Benchmarking on the Automated Analyses of Feature Models: A Preliminary Roadmap *

Sergio Segura and Antonio Ruiz-Cortés
Department of Computer Languages and Systems
University of Seville
Av Reina Mercedes S/N, 41012 Sevilla, Spain
{sergiosegura, aruiz} AT us.es

Abstract

The automated analysis of Feature Models (FMs) is becoming a well-established discipline. New analysis operations, tools and techniques are rapidly proliferating in this context. However, the lack of standard mechanisms to evaluate and compare the performance of different solutions is starting to hinder the progress of this community. To address this situation, we propose the creation of a benchmark for the automated analyses of FMs. This benchmark would enable the objective and repeatable comparison of tools and techniques as well as promoting collaboration among the members of the discipline. Creating a benchmark requires a community to share a common view of the problem faced and come to agreement about a number of issues related to the design, distribution and usage of the benchmark. In this paper, we take a first step toward that direction. In particular, we first describe the main issues to be addressed for the successful development and maintenance of the benchmark. Then, we propose a preliminary research agenda setting milestones and clarifying the types of contributions expected from the community.

1. Motivation

The automated analysis of feature models consists on the computer–aided extraction of information from feature models. This extraction is performed by means of analysis operations. Typical operations of analysis allow finding out whether a feature model is void (i.e. it represents no products), whether it contains errors (e.g. feature that cannot be part of any products) or what is the number of products of the software product line represented by the model. A wide range of analysis operations and approaches to automate them have been reported [5, 7].

Recent workshops [6] and publications [8, 11, 19, 21, 25, 26] reflect an increasing concern to evaluate and compare the performance of different solutions in the context of automated analyses of feature models. However, the lack of standard problems to perform these empirical tests often difficult getting rigorous conclusions widely accepted by the community. Experiments in this context are mainly ad-hoc and not public and subsequently not repeatable by other researchers. Thus, performance conclusions are rarely rigorous and verifiable. As a result, these conclusions can barely be used to guide further research hindering the progress of the different solutions and, in general, of the whole discipline.

A *benchmark* is a test (a.k.a. test problem) or set of tests used to compare the performance of alternative tools or techniques [22]. Benchmarks have contributed to the progress of many disciplines along the years providing a level playing field for the objective and repeatable comparison of solutions. From a technical standpoint, the usage of benchmarks leads to a rigorous examination of performance results. From these results, the strengths and weaknesses of each proposal are highlighted helping researchers to improve their solutions and identify new research directions. From a social standpoint, benchmarks promote the collaboration and communication among different researchers. As a result, these become more aware of the work carried out by their colleagues and collaborations among researchers with similar interests emerge naturally.

Developing a benchmark for the automated analyses of feature models could contribute to the progress of the discipline, both at the technical and the social level. This was one of the main conclusions of the first workshop on Analysis of Software Product Lines (ASPL, [6]). There, a number of attendants agreed on the need for a benchmark (i.e. set

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472) and the Andalusian Government project ISABEL (TIC-2533)

of standard feature models) to evaluate our solutions in rigorous and widely accepted way.

Creating a benchmark requires a community to share a common view of the problem faced and come to agreement about a number of issues related to the design, distribution and usage of the benchmark. In this paper, we take a first step toward that direction. In particular, we first describe the main issues to be addressed for the successful development and maintenance of the benchmark. Then, we propose a preliminary research agenda setting milestones and clarifying the types of contributions expected from the community.

The remainder of this paper is structured as follows. In Section 2 we detail the open issues to be addressed for the successful introduction of a benchmark in the community of automated analyses of feature models. Section 3 presents a preliminary research agenda for the development and maintenance of the benchmark. Finally, we summarize our conclusions in Section 4.

2. Open issues

We identify a number of open issues to be addressed for the successful introduction of a benchmark in the community of automated analyses of feature models. Next, we describe them.

2.1. Are we ready for it?

Sim *et al.* [22] draw attention to two preconditions that should exist in order to be success when introducing a benchmark into a research community.

The first precondition requires a minimum level of maturity in the discipline. As evidence that this minimum level has been reached, a sufficient number of different proposals to be evaluated using the benchmarks should be already available. This would provide some guarantee of the future interest of the community in the benchmark. This is a relevant condition since the effort needed to introduce a benchmark into a discipline is significant [17, 22, 24]. The community should have a strong commitment to participate actively on its development and maintenance, e.g. proposing new test problems regularly.

The second precondition point out the need for an active collaboration among researchers. These should be well-disposed to work together to solve common problems. According to Sim, these collaborations help researchers to gain familiarity and experience creating a community more receptive to the results and consequently more likely to use the benchmark. Some evidences of the willingness to collaborate may be deduced from previous collaboration between the members of the community, e.g. multi-author publications.

A number of evidences suggest that these two conditions already exist in the domain of automated analyses of feature models. In the context of maturity, existing surveys [5, 7] reflect that a sufficient number of proposals to be evaluated using the benchmark are already available. Additionally, an increasing concern to evaluate and compare the performance of tools and techniques is detected in recent publications [8, 11, 19, 21, 25, 26]. In the context of collaboration, recent workshops [6] and multi-authors publications such as [4] or [25] (best paper award at SPLC'08) also suggest that the community is ready to incur in the development of a benchmark. Despite this, we consider that the introduction of a benchmark must be still further debated by the community in order to find out the level of interest and commitment of its members to participate on it.

2.2. Agreeing a format

Reaching a consensus on a language to specify test problems is a key point for a benchmark being accepted by the community. To this end, the semantic, abstract and concrete syntax of the language should be carefully studied. The semantic should be well defined to avoid ambiguity and redundancies in the specification of the problems. The abstract syntax should be flexible enough to enable the usage of the benchmark with tools and techniques using different notations. Finally, the concrete syntax should be as simple as possible to simplify its understanding and manipulation.

For the semantic and abstract syntax of the language, an overview of the available papers surveying feature modelling notations would be desirable. A good starting point could be the work of Schobbens *et al.* [20]. In their work, the authors survey a number of feature diagram notations and study some of their formal properties. As a result, they propose a new feature diagram language, VFDs (Varied Feature Diagrams), embedding all other variants.

For the concrete syntax, that is, the specific format used to represent and distribute the problems, we foresee two main options: plain text and XML. These appear to be the most popular input formats used in the existing feature model analyses tools. An example of tool using plain text is the Ahead Tool Suite[1] in which feature models are represented as grammars. Some examples of tools using XML are the Feature Model Plug-in[2], the FAMA framework[3] and Pure::Variants[4]. For the selection of one format or another, advantages and drawbacks of each option should be evaluated and debated. On the one hand, plain text formats tend to be shorter than XML documents and usually more suitable to be written by human beings. On the other hand,

[1] http://www.cs.utexas.edu/users/schwartz/ATS.html
[2] http://gp.uwaterloo.ca/fmp/
[3] http://www.isa.us.es/fama/
[4] http://www.pure-systems.com/

XML is a widely extended mechanism to exchange information easy to be defined (e.g. XML schema) and parsed.

A review of the formats used in related software benchmarks could also be helpful to support a decision. In a first outlook to these benchmarks we noticed that plain text seems to be the preferred format especially on those benchmarks related to mathematical problems. Some examples are the DIMACS CNF format [1] for satisfiability problems, the MPS format [2] for linear programming or the AMPLE format [10] for linear and nonlinear optimization problems. We also found some related benchmark dealing with XML format such as GXL [12], introduced in the context of reverse engineering, or XCSP [3] for constraint programming.

Finally, feature modelling community could also benefit from the lessons learned in other domains when selecting a format. We found in XCSP an interesting case of this. In the current version of the format (i.e. 2.1), released in January 2008 for the Third International CSP Solver Competition[5], the authors felt the need to distinguish between two variants of the format: a *'fully-tagged'* representation and a *'abridged'* one. According to the authors, the tagged notation is *'suitable for using generic XML tools but is more verbose and more tedious to write for a human being'* meanwhile the abridged notation *'is easier to read and write for a human being, but less suitable for generic XML tools'*. As a negative consequence of this, authors of XCSP must now provide up-to-date support for two different formats which include updating documentation, parsers, tools to convert from one representation to another, etc. Studying the synergies between XCSP and our future benchmark format could help us to predict whether we could find the same problem using XML. Reporting similar lessons learned in other domains would be highly desirable for supporting a decision.

2.3. Selection of test problems

The design of test problems is recognized as one of the most difficult and controversial steps during the development of a benchmark [22, 24]. Walter Tichy advises:

> *'The most subjective and therefore weakest part of a benchmark test is the benchmark's composition. Everything else, if properly documented, can be checked by the skeptic. Hence, benchmark composition is always hotly debated.*' [24] (page 36)

These test problems should be representative of the real problems to be solved by the tool or technique under test. As discussed in [9, 14], there are essentially three sources of test problems: those which arise in real scenarios, those that are specifically developed to exercise a particular aspect of the tool or technique under test and randomly generated ones. There is not a consensus about the criteria for the selection of one type or another [14]. In practice, researchers from different research disciplines usually adopt a pattern of use. There exist well-documented deficiencies of each alternative. In the case of specific collection of problems, Jackson *et al.* [14] summarizes them as follows:

- The test set is usually small compared whit the total set of potential test problems.
- The problems are commonly small and regular. There may exist large-scale problems but they are often not distributed because it is difficult and time-consuming.
- Problems may have similar properties. As a result of this, some of the features of the tool or technique could be not exercised.
- Optimizing a technique or tool for a set of test problems may not provide ideal performance in other settings.

Randomly generated problems overcome some of the drawbacks detailed previously but also attract other negative opinions. In particular, these critics focus on the lack of realism of those problems and the systematic structures that sometimes may appear on them.

Two main types of problems are reported in the context of feature models: invented and randomly generated ones. One the one hand, invented feature models are usually small and regular. They are used for research purposes but they rarely can be used to showcase the performance of a tool or technique. On the other hand, randomly generated ones are more adequate to check the performance of tools but they do not represent real problems and rarely can be replicated by other researchers. There exist references in the literature to software product lines with thousand of features [23] (page 32) but to the best of our knowledge associated feature models are not available. We presume this may due to the effort required to distribute them or to confidentiality issues.

Different types of contribution would be welcome by the community of automated analyses of feature models in the context of a benchmark. Firstly, feature models from real scenarios are highly desirable to both studying their properties and using them as motivating inputs for the tools and techniques under evaluation. Notice that these feature models could include not only feature models from industry but also feature models extracted from OS projects (e.g. [13, 16]). Secondly, collection of problems published in the literature would be helpful since they represent widely accepted problems by the community. Finally, random feature models will be needed to evaluate the performance of tools and techniques dealing with large-scale problems. Regardless the type of problem proposed, this should be clearly

[5] http://cpai.ucc.ie/

justified by stating what characteristics make it a good test problem and what it is hoped to learn as a result of running it.

2.4. Benchmark development

The key principle underlying the benchmark development is that it must be a community effort [22, 24]. Members of the discipline should participate actively in the development and maintenance of the benchmark through a number of tasks. Some of these are:

- Agreeing a format for the test problems.
- Design and publication of test problems.
- Usage of the benchmark and publication of results.
- Regular submission of new test problems.
- Report errors or possible improvements in the format or existing test problems.

Note that continued evolution of the benchmark is required to prevent users from optimizing their tools or techniques for a specific set of test problems.

Based on their experience, Sim *et al.* [22] attributes the success of a benchmark development process to three factors, namely:

- *'The effort must be lead by a small number of champions'*. This small group of people should be responsible of keeping the project alive and will be commonly in charge of organizing and coordinating activities to promote discussion among the members of the community.
- *'Design decisions for the benchmark need to be supported by laboratory work'*. Some experiments may be needed to show the effectiveness of a solution and to support decisions.
- *'The benchmark must be developed by consensus'*. To this end, it is necessary to promote the discussion of the members of the community in many formats as possible. Some options are workshops, conferences, mailing lists, discussion forums, Request for Comments (RFC), etc. In this context, Sim point at face-to-face meeting in conferences and workshops as the most effective method.

For the successful development of a benchmark, community should be aware of how they can contribute. To this end, detailed information about the different tasks to be carried out and the effort required for each of them would be highly desirable. This paper pretend to be a first contribution in that direction (see Section 3).

2.5. Support infrastructure

Successful benchmarks are commonly provided together with a set of tools and mechanism to support its usage and improvement. Some examples are mailing list to enable discussion among users, test problems generators, parsers, documentation, etc. These contributions are welcome at any time but they are especially appealing during the release of the benchmark in order to promote its usage within the community. Participating at this level may required an important effort from the community but it also may appear as good opportunity to come in contact with other researchers working in similar topics.

Contributions from the community of feature models in any of these forms (i.e. generators, parsers, etc.) will be greatly welcome.

2.6. Using the benchmark

Performing experiments and reporting its results is not a trivial task. The analysis, presentation and interpretation of these results should be rigorous in order to be widely accepted by the community. To assist in this process, a number of guidelines for reporting empirical results are available in the literature. Some good examples can be found in the areas of mathematical software [9, 14] and software engineering [15, 18].

At the analysis level, aspects such as the statistic mechanisms used, the treatment of outliers or the application of quality control procedures to verify the results should be carefully studied.

A number of considerations should also be taken into account when presenting results. As an example, Kitchenham *et al.* [18] suggest a number of general recommendations in the context of experiments in software engineering. These include providing appropriate descriptive statistics (e.g. present numerator and denominator for percentages) or making a good usage of graphics (e.g. avoid using pie charts).

Finally, the interpretation of results should also follow some well-defined criteria and address different aspects. These may include describing inferences drawn from the data to more general conditions and limitations of the study.

Contributions for the correct usage of the benchmark in the context of automated analyses of feature models would be helpful. These may include guidelines and recommendations about how to get (e.g. useful measures), analyse (e.g. adequate statistics packages), present (e.g. suitable graphs) and interpret (e.g. predictive models) the benchmark results.

3. Preliminary roadmap

Based on the open issues introduced in previous sections, we propose a preliminary roadmap for the development of a benchmark for the automated analyses of feature models. In particular, we first clarify the types of contributions expected from the community. Then, we propose a research agenda.

3.1. Types of contributors

The main goal of this section is to clarify the ways in which the community can contribute to the development and usage of the benchmark. To this end, we propose dividing up the members of the discipline interested in the benchmark into three groups according to their level of involvement in it, namely:

- **Users.** This group will be composed of the members of the discipline interested exclusively in the usage of the benchmark and the publication of performance results. Exceptionally, they will also inform about bugs in the test problems and tools related to the benchmark.
- **Developers.** This group will be composed of members of the community interested in collaborating in the development of the benchmark. In addition to the tasks expected from users, the contributions from this group include:
  - Designing and maintaining new test problems
  - Proposing guidelines and recommendations for an appropriate usage of the benchmark.
  - Developing tools and/or documentation, e.g. test problem generators.
- **Administrators.** These will be the *'champions'* in charge of most part of the work. This group will be composed of a few researchers from one or more laboratories. In addition to the tasks associated to the users and developers, the contributions expected from this group include:
  - Organizing and coordinating activities to promote discussion (e.g. performance competitions).
  - Proposing a format to be accepted by the community.
  - Publication of test problems.
  - Setting mechanisms to promote contributions from the community, e.g. template for submitting new test problems

3.2. Research agenda

We identify a number of tasks to be carried out for the development and maintenance of a successful benchmark for the automated analyses of feature models. Figure 1 depicts a simplified process model illustrating these tasks using BPMN[6] notation. Rectangles depict tasks (T-X) and diamond shapes represent decisions (D-X). For the sake of simplicity, we distinguish tree group of tasks: those carried out by the community (i.e. administrators, developers and users of the benchmark), those performed by the administrators and those tasks accomplished by both administrators and developers.

As a preliminary step, community of automated analyses of feature models should evaluate whether we are ready to incur in the development of a benchmark (*T-01*). As discussed in Section 2.1, we consider this discipline is mature enough and has the necessary culture of collaboration to start working on it. However, we still consider that a noticeable interest from the members of the discipline to participate either in the development or the usage of the benchmark should be detected. If this precondition is not met (*D-01*), it does not mean the benchmark cannot be used. Rather, it means that some actions should be then carried out to establish this precondition. In this context, Sim suggests waiting for more research results and planning activities to promote collaboration among researchers (*T-02*).

Once the community agrees on the need for a benchmark, the design of a format for the test problems should be the first step (*T-03*). This should be proposed by the administrators and approved by a substantial part of the community (*D-02*). It should be presented in a standard format such a technical report and include a versioning system to keep record of its evolution. Note that several attempts (i.e. versions) could be needed until reaching a wide consensus.

Once an accepted format is available, an initial set of test problems (*T-04*) and support tools (*T-05*) should be released by administrators. At the very least, we consider these should include a parser for the test problems and platform to publish the material related to the benchmark (e.g. FTP site). At this point, the benchmark would already be fully usable.

Finally, a number of contributions from the community for the maintenance and improvement of the benchmark would be expected. These include $i)$ Using the benchmark and publishing the results (*T-06*), $ii)$ Reporting bug and suggestions (*T-07*), $iii)$ Organizing activities to promote discussion among the members of the community (*T-08*), $iv)$ Proposing new test problems (*T-09*), $v)$ Developing tools and documentation (*T-10*), and $vi)$ Providing guidelines and recommendations for an appropriate usage of the benchmark (*T-11*).

[6] http://www.bpmn.org/

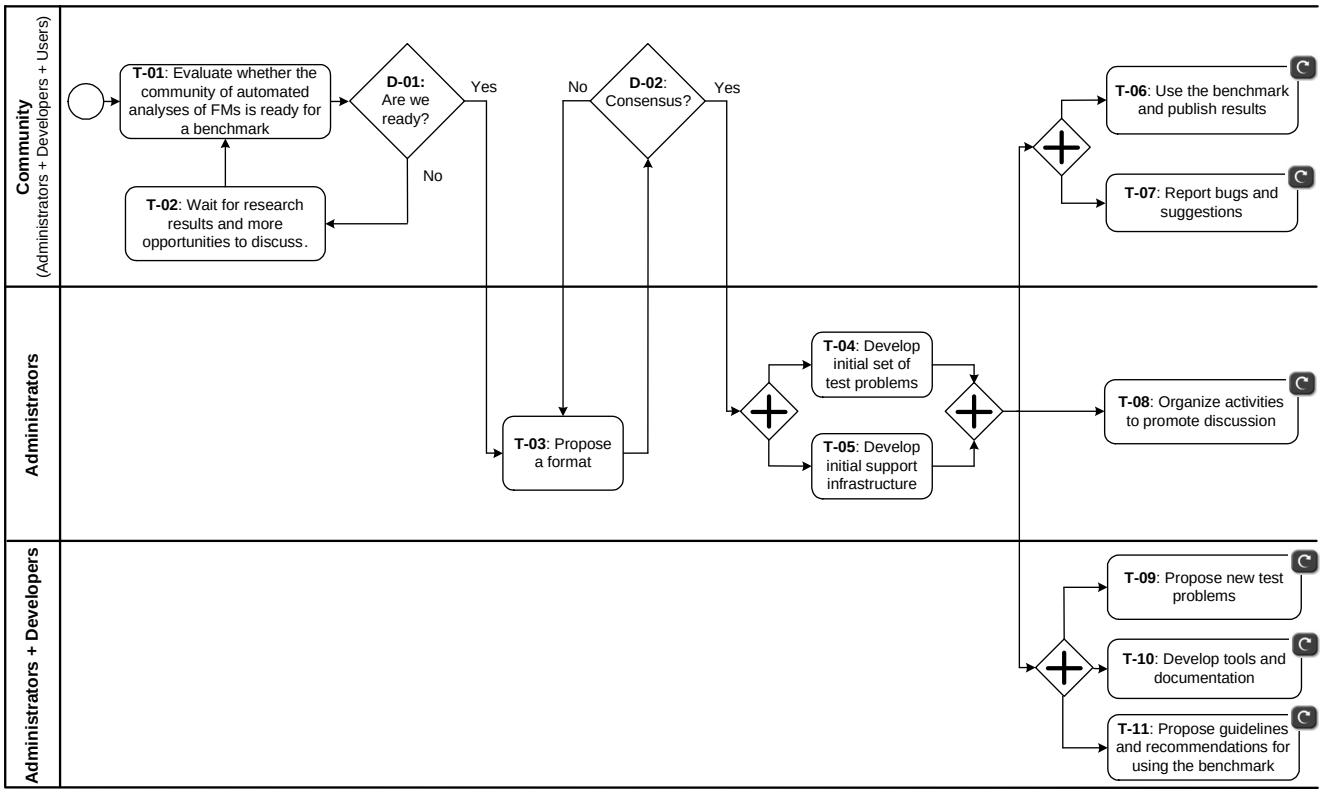

Figure 1. A process model of the proposed research agenda

4. Conclusions

The introduction of a benchmark for the automated analyses of feature models could contribute to the progress of the discipline by providing a set of standard mechanisms for the objective and repeatable comparison of solutions. A key principle underlying the creation a benchmark is that it must be a community effort developed by consensus. To this end, members of the discipline should first share a common view of the problem faced and the tasks to be carried out for its development. This is the main contribution of this paper. In particular, we first described the open issues to be addressed for the successful introduction of a benchmark for the automated analyses of feature models. Then, we proposed a preliminary roadmap clarifying the types of contributions expected from the community and the main steps to be taken. To the best of knowledge, this is the first contribution in the context of benchmarking on the automated analyses of feature models.

Acknowledgments

We would like to thank Dr. David Benavides whose useful comments and suggestions helped us to improve the paper substantially.

References

[1] DIMACS Conjunctive Normal Form format (CNF format). Online at http://www.satlib.org/Benchmarks/SAT/satformat.ps.

[2] Mathematical Programming System (MPS format). Online at http://lpsolve.sourceforge.net/5.5/mps-format.htm.

[3] XML Representation of Constraint Networks Format XCSP 2.1. Online at http://www.cril.univ-artois.fr/CPAI08/XCSP2_1.pdf.

[4] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: Challenges ahead. *Communications of the ACM*, December:45–47, 2006.

[5] D. Benavides. *On the Automated Analysis of Software Product Lines using Feature Models. A Framework for Developing Automated Tool Support.* PhD thesis, University of Seville, 2007.

[6] D. Benavides, A. Ruiz-Cortés, D. Batory, and P. Heymans. First International Workshop on Analyses of Software Product Lines (ASPL'08), September 2008. Limerick, Ireland.

[7] D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feture models. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, 2006.

[8] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. In *Managing Variability for Software Product Lines: Working With Variability Mechanisms*, 2006.

[9] H. Crowder, R.S. Dembo, and J.M. Mulvey. On reporting computational experiments with mathematical software. *ACM Transactions on Mathematical Software*, 5(2):193–203, 1979.

[10] R. Fourer, D.M. Gay, and B.W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[11] A. Hemakumar. Finding contradictions in feature models. In *First Workshop on Analyses of Software Product Lines (ASPL 2008). SPLC'08*, Limerick, Ireland, September 2008.

[12] R.C. Holt, A. Winter, and A. Schürr. GXL: Toward a Standard Exchange Format. In *WCRE '00: Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00)*, page 162, Washington, DC, USA, 2000. IEEE Computer Society.

[13] A. Hubaux, P. Heymans, and D. Benavides. Variability modelling challenges from the trenches of an open source product line re-engineering project. In *Proceedings of the Sofware Product Line Conference*, pages 55–64, 2008.

[14] R.H. Jackson, P.T. Boggs, S.G. Nash, and S. Powell. Guidelines for reporting results of computational experiments. report of the ad hoc committee. *Mathematical Programming*, 49(1):413–425, November 1990.

[15] A. Jedlitschka and D. Pfahl. Reporting guidelines for controlled experiments in software engineering. In *Empirical Software Engineering, 2005. 2005 International Symposium on*, pages 10 pp.+, 2005.

[16] C. Kastner, S. Apel, and D. Batory. A case study implementing features using aspectj. In *SPLC '07: Proceedings of the 11th International Software Product Line Conference*, pages 223–232, Washington, DC, USA, 2007. IEEE Computer Society.

[17] B.A. Kitchenham. Evaluating software engineering methods and tool. Part 1 to 12. *ACM SIGSOFT Software Engineering Notes*, 21-23(1), 1996-1998.

[18] B.A Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K.E. Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transaction on Software Engineering*, 28(8):721–734, August 2002.

[19] M. Mendonca, A. Wasowski, K. Czarnecki, and D. Cowan. Efficient compilation techniques for large scale feature models. In *GPCE '08: Proceedings of the 7th international conference on Generative programming and component engineering*, pages 13–22, New York, NY, USA, 2008. ACM.

[20] P. Schobbens, J.C. Trigaux P. Heymans, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, Feb 2006.

[21] S. Segura. Automated analysis of feature models using atomic sets. In *First Workshop on Analyses of Software Product Lines (ASPL 2008). SPLC'08*, pages 201–207, Limerick, Ireland, September 2008.

[22] S.E. Sim, S. Easterbrook, and R.C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.

[23] V. Sugumaran, S. Park, and K. Kang. Software product line engineering. *Commun. ACM*, 49(12):28–32, 2006.

[24] W.F. Tichy. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.

[25] J. White, D. Schmidt, D. Benavides P. Trinidad, and Ruiz-Cortés. Automated diagnosis of product-line configuration errors in feature models. In *Proceedings of the 12th Sofware Product Line Conference (SPLC'08)*, Limerick, Ireland, September 2008.

[26] J. White and D.C. Schmidt. Filtered cartesian flattening: An approximation technique for optimally selecting features while adhering to resource constraints. In *First International Workshop on Analyses of Software Product Lines (at SPLC'08)*, Limerick, Ireland, September 2008.