, editors, *Software Language Engineering - 6th International Conference, SLE 2013, Indianapolis, IN, USA, October 26-28, 2013. Proceedings*, volume 8225 of *Lecture Notes in Computer Science*, pages 249–259. Springer, 2013
- Marcel Heinz, Ralf Lämmel, and Mathieu Acher. Discovering indicators for classifying wikipedia articles in a domain - A case study on software languages. In Angelo Perkusich, editor, *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019*, pages 541–706. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019