S.P.L.O.T. – Software product lines online tools

Conference Paper · October 2009

DOI: 10.1145/1639950.1640002 · Source: DBLP

3 authors:

Marcílio Mendonça
Amazon.com Inc.
19 PUBLICATIONS **701** CITATIONS
SEE PROFILE

Moises Branco
University of Waterloo
7 PUBLICATIONS **293** CITATIONS
SEE PROFILE

Donald D Cowan
University of Waterloo
290 PUBLICATIONS **2,333** CITATIONS
SEE PROFILE

S.P.L.O.T. - Software Product Lines Online Tools

Marcilio Mendonca
University of Waterloo, Canada
marcilio@csg.uwaterloo.ca

Moises Branco
Banco do Nordeste do Brasil, Brasil
moises@bnb.gov.br

Donald Cowan
University of Waterloo, Canada
dcowan@csg.uwaterloo.ca

Abstract

This paper introduces S.P.L.O.T., a Web-based reasoning and configuration system for Software Product Lines (SPLs). The system benefits from mature logic-based reasoning techniques such as SAT solvers and binary decision diagrams to provide efficient reasoning and interactive configuration services to SPL researchers and practitioners. In addition, the system provides a feature model repository containing real and generated models to encourage knowledge sharing among researchers in the field.

Categories and Subject Descriptors D.2.2 [*Software Engineering*]: Design Tools and Techniques—CASE; D.2.1 [*Software Engineering*]: Requirements/Specifications—Tools

General Terms Design

Keywords Software Product Lines, Feature Models, Automated Reasoning, Interactive Configuration, Feature Model Analysis

1. Introduction

Software Product Lines (SPLs) [6, 7] is a contemporary approach to software development that promotes "reuse" to a first-class citizen aiming at reducing overall development time and cost while improving product quality. *Feature models* [9] play an important role in SPLs by providing means to represent the commonalities and variabilities within a family of systems which thereby allows individual family members to be safely configured. This year, the SPL community celebrates the 20^{th} anniversary of feature models as a recognition of its relevance to the field. Currently, several feature-based reasoning [14, 11] and configuration [1, 10, 5] tools are available taking advantage of mature logic-based systems such as SAT solvers [12], binary decision diagrams (BDDs) [13] and constraint solvers [3] among others.

OOPSLA 2009, October 25–29, 2009, Orlando, Florida, USA.
ACM 978-1-60558-768-4/09/10.

In this paper, we introduce a new reasoning and configuration system for feature models called S.P.L.O.T.[1] (available at www.splot-research.org). S.P.L.O.T. offers a dynamic Web-based interface backed up by a rich set of state-of-the-art algorithms for configuring and reasoning on Software Product Lines by means of feature models. To the best of our knowledge, S.P.L.O.T. is the first internet-based system to deliver such a variety of services for SPL end-users and researchers. Since its launch in May 2009, S.P.L.O.T. has been visited by several research groups from over 15 countries.

In the next sections, we introduce S.P.L.O.T., its reasoning infra-structure, provided services and the repository of feature models.

2. S.P.L.O.T.

2.1 Overview

S.P.L.O.T. is a Web-based system developed in Java[2] that uses an HTML template engine to build highly-interactive Ajax-based reasoning and configuration user interfaces. As the system is internet-based it strongly facilitates knowledge sharing (e.g. feature model repository) and does not require downloading software updates. S.P.L.O.T. is supported by sophisticated configuration engines and efficient automated reasoning systems based on public SAT (SAT4J [4]) and BDD (JavaBDD [15]) engines. In addition, S.P.L.O.T. offers a repository of feature models as discussed later.

2.2 Reasoning Techniques

Research exploring the connection between feature models and propositional logic [2] has enabled the use of logic-based techniques such as BDDs and SAT solvers to reason on feature models. However, it is well-known that those techniques can suffer from space (BDD) and/or time (SAT) intractability problems. To minimize such problems, S.P.L.O.T. makes use of novel BDD heuristics [13] to shrink the size of BDDs as much as possible. As well, the system capitalizes on the observed efficiency of SAT systems in the feature modeling domain [12] to provide high-performance SAT-based algorithms (e.g. valid domain computation).

[1] S.P.L.O.T. stands for *S*oftware *P*roduct *L*ines *O*nline *T*ools

[2] Java Servlet API 2.5

Specifically, S.P.L.O.T. uses a BDD engine to count valid configurations, to calculate the variability degree of feature models and to perform interactive configuration. Moreover, a SAT solver is also used to support interactive configuration and to perform debugging tasks such as checking the consistency of feature models and detecting common and dead features. We expect new reasoning techniques and operations to be supported in the near future.

2.3 Services

Currently, S.P.L.O.T. provides two major services: *automated reasoning* and *product configuration*. Reasoning focus on automating statistics computation (e.g. depth of the feature tree, number of features) and critical debugging tasks such as checking the consistency of feature models and detecting the presence of dead and common features. In addition, reasoning supports measuring properties such as the number of valid configurations and the variability degree of feature models. With regards to product configuration, S.P.L.O.T. currently supports *interactive configuration* [8] in which users make a decision at a time and the configuration system automatically propagates those decisions to enforce their consistency. This results in a *backtrack-free* configuration process directly benefiting users that are never forced to revisit past decisions. Among current operations supported are *configuring*, *undoing*, *toggling* (including a conflict resolution strategy), and *auto-completing*. S.P.L.O.T. achieves high performance in interactive configuration by using BDDs and a novel SAT-based valid domain computation algorithm[3] that minimizes the number of SAT checks performed.

2.4 Feature Model Repository

A major issue reported by SPL researchers is the lack of publicly-available feature models. S.P.L.O.T. tackles this issue by providing a public model repository containing over 20 real models previously published in the literature and several automatically-generated models with up to 10,000 features. We strongly encourage researchers to submit their own models to the repository.

3. Conclusion

We presented S.P.L.O.T., a Web-based reasoning and configuration system for Software Product Lines. The system benefits from logic-based reasoning techniques such as SAT solvers and BDDs to provide efficient reasoning and interactive configuration services to SPL researchers and practitioners. In addition, a feature model repository containing real and generated models is available to encourage knowledge sharing among researchers. Currently, S.P.L.O.T. is on experimental stage and we expect many new features to be added in the near future.

[3] An upcoming paper submission from the S.P.L.O.T. team

References

[1] M. Antkiewicz and K. Czarnecki. FeaturePlugin: Feature modeling plug-in for Eclipse. In *OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop*, 2004.

[2] D. S. Batory. Feature models, grammars, and propositional formulas. In *Software Product Lines, 9th Int. Conference, SPLC 2005, Rennes, France, September 26-29, 2005, Proceedings*, volume 3714 of *LNCS*, pages 7–20. Springer, 2005.

[3] D. Benavides, P. Trinidad, and A. Ruiz-Cortes. Automated reasoning on feature models. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, 2005*, LNCS. Springer, 2005.

[4] D. L. Berre, A. Parrain, O. Roussel, and L. Sais. *SAT4J: A satisfiability library for Java*, 2005. http://www.sat4j.org/.

[5] D. Beuche. pure::variants Eclipse Plugin. User Guide. pure-systems GmbH. Available from http://web.pure-systems.com/fileadmin/downloads/pv$_u ser guide.pdf$, 2004.

[6] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, MA, 2001.

[7] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Boston, MA, 2000.

[8] T. Hadzic, S. Subbarayan, R. M. Jensen, H. R. Andersen, J. Møller, and H. Hulgaard. Fast backtrack-free product configuration using a precompiled solution space representation. In *PETO Conference*, pages 131–138. DTU-tryk, June 2004.

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

[10] C. Kastner, T. Thum, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel. FeatureIDE: A tool framework for feature-oriented software development. In *ICSE '09*, pages 611–614, Washington, DC, USA, 2009.

[11] M. Mendonca. 4WATREASON - A Feature Model Reasoning Tool, 2008. Available at: http://csg.uwaterloo.ca/ marcilio/fmcompilation/index.html.

[12] M. Mendonca, A. Wasowski, and K. Czarnecki. SAT-based analysis of feature models is easy. In *Proceedings of the 13th Int. Software Product Line Conference (SPLC)*, 2009.

[13] M. Mendonca, A. Wasowski, K. Czarnecki, and D. D. Cowan. Efficient compilation techniques for large scale feature models. In *Int. Conference on Generative Programming and Component Engineering (GPCE'08)*, pages 13–22, 2008.

[14] P. Trinidad, D. Benavides, A. Ruiz-Cortes, S. Segura, and A. Jimenez. FAMA framework. *Software Product Line Conference, 2008. SPLC '08. 12th Int.*, pages 359–359, Sept. 2008.

[15] J. Whaley. The $JavaBDD$ BDD library, 2003–2007. Available at: http://javabdd.sourceforge.net/.