Workshop on Open and Original Problems in Software Language Engineering

Anya Helene Bagge, anya@ii.uib.no, http://www.ii.uib.no/~anya
Bergen Language Design Laboratory (BLDL), University of Bergen, Norway

Vadim Zaytsev, vadim@grammarware.net (main contact), http://grammarware.net,
Software Analysis & Transformation Team, Centrum Wiskunde & Informatica, The Netherlands

***Abstract*—The OOPSLE workshop is a discussion-oriented and collaborative forum for formulating and addressing with open, unsolved and unsolvable problems in software language engineering (SLE), which is a research domain of systematic, disciplined and measurable approaches of development, evolution and maintenance of artificial languages used in software development. OOPSLE aims to serve as a think tank in selecting candidates for the open problem list, as well as other kinds of unconventional questions and definitions that do not necessarily have clear answers or solutions, thus facilitating the exposure of dark data. We also plan to formulate promising language-related challenges to organise in the future. http://oopsle.github.io**

I. Summary

"Software languages" comprise all kinds of artificial languages used in software development: for programming, markup, pretty-printing, modelling, data description, formal specification, evolution, etc. Software language engineering is a relatively new research domain of systematic, disciplined and measurable approaches of development, evolution and maintenance of such languages. Many concerns of software language engineering are acknowledged by reverse engineers as well as by forward software engineers: robust parsing of language cocktails, fact extraction from heterogeneous codebases, tool interfaces and interoperability, renovation of legacy systems, static and dynamic code analysis, language feature usage analysis, mining repositories and chrestomathies, library versioning and wrapping, etc.

Some research fields have a list of acknowledged open problems that are being slowly addressed by the community: as examples of such lists, we can recall the Hilbert's problems [5], the POPLmark Challenge [10] and a list of open problems in Boolean grammars [9]. However, the field of software language engineering has not yet produced one. This workshop is meant to expose hidden expertise in coping with unsolvable or unsolved problems which commonly remain unexposed in academic publications. OOPSLE aims to serve as a think tank in selecting candidates for the open problem list, as well as other kinds of unconventional questions and definitions that do not necessarily have clear answers or solutions, thus facilitating the exposure of dark data [4]. We also plan to formulate promising language-related challenges to organise in the future. Beside the abovementioned POPLmark Challenge which can also be seen as a collection of benchmarks, there have been many more contests, challenges and competitions related to software language engineering: LDTA Tool Challenge held at the LDTA workshop in 2011 [7], CodeGeneration-affiliated Language Workbench Challenge [3] held yearly since 2011, Transformation Tool Contest held six times since 2007 [11], Rewrite Engines Competition held three times in 2006, 2008 and 2010 at WRLA [2], PLT Games held monthly since December 2012 [8].

II. Topics

We acknowledge the following list as non-exhaustive collection of examples of topics of interests of the workshop:

- Defining an unsolved problem by establishing both its provenance in prior research and the lack of a fully satisfactory solution.
- Identifying new problem areas in software language engineering that have not been previously studied due to lack of understanding, techniques, practical interest or scalability issues.
- Engaging in technological space travel by identifying similar problems in various sectors of software language engineering (i.e., grammarware, modelware, ontoware, XMLware, databases, spreadsheets, etc).
- Generalising and reformulating of several well-known problems into several sides of one open challenge (e.g., parsing and pretty-printing).
- Proposing systematic methods of assessment and comparison of existing and emerging solutions to a problem that is not or cannot be fully solved (e.g., choosing between parsing techniques, metaprogramming methodologies, software language workbenches).
- Arguing about definitions of terms commonly used in various senses (e.g., software language design, quality of a grammar, transformation).
- Presenting unconventional crossovers of popular research topics and software language engineering concerns (e.g., green IDEs and energy consumption considerations for parsing algorithms).
- Making an overview of major hindrances hindering solution of a standing problem (e.g., tool interoperability and reuse, tackling language and metalanguage diversity and versatility, consistency management).

- Designing open datasets in the software language engineering domain and the way we could share and incorporate them.
- Describing novel or unconventional ideas that are promising but are not yet fully validated, or where validation itself may be a challenge.
- Revealing solid negative results, failed experiments and disproven hypotheses.
- Constructing future community experiments and considering topics our community expects to see addressed by such a competition, if one decides to run it in the future.
- Critically reassessing a problem that is widely assumed to be solved but the solution is either underwhelming or could be "considered harmful" (model-driven engineering, domain-specific languages, test-based development).

III. Workshop Process

A. Before the Workshop

OOPSLE participants are encouraged to submit position papers up to 4 pages in length, sketching an open or original problem, idea or challenge. The submissions are screened by the workshop chairs, who will select papers based on potential for discussion and interest to the community, as well as the clarity of presentation and motivation — *OOPSLE is not a mini-conference*, and therefore it is not necessary for the work to be conclusive yet. The papers will be posted online prior to the workshop, so the participants have the opportunity to read them in advance.

B. At the Workshop

A keynote lecture titled *An Annotated and Illustrated Bibliography on Software Language Engineering*[1] will be given by Prof. Dr. Ralf Lämmel, one of the co-founders of the SLE conference, and is expected to set the context for the workshop by looking back at the history of the field and its open and closed challenges.

Each accepted paper is presented at the workshop as a brief summary of its main idea and a set of open questions to be discussed with the audience. Presenters will ask for input on how to proceed with experiments, validation or refinement of their ideas, collect opinions on the presented definitions, share similar experience. We expect the participants to be friendly but inquisitive, and ask hard questions back that may lead to deepening the initially presented insights. The workshop is planned to have short presentations and long discussions to stimulate direct collaboration afterwards.

C. After the Workshop

All workshop participants will be invited to submit a full paper to a special issue of the Electronic Communications of the EASST, an open access peer-reviewed journal (negotiations underway). Journal submissions will undergo peer review by the members of the program committee consisting of researchers in software language engineering and reverse engineering.

[1]Ralf Lämmel, An Annotated and Illustrated Bibliography on Software Language Engineering, blog post, 3 September 2013. http://professor-fish.blogspot.de/2013/09/an-annotated-and-illustrated.html.

IV. Examples

Cossette and Walker [1] investigate API migration and conclude that quite often none of known recommender techniques provide any useful advice, and even when they do, the recommendations are correct in only about 20 % of cases.

In 2011, a Tool Challenge was proposed at the LDTA workshop [7], for which participants needed to implement a range of features of the Oberon-0 [12] programming language. Submissions to the challenge were very versatile and gave insights into how the technologies behind them were related to one another.

Klint et al. [6, §7] list 15 research challenges for the domain of grammarware such as having a framework for grammar transformations, comprehensive grammarware testing, modular grammarware development, etc. Since 2005, most of them have been to some extent addressed by papers and tutorials of venues such as MODELS, SLE, WCRE, ICSM, CSMR, etc.

References

[1] B. E. Cossette and R. J. Walker, "Seeking the ground truth: a retroactive study on the evolution and migration of software libraries," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE '12. ACM, 2012, pp. 55:1–55:11.

[2] F. Durán, M. Roldán, J.-C. Bach, E. Balland, M. van den Brand, J. R. Cordy, S. Eker, L. Engelen, M. de Jonge, and K. T. Kalleberg, "The Third Rewrite Engines Competition," in *Eighth International Workshop on Rewriting Logic and Its Applications (WRLA)*, ser. LNCS, P. C. Ölveczky, Ed., vol. 6381. Springer, 2010, pp. 243–261.

[3] S. Erdweg, T. van der Storm, M. Völter, M. Boersma, R. Bosman, W. R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, G. Konat, P. J. Molina, M. Palatnik, R. Pohjonen, E. Schindler, K. Schindler, R. Solmi, V. Vergu, E. Visser, K. van der Vlist, G. Wachsmuth, and J. van der Woning, "The State of the Art in Language Workbenches. Conclusions from the Language Workbench Challenge," in *Proceedings of the Sixth International Conference on Software Language Engineering (SLE 2013)*, ser. LNCS, M. Erwig, R. F. Paige, and E. Van Wyk, Eds., vol. 8225. Springer, Oct. 2013, in print.

[4] T. Goetz, "Freeing the Dark Data of Failed Scientific Experiments," *Wired Magazine*, vol. 15, no. 10, 2007.

[5] D. Hilbert, "Mathematical Problems," *Bulletin of the American Mathematical Society*, vol. 33, no. 4, pp. 433–479, 1902.

[6] P. Klint, R. Lämmel, and C. Verhoef, "Toward an Engineering Discipline for Grammarware," *ACM Transactions on Software Engineering Methodology (TOSEM)*, vol. 14, no. 3, pp. 331–380, 2005.

[7] LDTA 2011, "11th International Workshop on Language Descriptions, Tools and Applications. Tool Challenge," 2011. [Online]. Available: http://ldta.info/tool.html

[8] B. McKenna, "The Programming Language Theory Games: a monthly programming language competition," Dec. 2012. [Online]. Available: http://www.pltgames.com

[9] A. Okhotin, "Conjunctive and Boolean Grammars: The True General Case of the Context-Free Grammars," *Computer Science Review*, vol. 9, pp. 27–59, 2013. [Online]. Available: http://users.utu.fi/aleokh/boolean/nine_open_problems.html

[10] B. C. Pierce, P. Sewell, S. Weirich, and S. Zdancewic, "It Is Time to Mechanize Programming Language Metatheory," in *Verified Software: Theories, Tools, Experiments*, ser. LNCS, B. Meyer and J. Woodcock, Eds. Springer Berlin Heidelberg, 2008, vol. 4171, pp. 26–30.

[11] A. Rensink and P. Van Gorp, "Graph Transformation Tool Contest 2008," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 12, no. 3–4, pp. 171–181, 2010.

[12] N. Wirth, *Compiler Construction*, ser. International computer science series. Addison-Wesley, 1996.