A Survey on Teaching of Software Product Lines

Mathieu Acher, Roberto Erick Lopez-Herrejon, Rick Rabiser

A Survey on Teaching of Software Product Lines

Mathieu Acher
University of Rennes 1, Inria
Rennes, France
mathieu.acher@irisa.fr

Roberto E. Lopez-Herrejon
Johannes Kepler University
Linz, Austria
roberto.lopez@jku.at

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

ABSTRACT

With around two decades of existence, the community of Software Product Line (SPL) researchers and practitioners is thriving as can be attested by the extensive research output and the numerous successful industrial projects. Education has a key role to support the next generation of engineers to build highly complex SPLs. Yet, it is unclear how SPLs are taught, what are the possible missing gaps and difficulties faced, what are the benefits, or what is the material available. In this paper, we carry out a survey with over 30 respondents with the purpose of capturing a snapshot of the state of teaching in our community. We report and discuss quantitative as well as qualitative results of the survey. We build upon them and sketch six concrete actions to continue improving the state of practice of SPL teaching.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education; D.2.9 [**Software Engineering**]: Management—*Life cycle*

Keywords

Software Product Lines, Software Engineering Teaching

1. INTRODUCTION

Virtually any system – hardware systems, operating systems, user interfaces, or a source code base – faces the need to be available in multiple variants. *Software Product Line* (SPL) engineering has emerged as the means to efficiently produce and maintain multiple similar software variants, exploiting their common properties and managing their differences [7, 15, 16, 22, 32, 34, 36]. With around two decades of existence, SPL is now well-established in research and industry [10]. The body of knowledge collected and organized by the SPL research community is still growing. Also, the scope of the community continuously broadens to other areas such as software ecosystems [12] and dynamic adaptive systems [9, 23, 31]. The long-term goal of the community is to provide systematic engineering methods, languages, and tools to assist practitioners in building well-structured and customizable systems.

However, without any effort for disseminating this knowledge, engineers of tomorrow are unlikely to be aware of the issues faced when engineering SPLs (or configurable systems) – up to the point they will not recognize this kind of systems. In turn, they will not use appropriate techniques and face problems such as scalability that the SPL community perhaps already studied or solved.

We believe education has a key role to play. The teaching of SPLs can enable the next generation of engineers to build highly complex, adaptive, and configurable software systems more frequently reported in the industry [10, 19, 35] or observed in the open source community [11, 29]. Also, research can benefit from teaching: students can be involved in controlled experiments and researchers involved in teaching can identify potential missing gaps of SPL engineering tools and techniques.

Teaching SPLs is challenging. Software engineering itself is a relatively young discipline: it is still challenging to find good teaching methods and the correct place in a rather large and evolving curriculum [21]. Moreover, SPL engineering encompasses a variety of topics, including requirements analysis, design, implementation, testing, and evolution. Another related challenge is to prepare teaching material – based on existing books, tools, and research papers – suitable for attracting students.

To the best of our knowledge, there is no published research work related to the teaching of SPLs. Dedicated venues exist for addressing the challenges, techniques, and best practices of software engineering teaching (e.g., SEET at ICSE, the CSEE&T conference, or the Educators Symposium at MODELS). However, the specificity of teaching SPLs and variability has not been discussed so far. Empirical studies and systematic reviews of both research and industrial practices of SPLs have been performed [5, 6, 8, 10, 14, 17, 18, 20, 24–26, 33], but none of them addresses the teaching aspect of SPL engineering.

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. To address this gap, we conducted an online survey with the purpose of capturing a snapshot of the state of teaching in our community. Our goal was to identify common threads, interests, and problems and build upon them to further understand and hopefully strengthen this important need in our community. This paper reports, analyzes, and discusses 34 responses of people involved in SPL teaching. The remainder of the paper is structured as follows. Section 2 describes the research method. Section 3 performs a qualitative and quantitative analysis of the answers. Section 4 highlights important points that need to be addressed in the future. We discuss threats to validity in Section 5 and conlude the paper in Section 6.

2. SURVEY DESCRIPTION

The goal of our survey is to qualitatively and quantitatively analyze the state of practice of SPL teaching.

Obviously, survey participants should be aware of SPL engineering and, hopefully, have experience in teaching. We considered that a public dissemination of the survey (e.g., on general mailing lists) can reach a large audience but also can have counter effects. Irrelevant or incomplete answers can be obtained since the participants are not concerned at all with SPLs or teaching. Also, relevant respondents (i.e., those teaching SPLs) may not be reachable by this medium.

We thus decided to directly contact potential participants of the survey based on our own networks and searching SPL courses online. We first selected a pool of SPL researchers we already know who teach SPLs and then added additional people found by searching for SPL courses online. This led to a list of 41 contacts.

To further extend this list, we reviewed the list of authors and co-authors of last years' accepted SPLC and VaMoS papers and selected people not yet in our list. Eventually, we augmented the original list with 51 more contacts. The final list contained 92 contacts, from which we were confident they are involved in SPL teaching. We ignored, at this step, if the contacts teach SPLs as a self-contained course or as part of another lecture such as courses on Software Engineering in general. Furthermore, when sending out the survey, we asked recipients to forward the survey to their own contacts, who also teach SPLs.

We set up the survey as an on-line questionnaire (it can be found at: `http://www.surveygizmo.com/s3/1342346/Teaching-Software-Product-Lines`). Before sending the link to our 92 contacts, we created a first draft of the questionnaire and informally presented it to diverse colleagues having experience in SPL teaching, e.g., at SPLC in August 2013 in Tokyo. Based on colleagues' remarks, we refined some of the questions, mainly to make them more precise.

The resulting questionnaire comprised 13 questions intended to collect quantitative results about the context of teaching SPLs, i.e., country, institution, department, experience of the instructors, targeted audience, length of the course, primary literature used, tools used, and parts of the SPL lifecycle covered. For each quantitative question, we provided a set of possible answers to select from and – for most of these questions – we allowed survey participants to select 'other' and specify a more accurate or additional answer themselves. The questionnaire, however, also comprised five open questions to encourage more qualitative and constructive remarks about the experience of SPL teaching. More specifically, we asked participants what the most challenging part of teaching SPLs is, what suggestions they have to improve the state of teaching SPLs, how SPL teaching can impact SPL research and industrial practice, and if they have any other comments, concerns, or remarks relevant to teaching SPLs or the survey.

From the 92 people we directly approached via mail, asking them to fill out the survey, we received 34 complete responses (with all questions of the survey completed) and 15 partial responses (only some questions completed). We decided to only take complete responses into account in our quantitative analyses but double checked partial responses to find additional qualitative suggestions.

For the 13 quantitative questions, we defined five categories to present the results in an aggregated manner. More specifically, the category ***'respondents and their institutions'*** aggregates the answers to the five questions on the name and country of the respondents' institution, the type of institution, the department, and the experience of respondents in SPL research and teaching. Category ***'primary literature used'*** presents respondents' answers on what primary literature they use in their course and category ***'tools used'*** what tools respondents mentioned. Category ***'course length, audience, and practical time'*** aggregates the answers to five questions on the context (self-contained course vs. part of another course), the audience, the total lecture time, the total practical time, and the length of the course. The last category ***'parts of the SPL lifecycle covered'*** reports what parts of the SPL lifecycle respondents cover in their courses.

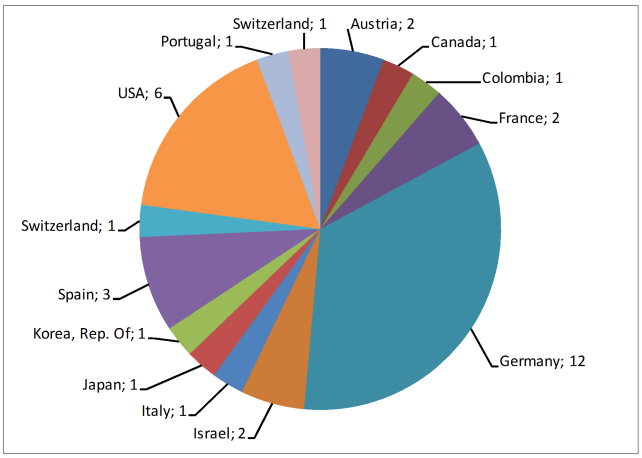

Figure 1: Respondents' locations.

3. SURVEY RESULTS

In this section we summarize the results gathered in our survey and highlight some of the important findings.

3.1 Respondents and their institutions

Figure 1 shows the 14 different countries represented by our 34 survey respondents. Not surprisingly, these countries are also key players in SPL research. Respondents' experience in SPL research is on average over 10 years (median: 8; min: 0; max: 37) while their experience in SPL teaching is on average over 6 years (median: 5; min: 1; max: 20). Surprisingly, almost half of all respondents said they have the same experience in research as in teaching SPLs. A possible interpretation is that SPL teaching is probably performed concurrently with research and thus it might not always be the result of mature research experience.

Regarding the types of institutions: Most (15) are research-focused or (9) teaching-focused (colleges), seven have both focuses, and three are industrial institutions. SPL is mainly taught at computer science (13 responses) or software engineering departments (9 responses) and less in information technology departments (2 responses). Other departments (10 responses) mentioned by respondents were: software science and technologies, computer science and computer engineering, industrial and management systems engineering, information systems, mathematics and informatics, systems and computing engineering, informatics, as well as comput-

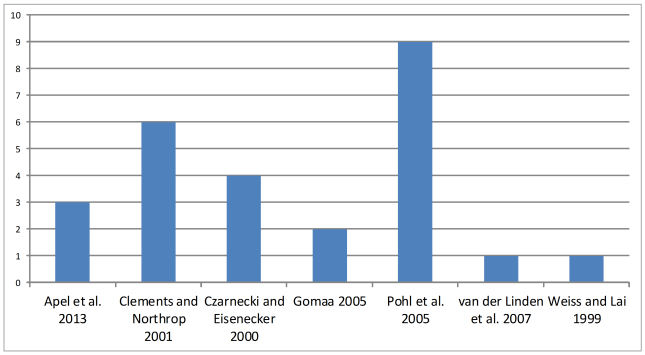

Figure 2: Primary literature (books) used in teaching SPLs.

ing and information systems.

About half (16) of the respondents hold full and self-contained SPL courses while the other half (18) teach SPL topics as part of other courses, among them are: software engineering (8), requirements engineering (4), automated software design (1), principles of software construction (1), domain engineering (1), software architecture (1), and factory development of software (1). Given the relative novelty of the topic, it is not surprising that over half of respondents teach SPL topics as part of other courses.

3.2 Primary literature used

Figure 2 shows the key primary literature (text books) used in teaching SPLs in alphabetical order, i.e., Apel et al. [7], Clements and Northrop [15], Czarnecki and Eisenecker [16], Gomaa [22], Pohl et al. [32], van der Linden et al. [34], and Weiss and Lai [36]. Most (28) survey respondents use these books, many (25) use research papers, and only about a third (12) use case studies. The respondents mentioned case studies by van der Linden et al. 2007 [34] (two respondents); BigLever case studies (one respondent); Renault, STAGO, LINUX, and SPLOT models (one respondent); and own case studies (one respondent). Several research papers were also listed by some survey respondents; however, only one paper was mentioned more than once, i.e., the original FODA report by Kang et al. [27]. The other papers seem to be selected based on teachers' personal preferences and their topics covered.

While using text books and some research papers for teaching is pretty much standard in computer science, using case studies is also essential, especially for teaching SPLs. Why only about a third of respondents use case studies to teach SPLs should be further investigated (cf. Section 3.6).

3.3 Tools used

Adequate tooling support is essential for the success in applying SPL techniques. Thus, we were interested in finding the tools used in the courses: Most (26) respondents said they use SPL tools while only about a quarter (8) said they don't use any. Table 1 shows an overview of the tools mentioned. When compared with the survey by Berger et al. [10], we see a similar trend. Industry-strength tools such as BigLever's GEARS and Pure Systems's pure::variants are used more commonly, while the rest of the tools seem to be spread across research groups, each using its own research prototypes. However, FeatureIDE is more widespread than other research prototypes.

SPL Tool	# uses
None	8
FeatureIDE	6
BigLever's GEARS	4
FeatureHouse	3
pure system's pure::variants	3
AHEAD	2
CIDE	2
CVL	2
Feature Modelling Plug-in (FMP)	2
DOPLER	1
EasyProducer	1
FaMa	1
Familiar	1
FeatureMapper	1
Munge	1
SPLAR	1
SPLOT	1
VARIAMOS	1
Varmod	1
C++, Metaprogramming, Software Generators, Xtext, MPS	1
Different tools for creating feature diagrams and UML-based models	1
Feature modeling UML tools extended for SPL	1
Haskell-Embedded Variation DSL	1
Own research prototypes	1

Table 1: SPL Tools used for teaching SPLs.

3.4 Course length, audience, practical time

Most courses are held over a semester (26), followed by those offered 'on demand' (i.e., depending on number of registrations) (3), and week-long courses (2), quarter (1), 1-2 days seminars (1), and with a flexible schedule, i.e., held one week (blocked) or over a whole semester (1). The average course length (for a course held for one semester) is 24 hours (min: 1 hour; max: 120 hours).

Figure 3 depicts the audience of SPL courses and shows a tendency, i.e., most courses are held for graduates and/or industry people. This is also reflected by the relation of practical time to overall course time (i.e., exercises, working on a concrete project), which is 65%. This means almost two thirds of course time is spent in practical exercises and work. One respondent even specified 2.5 times more practical time than lecture time. However, there also are eight courses where no practical time is allotted.

3.5 Parts of the SPL lifecycle covered

Table 2 depicts the parts of the SPL development lifecycle covered in SPL courses as specified by survey respondents. More specifically, modeling, requirements engineering, and implementation are the main focus of teaching SPLs. Processes and Maintenance & Evolution are also essential. Reverse Engineering and Adoption along with Testing are topics not as frequently covered, despite the extensive research and results on both areas. Other parts of the SPL development lifecycle commented by respondents as being covered in their courses are: business and organizational foundations (4), verification (2), architecture and design (2), decision models (1), measurement and analysis (2), tools (1),

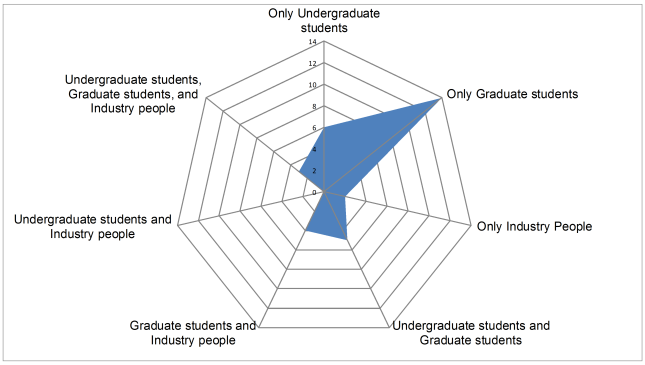

Figure 3: Audience of SPL courses.

Topic	Department			Total
	CS	SE	OT	
Requirements engineering	7	4	12	23
Testing	3	1	4	8
Modelling	9	8	13	30
Implementation	9	8	8	25
Maintenance & Evolution	9	3	5	17
Reverse engineering & SPL Adoption	7	2	1	10
Processes	5	6	8	19
Other	5	2	3	10

CS: Computer Science, SE: Software Engineering, OT: Other departments.

Table 2: Parts of the SPL development lifecycle covered in SPL courses.

scoping (1).

Furthermore we divide the topics by departments – computer science, software engineering and other – and highlight for each topic the department that got the highest count. We found a clear and even split between computer science and other departments each of which got the highest count in half of the topics. This find may suggest a trend in the courses taught at computer science departments towards implementation, maintenance and evolution, reverse engineering and adoption, and other miscellaneous topics that may include more advanced research. In contrast, the courses offered at other departments seem to lean towards requirements engineering, modelling and processes. It would be interesting to corroborate or disprove these trends and their potential causes in the future work.

3.6 Challenges of teaching SPL

In our survey we posed the open question *'What is the most challenging part for teaching SPL? (e.g., administrative delays, acceptance in the curriculum, lack of material)'.* Only four respondents said that they see no challenges at all, the rest at least mentioned one challenge. Analyzing and categorizing respondents' answers resulted in the following key challenges for teaching SPL:

Lack of and availability of SPL tools. Five respondents said that a key challenge is that many SPL tools are not freely available via an open source license and those that are available are often poorly documented research prototypes. Also, two respondents pointed out that there is a lack of integrated tooling to show a complete toolchain, i.e., SPL tools often focus only on one part of the SPL development lifecycle such as feature modeling.

Lack of and availability of well-documented real-world examples and case studies suitable for teaching. Almost half of the respondents named the lack of and availability of examples and case studies as a key challenge for teaching SPLs. Some responded that they would require good textbook examples, i.e., small, plausible, yet non-trivial examples to entice students. Others said that large-scale industrial case studies would be better. Agreement among respondents was that good, well-documented examples and case studies are hard to find despite the many available SPL textbooks and research papers describing case studies. Most available examples and case studies have not been developed and described for the purpose of teaching SPLs but rather to report interesting findings in the research community. Available material should be restructured, prepared and organized for teaching.

Complexity of the subject and required background knowledge. Nine respondents said that SPL is a complex topic including many aspects (in addition to software engineering) making it hard to teach to computer science students and requires students to have a strong background knowledge. One respondent summarized the challenge quite nicely: *even software engineering can be hard to teach as developing large-scale systems does not connect to students hands-on experience of developing rather small solutions. Teaching SPL means SE for many systems, this does even less relate to students' experiences.* Respondents reported that their students have a hard time understanding how to develop large-scale systems and usually have not much experience with big projects. They require background knowledge in diverse areas such as model-driven architectures or constraint programming.

Acceptance on the curriculum and opening the mind of students for the topic. Two respondents said they had difficulties getting the course accepted in their curricula. This might be related with the former challenge, i.e., complexity of the subject and required previous knowledge. Other two respondents also said that it can be quite tricky to open the mind of students for SPLs and make them interested.

3.7 How to improve the state of teaching SPL

We also asked people *'What is missing to make SPL mainstream, what would you suggest to improve the state of teaching SPLs?'.* From respondents' answers (29/34 provided at least a small comment) we extracted and categorized the following suggestions:

Better tools for students. Seven respondents said that teaching SPLs could be improved with 'better tools', i.e., mature, stable, and 'industry-ready' tools that students can use without days of training. Respondents also seem to be in favor of having a standardized tool recognized by a large community based on a common technological basis, e.g., regarding variability mechanisms and derivation technologies. They doubt the usefulness of research prototypes for teaching SPLs. Research efforts along the lines of the Common Variability Language (CVL) [2] could be candidates to fill this gap.

Improved textbook examples and case studies. 13 respondents see the need to improve (textbook) examples and

(industrial) case studies to support teaching SPL. They identify a key issue here, i.e., that the material – be it textbook examples or case studies – is simply not designed specifically for students. Good examples and case studies should be a showcase for how and why SPLs are introduced in industry and not make the subject look like a pure research endeavor. Also – as commented by one respondent – we must work on joining theory from the books with the tools used in practice, i.e., better map theoretical concepts and practical solutions.

Broaden the focus of teaching SPLs. Three respondents also suggested to broaden the focus of teaching SPLs, i.e., to somehow 'grow out of the software engineering realm' and also teach and use similar approaches like component-based systems and/or service-oriented architectures to motivate SPLs. A particular suggestion was to move away from feature models as putting the focus only on these models hinders progress and instead to encourage the use of domain-specific languages. In summary, respondents said that feature models should be part of teaching SPL but it is also important to show other approaches.

Is SPL already mainstream? Two respondents said that in their experience SPL is already mainstream in industry and this information should be conveyed to students. However, industry does not make use of very special techniques and approaches but often develop their own custom solutions [28] to support reuse and variability management based on whatever works for them. This means, when teaching SPLs the focus should not be put on teaching too 'exotic' SPL approaches. On the other hand, these approaches might some day become mainstream. Respondents commented that a balance between industry-strength methods/tools and emerging research ideas should be found by instructors.

Other ideas. An original idea by one respondent was to collect videos of experts that can be shown to students to motivate SPLs and explain key concepts and ideas. Another idea was to incorporate the notions and terminology of SPLs more in other courses, which is something scholars who teach SPLs presumably already do. Developing a standard curriculum and evaluation scheme for teaching SPLs, as suggested by one respondent, would also be a very important achievement towards making SPLs really mainstream.

3.8 Impact of teaching on research

Another open question we asked was '*How can teaching improve/impact research on SPLs?*'. We received comments by 29 of 34 respondents, which we analyzed and grouped as follows.

Student participation in research evaluations. 11 respondents commented that they find students useful participants in evaluating their research, either in experiments or in case studies. However, as commented by two other respondents, this can be a tricky issue because most likely only small, (quasi-)experiments can typically be done (depending on factors such as class size, course duration, experience level of students), which in turn do not provide a lot of empirical evidence. In the end, however, performing experiments with students is a good experience for both students and scholars. Even if the results might not be very useful or directly usable when writing a research paper, they can provide initial evidence and direct further studies; in other words, students can be very useful in performing pilot studies.

Feedback on and discussion about tools, examples, and case studies. Six respondents also commented that students are a very useful audience when it comes to simply discussing SPL research results and developed tools. Students can provide researchers with useful feedback on improving their work (even if only improving the presentation of their work). Also, students can help in tool development as well as elaborating and trying out examples and case studies that could be used later as teaching materials.

Finding research personnel. Eight respondents commented that they find teaching SPLs useful in finding personnel, being it research students or candidates for master and PhD theses. Also, the general benefit of attracting students to the area must not be underestimated, whether students end up in research or industry.

Connecting with industry. Five respondents also commented on the fact that some students in SPL courses either are already working in industry or will be working in industry soon. This allows connections with industry early on, i.e., to trigger future industry-academia collaborations or simply to increase awareness in industry.

Discussion of open research issues with students. Two respondents said they like to discuss research issues and challenges with students because getting a different view is always helpful to drive research, even if its 'only' from a student. Another comment was that missing teaching material is also sometimes an indicator for missing research (i.e. research venues that should be pursued).

3.9 Impact of teaching on industrial practice

Another open question in our survey asked '*How can teaching SPL impact industrial practice of SPL?*'. Here, 30 of 34 respondents provided us with comments, all similar or overlapping. Thus, we do not categorize comments.

Over a fifth of respondents commented that teaching SPLs makes students aware of the topic and students eventually end up in industry or even are already working in industry. Thus, teaching SPLs increases awareness of the field in industry and trains future practitioners. It can even open the door to companies, which have not been previously exposed to SPLs. 13 respondents went one step further and commented that teaching SPLs is already the preparation of the introduction of SPL engineering in industry, i.e., students will be the future drivers of the adoption of SPL engineering in industry. Some students already work in industry while they are being taught and thus might start triggering a paradigm shift almost right away. Furthermore, as commented by two respondents, teaching SPLs can be the start of industry-academia collaboration projects.

3.10 Other comments

Finally, we asked an open question for any other comments and concerns for teaching SPLs. We received comments from less than a third of respondents. We only discuss those not already covered in this section.

One respondent commented that they teach SPLs together with model-driven development and find the interplay of both approaches very interesting from a teaching perspective. Another respondent commented that *having a common, clearly defined basis of terminology and concepts that is taught at the majority of the institutions would help a lot.* Three respondents made a very similar comment, i.e., that real-world SPL engineering is very different from research in many aspects. For instance, while variability models seem to be *the* key topic in research, in practice it often is just one

technique for some roles. Industry often manages variability without dedicated models and still are successful with their product lines. This should be reflected in teaching. Finally, we received one comment that our survey was not a perfect fit for SPL consultants and trainers in industry. This is true and we have to address this in our future work. Here, we focused more on scholars teaching SPLs.

4. PESPECTIVES

In this section we build upon qualitative and quantitative results of previous sections and put forward six concrete actions to address some of the issues raised in the survey:

1. *Tool and artifact recognition.* Even though there are many tool options (see Section 3.3), the responses regarding challenges of teaching SPLs (cf. Section 3.6) and improvement of SPL teaching (cf. Section 3.7) point out the need for adequate SPL tools. We propose that conference or workshop events give recognition – maybe even an award – to artifacts, tools, as well as papers, that are made available and can be used for replicated studies and as teaching material. This will address the need of more robust and better documented SPL tools. The artifact recognition awards have become a staple of the ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE) in recent years.

2. *Create a virtual meeting place for the community.* Well-documented examples and case studies suitable for teaching are missing (see Section 3.6 and Section 3.7). Most available examples and case studies have not been developed and described for the purpose of teaching SPLs, but rather to report interesting findings in the research community. Available material should be restructured, prepared and organized for teaching. We propose to set up a community wiki (possibly independent of editorials, books, universities) where teaching materials and tools can be collected and shared. Additionally, dedicated web communities in social networks such as LinkedIn, ResearchGate, or Mendeley can be created. Examples of such virtual communities are the Repository for Model Driven Development (ReMoDD) [4], the Center of Excellence for Software Traceability [1], and the initiative of the Empirical Research in Software Engineering community behind the organization of the International Conference on Predictive Models in Software Engineering (PROMISE) who promotes and encourages replication of experiments and studies by sharing all data and software artifacts [30].

3. *Strive for teaching benchmarks.* Some of the respondents state that issues faced by the industry are the key to motivating the SPL course (see Section 3.9). Like other communities, we propose to create classroom-ready case studies that cover realistic scenarios faced in industry, which are engaging, realistic, and challenging for the students.

4. *Involve industry in teaching.* As mentioned by several respondents, the key benefits of SPLs can only be demonstrated using realistic examples (see Section 3.6). Another proposition is to invite industry experts to give talks within SPL lectures. It will hopefully increase students interest and improve the overall teaching experience.

5. *Create a teaching track at conference venues.* Our survey aims at providing a first overview of the state of practice of SPL teaching. Detailed experience reports and more advanced discussions would be beneficial and complement our effort. If there is enough interest and momentum, we plan to propose teaching tracks or dedicated workshops for SPL teaching at SPLC or related conferences/workshops. A role model for this point is the Educator's Symposium that takes place at the International Conference on Model Driven Engineering Languages and Systems (MODELS) [3].

6. *Develop a baseline curriculum and evaluation scheme.* It seems unrealistic to create one standard curriculum for teaching SPLs worldwide. For instance, no respondent reported on a full curriculum based on SPL engineering. Yet, having a baseline for creating curricula and performing student evaluations would still be helpful for the community. An inspiring example is a curriculum proposal on Data Mining put forward by the respective research community [13].

5. THREATS TO VALIDITY

Like any empirical study, our survey exhibits a number of threats to validity.

First of all, the survey was designed by just three researchers, based on their experience in research and teaching SPLs. To address this threat, we asked colleagues for feedback and refined the survey to incorporate their feedback before sending it out. Nevertheless, other researchers might have asked differend questions and, particularly, might have provided different selectable answers for certain questions.

A further threat to the internal validity of our results is that participants misunderstood questions. We mitigated this threat by test-driving the questionnaire with a range of colleagues with experience in teaching SPLs during its development. Also, the open question at the end of the survey allowed participants to raise concerns and ask for clarification. Two participants even contacted us directly before completing the survey with some clarification questions. We discussed among authors what to respond to them to not bias them with our response.

An external threat is that, perhaps, we did not contact a representative pool of SPL educators. In addition to selecting teachers we know, we also performed a web search for SPL courses and we reviewed the list of authors of SPLC and VaMoS papers to mitigate this threat. As commented by one respondent, the survey was not a perfect fit for SPL consultants and trainers in industry, which is something we have to address in our future work. To mitigate this threat we compared responses by industry people with responses by academics to check whether the answers are too divergent (they were not).

Our possibly wrong interpretation of the answers to open questions is a further important threat. We are confident, however, that we could capture the essence of these answers in our interpretations, especially, as for most answers we could find more than one similar instance.

6. CONCLUSIONS AND FUTURE WORK

We presented the results of a survey we conducted to assess the state of practice of teaching SPLs. 34 responses allowed us to report quantitative as well as qualitative results, i.e., about the context of teaching SPLs (country, institution, department, experience of the instructors, targeted audience, length of the course, primary literature used, tools used, and parts of the SPL development lifecycle covered) as well as about the experience of SPL teaching (challenges of teaching SPLs, suggestions to improve the state of teaching SPLs, impact on SPL research and industrial practice).

We sketched six conrete actions to continue improving teaching SPLs, i.e., tool and artifact recognition, creating a virtual meeting place for the community, striving for teaching benchmarks, involving industry in teaching, creating a teaching track at conference venues, and developing a baseline curriculum and evaluation scheme. Our next steps will be to evaluate the interest within the SPL community to participate in these actions as well as to collect additional suggestions for further steps.

Acknowledgements. First and foremost we would like to thank the participants of the survey and the colleagues that provided us with valuable feedback on earlier drafts. This research is partially funded by the Austrian Science Fund (FWF) Lise Meitner Fellowship M1421-N15. The research is also partially supported by Siemens VAI Metals Technologies and the Christian Doppler Forschungsgesellschaft, Austria. Acher's work was partially supported by the French BGLE Project CONNEXION.

7. REFERENCES

[1] Center of excellence for software traceability. `http://www.coest.org/`.

[2] Common variability language (cvl). `http://www.omgwiki.org/variability/doku.php?id=start`.

[3] International conference on model driven engineering languages and systems. `http://modelsconference.org/`.

[4] Repository for model driven development (remodd). `http://www.cs.colostate.edu/remodd/v1/`.

[5] E. K. Abbasi, A. Hubaux, M. Acher, Q. Boucher, and P. Heymans. The anatomy of a sales configurator: An empirical study of 111 cases. In *CAiSE'13*, volume 7908 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2013.

[6] V. Alves, N. Niu, C. Alves, and G. Valenca. Requirements engineering for software product lines: A systematic literature review. *Information and Software Technology*, 52(8):806–820, 2010.

[7] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines*. Springer, 2013.

[8] D. Benavides, S. Segura, and A. R. Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, 2010.

[9] N. Bencomo, S. O. Hallsteinsen, and E. S. de Almeida. A view of the dynamic software product line landscape. *IEEE Computer*, 45(10):36–41, 2012.

[10] T. Berger, R. Rublack, D. Nair, J. Atlee, M. Becker, K. Czarnecki, and A. Wasowski. A survey of variability modeling in industrial practice. In *Seventh International Workshop on Variability Modelling for Software-Intensive Systems (VaMoSŠ'13)*, pages 7–14, Pisa, Italy, 2013. ACM.

[11] T. Berger, S. She, R. Lotufo, A. Wasowski, and K. Czarnecki. A study of variability models and languages in the systems software domain. *IEEE Transactions on Software Engineering*, 99(PrePrints):1, 2013.

[12] J. Bosch and P. Bosch-Sijtsema. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67–76, 2010.

[13] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro, and W. Wang. Data mining curriculum: A proposal (version 1.0), April 2006.

[14] L. Chen and M. Ali Babar. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, 53(4):344–362, 2011.

[15] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering, Addison-Wesley, 2001.

[16] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison-Wesley, 2000.

[17] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski. Cool features and tough decisions: A comparison of variability modeling approaches. In *6th International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS 2012)*, pages 173–182, Leipzig, Germany, 2012. ACM.

[18] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira. A systematic mapping study of software product lines testing. *Information and Software Technology*, 53(5):407–423, 2011.

[19] Y. Dubinsky, J. Rubin, T. Berger, S. Duszynski, M. Becker, and K. Czarnecki. An exploratory study of cloning in industrial software product lines. *2011 15th European Conference on Software Maintenance and Reengineering*, 0:25–34, 2013.

[20] E. Engström and P. Runeson. Software product line testing – a systematic mapping study. *Information and Software Technology*, 53(1):2–13, 2011.

[21] C. Ghezzi and D. Mandrioli. The challenges of software engineering education. In *Software Engineering Education in the Modern Age*, pages 115–127. Springer, 2006.

[22] H. Gomaa. *Designing Software Product Lines with UML*. Addison-Wesley, 2005.

[23] M. Hinchey, S. Park, and K. Schmid. Building dynamic software product lines. *Computer*, 45(10):22–26, 2012.

[24] G. Holl, P. Grünbacher, and R. Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Information and Software Technology*, 54(8):828–852, 2012.

[25] A. Hubaux, A. Classen, M. Mendonça, and P. Heymans. A preliminary review on the application of feature diagrams in practice. In *Proceedings of the 4th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10)*, pages

53–59, Linz, Austria, 2010. Universität Duisburg-Essen.

[26] M. F. Johansen, Ø. Haugen, and F. Fleurey. A survey of empirics of strategies for software product line testing. In *Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, ICSTW '11*, pages 266–269. IEEE, 2011.

[27] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Technical Report CMU/SEI-90TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 1990.

[28] D. Lettner, M. Petruzelka, R. Rabiser, F. Angerer, H. Prähofer, and P. Grünbacher. Custom-developed vs. model-based configuration tools: Experiences from an industrial automation ecosystem. In *MAPLE/SCALE 2013, Workshop at the 17th International Software Product Line Conference (SPLC 2013)*, pages 52–58, Tokyo, Japan, 2013. ACM.

[29] J. Liebig, S. Apel, C. Lengauer, C. Kästner, and M. Schulze. An analysis of the variability in forty preprocessor-based software product lines. In *ICSE'10*, pages 105–114, 2010.

[30] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data. `http://promisedata.googlecode.com`, June 2012.

[31] B. Morin, O. Barais, J.-M. Jézéquel, F. Fleurey, and A. Solberg. Models@ run.time to support dynamic adaptation. *IEEE Computer*, 42(10):44–51, 2009.

[32] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

[33] R. Rabiser, P. Grünbacher, and D. Dhungana. Requirements for product derivation support: Results from a systematic literature review and an expert survey. *Information and Software Technology*, 52(3):324–346, 2010.

[34] F. van der Linden, K. Schmid, and E. Rommes. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg, 2007.

[35] M. Vierhauser, R. Rabiser, P. Grünbacher, C. Danner, and S. Wallner. Evolving systems of systems: Industrial challenges and research perspectives. In *1st International Workshop on Software Engineering of Systems-of-Systems (SESoS), in conjunction with the 27th ECOOP edition, the 9th ECMFA edition and the 7th ECSA edition (ECMFA, ECOOP and ECSA 2013)*, pages 1–4, Montepellier, France, 2013. ACM.

[36] D. Weiss and C. Lai. *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison Wesley Professional, 1999.