Teaching Software Product Lines: A Snapshot of Current Practices and Challenges (Journal-First Abstract)*

Mathieu Acher
Univ Rennes, Inria, CNRS, IRISA
Rennes, France
mathieu.acher@irisa.fr

Roberto E. Lopez-Herrejon
Ecole de Technologie
Supérieure
Montreal, Canada
Roberto.Lopez@etsmtl.ca

Rick Rabiser
CDL MEVSS
Johannes Kepler University
Linz, Austria
rick.rabiser@jku.at

ABSTRACT

This extended abstract summarizes our article entitled "Teaching Software Product Lines: A Snapshot of Current Practices and Challenges" published in the ACM Transactions on Computing Education, vol. 18 in 2017 (http://doi.acm.org/10.1145/3088440). The article reports on three initiatives we have conducted with scholars, educators, industry practitioners, and students to understand the connection between software product lines and education and to derive recommendations for educators to continue improving the state of practice of teaching SPLs.

CCS CONCEPTS

• **General and reference → Surveys and overviews**; • **Social and professional topics → Computing education**; • **Applied computing → Education**; • **Software and its engineering → Software product lines**;

KEYWORDS

Software Product Lines, Variability Modeling, Software Product Line Teaching, Software Engineering Teaching

ACM Reference Format:
Mathieu Acher, Roberto E. Lopez-Herrejon, and Rick Rabiser. 2018. Teaching Software Product Lines: A Snapshot of Current Practices and Challenges (Journal-First Abstract). In *22nd International Systems and Software Product Line Conference - Volume A (SPLC '18), September 10–14, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SUMMARY

Teaching software product lines (SPLs) and variability is challenging; there are several reasons. First, software engineering itself is a relatively young discipline: it is still an open problem to find good teaching methods and the correct place in a rather large and evolving curriculum [8]. Second, SPL engineering encompasses a variety of topics, including requirements analysis, design, implementation, testing, modeling, and evolution. Another related challenge is to prepare teaching material – based on existing books, tools, and research issues have been reported in other software engineering communities, e.g., global software engineering [2], model-driven engineering [9], software language engineering [5]. To the best of our knowledge, however, there is no published research work (except our own work) related to the teaching of SPLs. Besides, many empirical studies and systematic reviews of both research and industrial practices of SPLs have been performed [3, 4, 6, 7] (to name but a few), but none of them addresses the teaching aspect.

Currently, it is unclear how SPLs are taught, what are the possible gaps and difficulties faced, what are the benefits, or what is the material available. As a result there is an important educational gap between all stakeholders of the SPL community – researchers, educators, students, and industry practitioners. In our article [1] we present and discuss the results of three initiatives we conducted to assess the state of practice of teaching SPLs, i.e., an online survey on teaching SPLs we performed with 35 scholars, another survey on learning SPLs we conducted with 25 students, as well as two workshops held at the International Software Product Line Conference in 2014 and 2015 with both researchers and industry practitioners.

We also discuss how to continue improving teaching SPLs, e.g., creating a virtual meeting place for the community (e.g., http://teaching.variability.io/) and developing a baseline curriculum. Building upon our empirical results, we make concrete recommendations for SPL educators and for the SPL community in general.

*The full version of this work was published as journal article [1].

SPLC '18, September 10–14, 2018, Gothenburg, Sweden
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

REFERENCES

[1] Mathieu Acher, Roberto E. Lopez-Herrejon, and Rick Rabiser. 2017. Teaching Software Product Lines: A Snapshot of Current Practices and Challenges. *ACM Transactions on Computing Education* 18, 1 (2017), 2:1–2:31.
[2] Sarah Beecham, Tony Clear, John Barr, Mats Daniels, Michael Oudshoorn, and John Noll. 2017. Preparing Tomorrow's Software Engineers for Work in a Global Environment. *IEEE Software* 34, 1 (2017), 9–12.
[3] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wasowski. 2013. A Survey of Variability Modeling in Industrial Practice. In *7th Int'l Workshop on Variability Modelling for Software-Intensive Systems.* ACM, 7–14.
[4] Lianping Chen and Muhammad Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology* 53, 4 (2011), 344–362.
[5] Benoît Combemale, Ralf Lämmel, and Eric Van Wyk. 2017. SLEBOK: The Software Language Engineering Body of Knowledge (Dagstuhl Seminar 17342). *Dagstuhl Reports* 7, 8 (2017), 45–54. DOI: http://dx.doi.org/10.4230/DagRep.7.8.45
[6] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wasowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *6th Int'l Workshop on Variability Modeling of Software-Intensive Systems.* ACM, 173–182.
[7] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2014. Variability in Software Systems – A Systematic Literature Review. *IEEE Transactions on Software Engineering* 40, 3 (2014), 282–306.
[8] Carlo Ghezzi and Dino Mandrioli. 2005. The challenges of software engineering education. In *Proc. of the Int'l Conf. on Software Engineering.* Springer, 115–127.
[9] Ludwik Kuzniarz and Luiz Eduardo G Martins. 2016. Teaching Model-Driven Software Development: A Pilot Study. In *Proc. of the 2016 ITiCSE Working Group Reports.* ACM, 45–56.