# Trends in Topics at SE Conferences (1993-2013)

George Mathew
Computer Science, NCSU
Raleigh, North Carolina
george.meg91@gmail.com

Amritanshu Agrawal
Computer Science, NCSU
Raleigh, North Carolina
aagrawa8@ncsu.edu

Tim Menzies
Computer Science, NCSU
Raleigh, North Carolina
tim@menzies.us

***Abstract*—We report a topic modeling analysis of the the abstracts and titles from 9291 papers published in 11 top-ranked SE conferences between 1993 to 2013. Seven topics are identified as the dominant themes in modern software engineering. We show that these topics are not static; rather, some of them are becoming decidedly less prominent over time (modeling) while others are become very prominent indeed (defect analysis).**

**Also, by clustering conferences according to the topics they usually publish, we can see that SE conferences fall into four large groups. For example, for the last 20 years, ASE, FSE and ICSE have been publishing mostly the same work (exceptions: FSE publishes somewhat more work on program analysis compared to ASE or ICSE).**

**Our results highlight the enormous impact of a program committee (PC) on a research conference. Even though PC members comprise less than a quarter of the authors with published papers, the topics of their papers matches almost exactly the topics seen in all published papers.**

**Using these results, we offer numerous recommendations including how to plan an individual's research program; how to intelligently make new conferences or merge old ones; how to better recruit program committees; and how to encourage a broader range of topics at SE conferences.**

***Keywords*-Software Engineering; Bibliometrics; Topic Modelling; Text Mining**

## I. Introduction

Understanding the SE literature is a complex and important task. If done correctly, researcher can better discover new and exciting topics in SE as well to better showcase their works and build their research career. If done incorrectly, researchers could waste much time on unpromising directions to obtain results that may not be publishable.

Figure 1 gives the annual count of the total number of papers published in 11 top SE conferences(listed in Table I). We can see that there is a steady trend in the rise in number of papers published and that over 9000 papers from more than 10000 different authors have been published in top SE conferences over the last decade. How can we obtain an overview of that large corpus?

Existing methods for studying large text corpuses can be (very) labor intensive and, systematic literature reviews [39], [78] and manual content analysis [13], [15], [27], rely extensively on subjective human judgment. Thus, it can be very hard to reproduce/ improve/ or even refute such studies.

Automatic methods have their problems as well. Simple methods can rank top performing SE institutions and SE scholars [26]; or list the most cited SE articles [78]. But

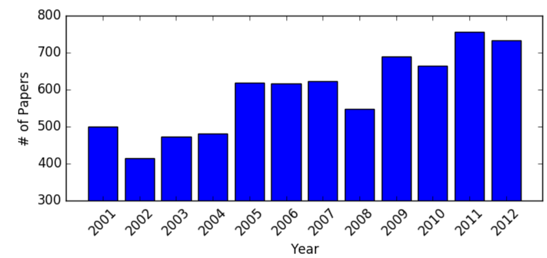


Fig. 1: # Papers published in SE in 11 top conferences.

those methods offer little insight on the changing nature of the field, nor where to focus effort for (say) the next five years of research. More complicated methods like LDA (Latent Dirichlet Allocation [12], [49]) can summarize a large set of documents as a small set of "topics" (where each "topic" is a set of words commonly occuring together in particular documents). However, we cannot recommend standard LDA since recently we have found that LDA's topics are *highly unstable*; i.e. they will dramatically change [1] if (a) the input ordering of the documents is changed, or (b) if the parameters controlling LDA are altered.

When applying topic modeling to SE, we recommend the LDA-DE *stable topics* analysis [1]. LDA-DE is combination of LDA and a search-based software engineering method called DE (Differential Evolution [61]). LDA-DE automatically tunes LDA's parameters in order to remove unstable topics. Using the stable topics from LDA-DE, this paper automatically analyzes the titles and abstracts of 9291 papers published in the 11 SE conferences of Table I from 1993 to 2013.

This paper explores the following research questions:

**RQ1: What topics are most frequently accepted to which conferences?**

> **Result 1**
> *Table II lists the seven topics dominate the SE literature.*

Note that this study found *more* than seven topics (and all those topics are listed below). However, several of those appeared in so few papers that this study focused on the remaining topics. This paper will use this first result to create very simple visual presentations of the topics seen in 9291 research papers.

| Short | Name | Impact | Start | Group |
|---|---|---|---|---|
| FASE | International Conference on Fundamental Approaches to Software Engineering | 42 | 1998 | A |
| GPCE | Generative Programming and ComponentEngineering | 37 | 2000 | A |
| MSR | Working Conference on Mining Software Repositories | 32 | 2004 | B |
| WCRE | Working Conference on Reverse Engineering | 43 | 1995 | B |
| ICPC | IEEE International Conference onProgram Comprehension | 43 | 1997 | B |
| SCAM | International Working Conference on Source Code Analysis & Manipulation | 15 | 2001 | C1 |
| ICSM | IEEE International Conference on Software Maintenance | 53 | 1994 | C2 |
| CSMR | European Conference on Software Maintenance and Reengineering | 40 | 1997 | C2 |
| ICSE | International Conference on Software Engineering | 117 | 1994 | C3 |
| FSE | ACM SIGSOFT Symposium on the Foundations of Software Engineering | 59 | 1993 | C3 |
| ASE | IEEE/ACM International Conference on Automated Software Engineering | 55 | 1994 | C3 |

TABLE I: Corpus of conferences studied in this paper. The "Group" column shows conferences that publish "very similar" topics (where "very similar" is computed via a cluster analysis shown later in this paper).

| # | Top 7 terms | Name |
|---|---|---|
| 0 | test, execution, based, techniques, program, approach, cases | Testing |
| 1 | web, server, applications, client, devices, security, mobile | Applications |
| 2 | program, analysis, code, language, static, dynamic, | Program Analysis |
| 3 | software, engineering, tools, work, project, package, process | Tools & Project |
| 4 | code, source, software, bug, developers, defect, approach | Defect Analysis |
| 5 | model, systems, design, requirements, architecture, approach, components | Modelling |
| 6 | quality, maintenance, study, change, metrics, results, analysis | Maintenance |

TABLE II: The top 7 topics of SE as seen in our corpus.

| Topic | Top Papers |
|---|---|
| Testing | 1999: Dynamically discovering likely program invariants to support program evolution, J. Cockrell, W. G. Griswold, D. Notkin, M. D. ernst [18]<br>2002: Visualization of test information to assist fault localization, J. T. Stasko, M. J. Harrold, J. A. Jones [36];<br>2005: Cute: a concolic unittesting engine for C, G. A. Agha, K. Sen, D. Marinov [58] |
| Applications | 2009: Automatic creation of sql injection and cross-site scripting attacks, A. Kiezun, K. Jayaraman, P. J. Guo, M. D. Ernst [37];<br>2012: On the naturalness of software, M. Gabel, E. T. Barr, A. Hindle, Z. Su, P. T. Devanbu [34]<br>2008: Static detection of cross-site scripting vulnerabilities, G. Wassermann, Z. Su [76] |
| Program Analysis | 2004: Static checking of dynamically generated queries in database applications, P. T. Devanbu, Z. Su, C. Gould [28];<br>2002: Tracking down software bugs using automatic anomaly detection, M. S. Lam, S. Hangal [31];<br>2000: Model checking programs, S. Park, G. P. Brat, W. Visser, K. Havelund [75] |
| Tools & Project | 2010: Codebook: discovering and exploiting relationships in software repositories, T. Zimmermann, Y. P. Khoo,A. Begel [9]<br>2008: Latent social structure in open source projects, C. Bird [11]<br>2002: Expertise browser: a quantitative approach to identifying expertise. J. D. Herbsleb, A. Mockus [46] |
| Defect Analysis | 2006: Who should fix this bug?, J. Anvik, G. C. Murphy, L. Hiew [4]<br>2003: Populating a release history database from version control and bug tracking systems, M. Fischer, H. C. Gall, M. Pinzger [20];<br>2005: When do changes induce fixes?, J. Sliwerski, T. Zimmermann, A. Zeller [60] |
| Modelling | 2003: Scaling step-wise refinement, D. S. Batory, A. Rauschmayer, J. N Sarvela [7]<br>2001: Identifying similar code with program depenence graphs, J. Krinke [40]<br>2006: Feature oriented refactoring of legacy applications, D. S. Batory, J. Liu, C. Lengauer [41] |
| Maintainance | 1998: Clone detection using abstract syntax trees, A. Yahin, L. M. De Moura, L. Bier, M. Sant'anna, I. D. Baxter [8]<br>1998: Detection of logical coupling based on product release history, K. Hajek, M. Jazayeri, H. C. Gall [21]<br>2006: Mining metrics to predict component failures, N. Nagappan, A. Zeller, T. A. Ball [48] |

TABLE III: Most cited papers within the seven SE topics seen within our corpus 1993-2013

| Topic | Top Papers |
|---|---|
| Testing | 2009: Automatically finding patches using genetic programming, C. le Goues, W. Weimer,S. Forrest, T. H. Nguyen [77]<br>2009: MSeqGen: Object-Oriented Unit-Test Generation via Mining Source Code, S. Thummalapenta, W. Schulte, N. Tillmann, J. de Halleux, T. Xie [69];<br>2009: Lightweight fault localization using multiple coverage types, M. Harrold, Y. Yu, R. A. Santelices, J. A. Jones [57] |
| Applications | 2009: Automatic creation of sql injection and cross-site scripting attacks, A. Kiezun, K. Jayaraman, P. J. Guo, M. D. Ernst [37];<br>2011: Patching vulnerabilities with sanitization synthesis, T. Bultan, M. Alkhalaf, F. Yu [81]<br>2012: On the naturalness of software, M. Gabel, E. T. Barr, A. Hindle, Z. Su, P. T. Devanbu [34] |
| Program Analysis | 2009: Mining exception handling rules as sequence association rules, S. Thummalapenta, T. Xie [68];<br>2009: Refactoring sequential Java code for concurrency via concurrent libraries, D. Dig, J. Marrero, M. D. Ernst [16];<br>2010: Template based reconstruction of complex refactorings, M. Kim, N. Sudan, K. Prete, N. Rachatasumrit [53] |
| Tools & Project | 2010: Codebook discovering and exploiting relationships in software repositories, T. Zimmermann, Y. P. Khoo, Andrew Begel [9]<br>2009: Predicting build failures using social network analysis on developer communication, D. Herlea, A. Schroter, T. H. Nguyen, T. Wolf [79]<br>2009: How tagging helps bridge the gap between social and technical aspects in software development, M. D. Storey, C. Treude [71] |
| Defect Analysis | 2009: Fair and balanced bias in bugfix datasets, A. Bernstein, C. Bird, V. Filkov, P. T. Devanbu, J. Duffy, E. Aune, A. Bachmann [10]<br>2011: Dealing with noise in defect prediction, S. Kim, R. Wu, H. Zhang, L. Gong<br>2009: The secret life of bugs: Going past the errors and omissions in software repositories, J. Aranda, G. D. Venolia [5] |
| Modelling | 2009: Learning operational requirements from goal models, J. Kramer, D. Alrajeh, A. Russo, S. Uchitel [3]<br>2009: Model evolution by runtime parameter adaptation, G. Tamburrelli, C. Ghezzi, R. Mirandola, I. Epifani [17]<br>2010: Model checking lots of systems efficient verification of temporal properties in SPL, J. Raskin, A. Legay, A. Classen, P. Heymans, P. Schobbens [14] |
| Maintainance | 2009: Predicting faults using the complexity of code changes, A. E. Hassan [32]<br>2009: How we refactor and how we know it, C. Parnin, A. P. Black, E. R. Murphy-Hill [47]<br>2009: Discovering and representing systematic code changes, M. Kim, D. Notkin [38] |

TABLE IV: Most cited papers within the seven SE topics seen in our corpus 2009-2013

**RQ2: What are the fewest number of papers that most reflect current topics in SE?**

> **Result 2**
> *Table III & Table IV lists the 3 papers per topic with most citations.*

These 7*3=21 papers would be useful for educators running a one semester introductory subject on software engineering research.

**RQ3: What SE topics are growing more/less prominent?**

> **Result 3**
> *Of our seven topics, one is clearly growing less prominent while two others are becoming very prominent.*

This result could be used by funding bodies or new faculty to decide the focus of their work for the next few years.

**RQ4: Which SE conferences have the most overlap with other conferences?**

> **Result 4**
> *As seen in Table I, our conferences fall into 3 groups (A,B,C), the third of which sub-divides into (C1,C2,C3).*

This result could be used to reduce the number of SE conferences. For example, the eleven conferences studied here might be replaced with five.

On the other hand, if the goal is to *strength* rather than *replace* conferences, then the results of **RQ5** would be useful.

**RQ5: What topics most distinguish different conferences?**

> **Result 5**
> *Within many of the groups there exist* strong topics *that select for one particular conference, but exclude others.*

The **RQ5** results could guide the creation of new conference call for papers(CFP) that focus more on the topics that most distinguish particular conferences.

**RQ6: What is the connection between a conference's program committee and the accepted papers?** In one of the strongest results of this paper, we find that:

> **Result 6**
> *The topics of the papers of the program committee are nearly a perfect match for the papers accepted to a conference.*

This result could be used by PC chairs to recognize when submissions do not overlap with the papers of the PC (at which point the chairs could either recruit new PC members or advise authors that they should consider withdrawing their paper and submit elsewhere).

**RQ7: How to focus a research career?** We look at the top 20%, 5%, 1% most cited authors and ask "how many topics should a researcher focus on?" Our conclusion is:

> **Result 7**
> *Do not focus. A clear pattern in the most cited authors is that their citations increase as they venture into more topics.*

That is, the most successful SE researchers are generalists that do not focus on one particular topic.

### A. Structure of this paper

After notes on the threats to validity and related work, this paper describes the dataset, LDA and the stable topic analysis of LDA-DE. After that, we explore in detail the above seven questions.

### B. Contribution of this paper

Prior research in this area explored fewer topics via a mostly manual analysis that focused on less than explored here; For example,

- We use titles and abstracts while other studies just use the titles [24];
- We study 9291 papers and many other studies explore orders of magnitude fewer papers [13], [74];

A key contribution of this analysis is that it is fully automatic and repeatable. Unlike traditional literature review methods (which can be very labor intensive) all the conclusions made in this paper were built from auto-generated structures created by open source data mining tools. We recommend such a repeatable approach since this allows other researchers to repeat/extend/improve or even refute our conclusions. The same can be not be said for many of the traditional methods for studying the literature.

### C. Threats to Validity

The main threat to validity of our results is *sampling bias*. This study can access the accepted papers *but not the rejected ones*. Perhaps if both accepted and rejected papers are studied, then some patterns other that the ones reported here might be discovered. That said, sampling those rejected papers is problematic. We can access 100% of accepted papers via on-line means. Sampling rejected papers imply asking researchers to supply their rejected papers. The experience in the research community with such surveys is that only a small percent of respondents reply [56] in which case we would have a second sampling bias amongst the population of rejected papers. At the time of this writing, we do not know how to resolve this issue.

As to the questions of which accepted papers to study, this paper is based on papers accepted to the 11 conferences studied in Vasilescu et al. 's MSR'13 paper [73] and shown in Table I. We use this source since it is a recent paper that has survived peer review and that offers its conference data along with tools to analyze that data.

As to other threats to validity, this paper is less prone to *tuning bias* and *order bias* that other software analytics papers. As discussed later, we use search-based SE methods to find our tunings and those tunings are selected to mitigate against the effects of changing the order of the training examples.

## II. Related Work

This section discusses few of the prominent works with their notable findings. The overall picture we will offer is that many researchers have published studies about the SE literature. However, the methods of this paper explore a far larger space.

### A. Bibliometrics

Numerous bibliometric studies analyzing SE conferences have been published in SE, most of which have been in the last 2 decades.

Cai & Card [13] analyzed 691 papers from 14 leading journals and conferences in SE. They find that 89% of conference papers focus on 20% of subjects in SE, including software/program verification, testing and debugging, and design tools and techniques. Note that our **RQ1** results (that most SE explores just 7 topics) is a conformation of the Cai & Card finding.

Fernandes [19] studies 70,000 articles in SE and finds that the number of authors for articles in software engineering is increasing on an average of around 40% per decade. The author does not analyze the publishing trends of successful authors in SE (this is an issue on which we offer much detail).

Bibiliometric studies based on Turkish [23] and Canadian [25] SE communities were performed by Garousi et al. to study the citation landscape in the respective countries. They identify a lack of diversity in the general SE spectrum for example, limited focus on requirements engineering, software maintenance and evolution, and architecture. They also identify a low involvement from the industry in SE. Since these studies were localized to a certain country, it explored lesser number of papers

Fernandes et al. performs a citation based study to identify the top cited paper in SE using total number of citations and average annual number of citations. In this paper(see **RQ2**), we augment their approach to identify top cited paper in different topics in SE.

Vasilescu et al. study the health of SE conferences with respect to community stability, openness to new authors, inbreeding, representativeness of the PC with respect to the authors community, availability of PC candidates, and scientific prestige [74].

Systä et al. in their 2012 publication "Inbreeding in Software Engineering" [64] analyzes acceptance of papers (co)authored by PC members in 6 leading SE conferences. They conclude that there is a lot of variance in the acceptance rate(0-70%) of papers (co)authored by PC members. Although their research methodology is very insightful, the data used in their study is limited to 6 conferences over a period of 5 years. Thus, in this study, in **RQ6**, we expand on their research using data from 11 conferences over 20 years to draw a clearer conclusion and we further examine the connection between PC and accepted papers.

### B. Topic Modelling

Topic Modeling has been used in various spheres of Software Engineering. According to a survey reported by Sun et al. [63], topic modeling is applied in various SE tasks, including source code comprehension, feature location, software defects prediction, developer recommendation, traceability link recovery, re-factoring, software history comprehension, software testing and social software engineering. There are works in Requirements Engineering where it was necessary to analyze the text and come up with the important topics [6], [44], [66]. People have used topic modeling in prioritizing test cases, and identifying the importance of test cases [33], [80], [82]. Increasingly, it has also become very important to have automated tools to do a Systematic Literature Review (SLR) [72]. We found these papers [2], [43], [55] who have used clustering algorithms (topic modeling) to do SLR.

Outside of SE, in the general Computer Science(CS) literature, a 2013 paper by Hoonlor et al. highlighted the prominent trends in CS [35].This paper identified trends, bursty topics, and interesting inter-relationships between the American National Science Foundation (NSF) awards and CS publications, finding, for example, that if an uncommonly high frequency of a specific topic is observed in publications, the funding for this topic is usually increased. The authors adopted a Term Frequency Inverse Document Frequency(TFIDF) based approach to identify trends and topics. A similar approach can be performed in SE considering how closely CS is related to SE.

Garousi and Mantyla recently have adopted a Topic Modeling and Word Clustering based approach to identify topics and trends in SE [24] similar to the research by Hoonloor et al. [35]. Although their method is very novel and in line with the current state of the art, they use only the titles of the papers for modelling topics. This might lead to inaccurate topics as titles are generally not very descriptive of the field the paper is trying to explore. This issue is addressed in the current work where we use the abstracts of the paper to build topic models.

## III. Dataset & Research Method

### A. Gathering Data

For studying and analyzing SE conferences we use a database of 11 conferences and its program committee members from 1993-2013. This data was obtained in the form of a SQL dump from the work of Vasilescu et al. "A historical dataset of software engineering conferences" in MSR 2013 [73] and is shown in Table I. Note that the last column shows the first edition of the conference in the available dataset while the third column of the table shows the conference series impact factor computed based on the Simple H-INdex Estimator (SHINE) [59], as follows:

- For a given series of conference, an impact of $h = 40$ means that it has 40 papers, each with at least 40 citations during this period.

Note that the numbers in the third column are computed for the region 2001 to 2012 since not not all our conferences exist outside that time period. The data contains 9291 papers and 10183 distinct authors who have published in these 11 conferences from Table I over 20 years.

The data published by Vasilescu et al. does not contain the abstract or other sections of the papers which can be used to perform text based analytics. Thus, we join the data from the SQL tables with the abstracts of all papers from the Association of Computing Machinery(ACM) harvested by Tang et al. on "aminer" [65][1].

### B. Latent Dirichlet Allocation

This paper uses Latent Dirichlet Allocation (LDA) to summarize the SE papers. LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

The basic LDA model [12], [49] is shown in Figure 2. A collection of $D$ documents is assumed to contain *T* topics expressed with *W* different words. Each document $d \in D$ of length $N_d$ is modelled as a discrete distribution $\theta(d)$ over the set of topics. Each topic corresponds to a multinomial distribution over the words. Discrete priors $\alpha$ are assigned to the distribution of topics vectors $\theta$ and $\beta$ for the distributions of words in topics.

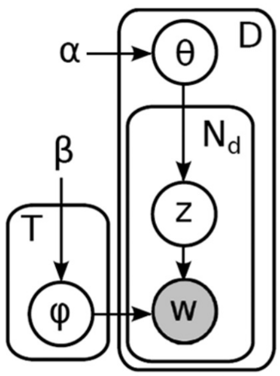

Fig. 2: LDA Model

As shown in Figure 2, the outer plate spans documents and the inner plate spans word instances in each document (so the *w* node denotes the observed word at the instance and the *z* node denotes its topic). The inference problem in LDA is to find hidden topic variables *z*, a vector spanning all instances of all words in the dataset. LDA is a problem of Bayesian inference. The original method used is a variational Bayes approximation of the posterior distribution [12] and alternative inference techniques use Gibbs sampling [29] and expectation propagation [45].

LDA learns the various distributions (the set of topics, their associated word probabilities, the topic of each word, and the particular topic mixture of each document).To find these various distributions, LDA takes particular set of parameters namely, number of clusters (*k*), number of iterations (*n*), document topic prior ($\alpha$), and word topic prior ($\beta$).

The literature suggests that different sets of these parameters can give rise to unstable topics [50], [54]. Rocco et al. [50] used LDA for traceability link recovery and ran multiple experiments to verify the topics generated by LDA are not unstable to conclude their results. Various other researchers [30], [67] have used LDA in various tasks of SE and have mentioned that LDA needs to be stable. To overcome the instability problem, multiple researchers recommend tuning the above mentioned parameters of LDA using Genetic Algorithm (GA) [42], [51], [62] rather than having to manually explore the parameters without much explanation. A major drawback of LDA-GA was that it used stochastic analysis to use topics generated as a *feature selector* for further experiments. Some people [22], [70] achieved higher stability by just increasing the number of cluster size. However, these studies used manual methods to find those tunings and hence, for pragmatic reasons, they did not explore a large range of tunings for the parameters. Recently, Agarwal et al. [1] have used DE to stabilize the LDA model. Their LDA-DE automatically tunes LDAs parameters in order to stabilize LDA model. They explored much larger range of tunings within tens of evaluations of the model, not hundreds.

Differential evolution just randomly picks three different vectors $B, C, D$ from a list called $F$ (the *frontier*) for each parent vector A in $F$ [61]. Each pick generates a new vector $E$ (which replaces $A$ if it scores better). $E$ is generated as follows:

$$\forall i \in A, E_i = \begin{cases} B_i + f * (C_i - D_i) & \textit{if } \mathcal{R} < cr \\ A_i & \textit{otherwise} \end{cases} \quad (1)$$

where $0 \leq \mathcal{R} \leq 1$ is a random number, and $f, cr$ are constants that represent mutation factor and crossover factor respectively(following Storn et al. [61], we use $cr = 0.3$ and $f = 0.75$). Also, one $A_i$ value (picked at random) is moved to $E_i$ to ensure that $E$ has at least one unchanged part of an existing vector. The Goal of this DE is to maximize the Raw Score ($\Re_n$) which is very similar to Jaccard Similarity [22]. Agarwal et al. [1] have explained the $\Re_n$ in much more detail.

In order to answer the research questions we adopt this topic modeling based approach [1] coupled with a bibliometric based study(§III-B) on the dataset described in §III-A. Topic modeling using LDA needs three parameters to be set primarily; 1) $k$: Number of topics 2) $\alpha$ : Dirichlet prior on the per-document topic distributions 3) $\beta$: Dirichlet prior on the per-topic word distribution.

In similar studies [24], the authors suggest using 67 topics and default values of $\alpha$ and $\beta$ as provided by the LDA package in R statistical toolkit [52]. They [24] choose the number of topics by minimizing the absolute Likelihood error. But we optimized these tunable parameters LDA-DE as described earlier in §III-B. Post tuning, the optimal values for $k$, $\alpha$ & $\beta$ are 12, 0.847 & 0.764 respectively.

## IV. Answers to Research Questions

Using these parameters, LDA was performed on the abstracts(titles if abstract was not available) of the dataset. The heatmap of Figure 3 shows scores denoting the occurrence of a certain topic at a particular conference. An interesting find from the heatmap is that the topics 2, 4, 5, 6 and 10 do not contribute much to any of the conferences used in this study (less than 5% in all of the conferences). Thus, we ignore those rare topics and remodel the data using 7 topics as shown in Figure 4. Note the difference between Figure 3 & Figure 4 which arises due to assigning the 5 minority topics to the other 7 topics. We label these topics based on their terms and is shown in Table II. This analysis will hence use these 7 topics to address our research questions.

[1]http://aminer.org/citation

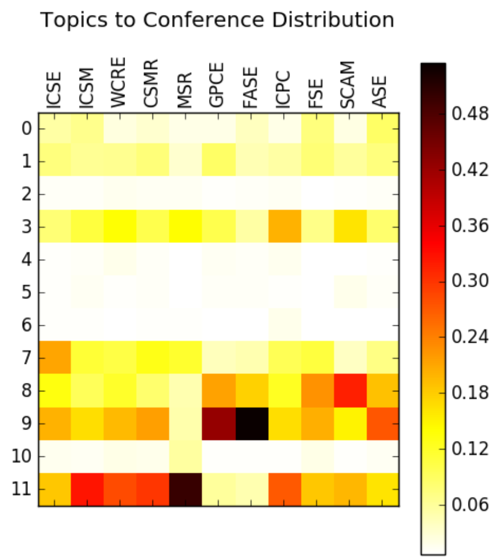


Fig. 3: Preliminary heatmap showing the relation between Topics and Conferences. Colors denote how often a particular conference has papers with certain topics. Note that all columns sum to 1.0 and many rows(topics) have very low scores. For the rest of this paper, we discard the rows that are mostly whitespace(i.e. the second, fourth, fifth, and sixth rows) and remodel the data using 7 topics as showin Figure 4.

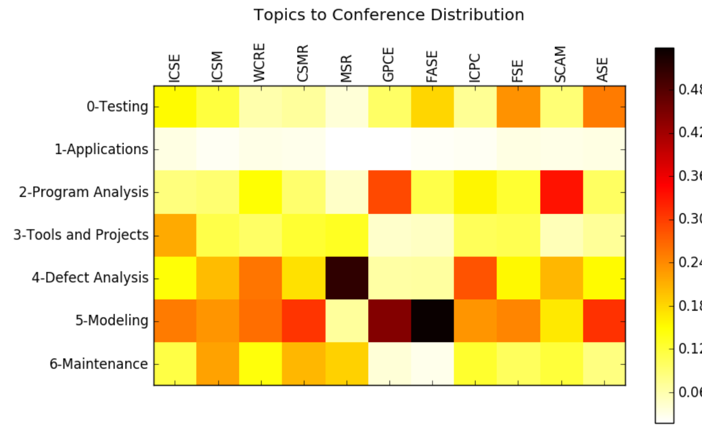


Fig. 4: Heatmap showing relation between the 7 reduced Topics and Conferences between 1993 to 2013. We study the terms in these topics and label them. Refer Table II.

| Years | Authors |
|---|---|
| 1993-2013 | T. Zimmermann, A. Zeller, M. J. Harrold, T. Xie, M. D. Ernst, G. C. Murphy, P. T. Devanbu, D. Notkin, S. Kim, N. Nagappan, H. C. Gall, A. Orso, J. Whitehead, C. Bird, A. E. Hassan, D. Marinov, D. S. Batory, Z. Su, K. Sen, M. P. Robillard, J. A. Jones, M. Pinzger, T. A. Ball, M. B. Dwyer, D. Lo |
| 2009-2013 | T. Xie, P. T. Devanbu, A. E. Hassan, C. Bird, D. Lo, T. Zimmermann, T. N. Nguyen, D. Poshyvanyk, M. Kim, T. T. Nguyen, M. Lanza, B. Adams, H. A. Nguyen, A. Bernstein, J. M. Al-Kofahi, S. Kim, A. Bachmann, N. Nagappan, B. Murphy, H. Mei, H. C. Gall, S. Thummalapenta, C. Kastner, S. Apel, D. M. German |

TABLE V: Top cited authors in SE seen within our corpus, at different time intervals. Listed in descending order of citations (so most cited appear first).

**RQ1: What topics are most frequently accepted to which conferences?**
Table II shows the seven most frequent topics in our conference papers. Figure 4 shows how often these topics appear in our 11 conferences.

Is it reasonable to claim that seven topics capture the depth and extend of SE research? Perhaps. Recall the Cai & Card study from our *Related Work* section which showed that 89% of conferences papers covered a more 20% of known topics in SE. Which is to say that there is some precedent of claims that SE conferences usually condone research only in certain restricted narrow areas.

**RQ2: What are the fewest number of papers that most reflect current topics in SE?**
Table III and Table IV showed what we would call the canonical papers in software engineering. The first table shows the top 3 most cited papers in each topic since 1993 while the second table focuses on more recent papers since 2009. Also shown in Table V are the 25 most cited authors from our sample of conferences.

**RQ3: What SE topics are growing more/less prominent?**
Figure 5 shows the frequency at which topics appeared at our 11 SE conferences for the period 1993 to 2013. For each year, topics are sorted top to bottom in increasing frequency of occurrence (so topics shown at the *bottom* are *more* frequent).

The arrows in that figure show some trends in three topics becoming more/less prominent over time. For example:

- While topic 5 (Software Modeling) was highly popular from 1995-2009, it has become decidely less so in recent years.
- On the other hand, topic 4 (Defect Analysis) has gained popularity gradually from 2004 and is the most popular topic from 2010 onwards.
- Topic 1 ("Software Apps") is found very rarely in these SE conferences. The explanation for this is that this topic ventures into computer security and reliability of web services. This is an area of explosive growth as witnessed by the very long list at http://goo.gl/z35YKz. We conjecture that those researchers went outside of the conferences studied in this paper in order to build their community.
- A topic to watch in the near future is Topic 0 (Testing). This as slowly gaining popularity from 2008 and based on the trend, it might become the most popular topic in SE.

Further to that last point, one reason for the growth in Topic 0 (Testing) is the recent success of a conference not covered by our corpus of 11 conferences. ISSTA (the International Symposium on Software Testing and Analysis) is a vibrant, but not large, conference on testing which attracts many of the most cited authors in SE. We conjecture that, as ISSTA grows more successful, then papers that do not fit into the smaller ISSTA venue "spill over" into our 11 conferences.

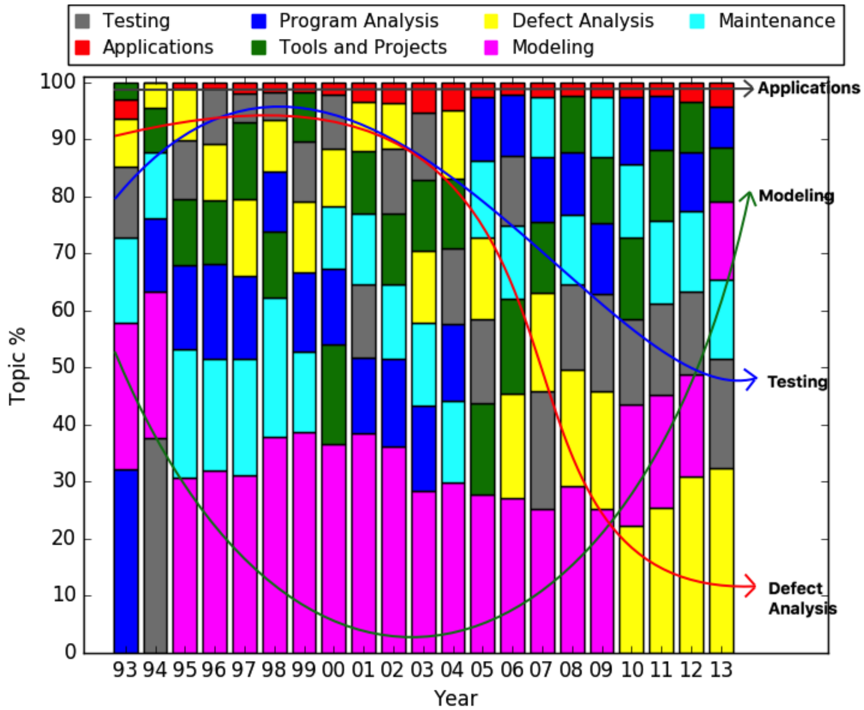


Fig. 5: This figure shows a stacked column chart of the contribution(in %) of each topic in a certain year between 1993-2013. The bars are stacked in ascending order;i.e. the lowest bar is most published that year and the highest bar is the least. For example, in the year 2013, *Defect Analysis* is the most published topic with 34% whereas *Applications* is the least with less than 5%. The arrows indicates the trend in a topic over the years.

**RQ4: Which SE conferences have the most overlap with other conferences?**
Figure 6 shows the results of a bottom-up nearest-pair clustering of topics and conferences for the period 2009 to 2013. The topics are clustered based on the terms present in the them and the conferences are clustered based on the distribution of topics within them. In the figure, the colors indicates how strongly or weakly a topic on the y-axis is covered in a conference on x-axis. This clustering shows that most conferences accept papers on Topic 4 (Defect Analysis) with the exception of ICSM and SCAM which rarely accepts papers on that topic.

From the hierarchical tree on the x-axis we can see that there is a strong association between certain conferences with respect to the topics they publish. Those associations are:

- A: GPCE, FAE
- B: MSR, WCRE, ICPC
- C: other. This group sub-divides into:
  - C1: SCAM
  - C2: ICSM, CSMR
  - C3: ICSE, FSE, ASE

For the reader interested in the longer term history of our field, we repeat the analysis of Figure 6 for two prior time periods:

- Figure 7 clusters topics and conferences for 2001 to 2008. This figures show that, earlier in its history, MSR was very different to WCRE and ICPC.

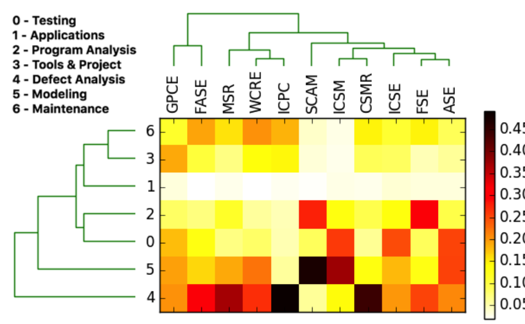


Fig. 6: Figure shows the hierarchical clustering heatmap (described in **RQ4**) of Topics and Conferences between the years 2009-2013. The values in each cell are similar to Figure 3 and Figure 4. For example, WCRE and ICPC are similar to each other. Similarly the terms in the topic *Testing* is similar to the terms in the topic *Program Analysis*.

- Figure 8 clusters topics and conferences for 1993 to 2000. MSR is missing from this figure since that conference did not start till 2004. Interestingly, ICSE, FSE, ASE appear clustered into the same tree; i.e. the C3 group mentioned above comprising ICSE, FSE, ASE has existed now for 20 years or more.

Based on this clustering, we suggest a merge rule for long running conferences. Every so often, steering committees

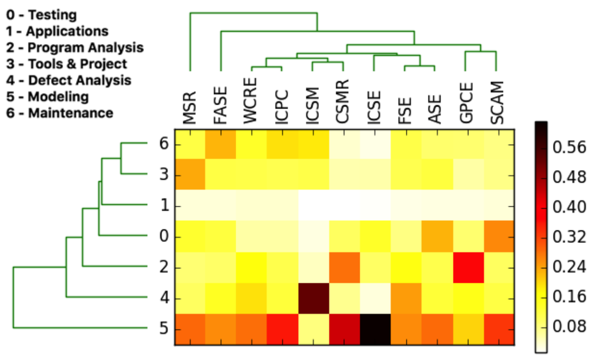


Fig. 7: Hierarchical Clustering Heatmap Results 2001-2008: Topics vs Conferences. Similar to Figure 6.

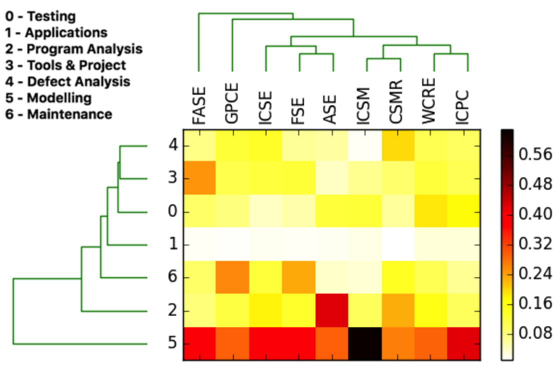


Fig. 8: Hierarchical Clustering Heatmap Results 1993-2000: Topics vs Conferences. Similar to Figure 6.

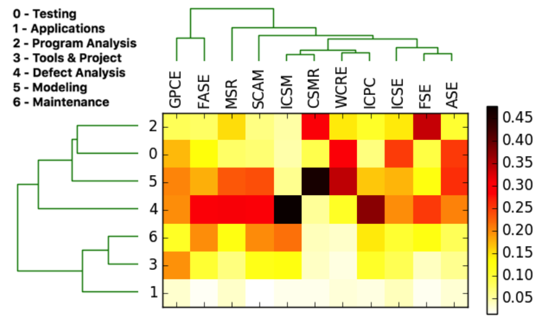


Fig. 9: Hierarchical Clustering Heatmap showing association between Topics and Conferences from the Program Committee members between 2009-2013

| Conference | X = # papers accepted | Y = # paper with PC authors | 100*Y/X Percent from PC |
|---|---|---|---|
| MSR | 169 | 75 | 44% |
| ICPC | 202 | 79 | 39% |
| CSMR | 272 | 100 | 37% |
| ICSM | 373 | 137 | 37% |
| WCRE | 239 | 82 | 34% |
| GPCE | 98 | 25 | 26% |
| SCAM | 98 | 24 | 24% |
| ASE | 399 | 74 | 19% |
| ICSE | 944 | 164 | 17% |
| FSE | 295 | 51 | 17% |
| FASE | 149 | 23 | 15% |
| all | 3238 | 834 | 26% |

TABLE VI: Percentage of papers from member of the program committee, sorted in descending order by percentage. Data from 2009-2013

should conduct a topic analysis and consider merging with similar conferences. If applied to the 11 conferences here, then that might mean merging (ASE and FSE) or (ICSM and CSMR) or (ICSM and CSMR).

Merging of ASE and FSE seems timely since, post merge, the SE calendar would contain two (not three) major events each year: ICSE in May and ASE+FSE in (say) November.

**RQ5: What topics most distinguish different conferences?**
Rather than *merge* similar conferences, an alternate strategy might be to *emphasize* their differences. Consider, for example, FSE and ASE. While they are very similar, there are clear differences in their focus areas:

- FSE publishes much more than ASE on Topic 2 (Program Analysis);
- ASE publishes much more than FSE of Topics 0,5 (Testing, Software Modeling)

If the community finds these distinctions important enough to preserve, then ASE and FSE could change their call for papers and program committees such that each accept more papers on their particular focus area.

**RQ6: What is the connection between a conferences program committee and the accepted papers?**
Figure 9 shows the frequency of topics from accepted papers from conference program committee (PC) members 2009-2013. A curious feature of this paper is how similar this appears to the percentage of topics from *all* authors (see Figure 6): while the clustering of the rows is slightly different, the rows from the same topics in the two figures is nearly identical. What makes this similarity very curious is that the papers from PC members of makes up (on average only 26% of all papers: see Table VI). That is, even though the PC members are in the minority, the distribution of topics in their accepted papers mimics almost exactly the distributions seen in all the authors.

It should be expected that there is *some* degree of similarity between the topics seen in the papers of the PC and the rest of the conference's papers. That said, the degree of similarity of the rows in Figure 6 and Figure 9 is most remarkable.

These **RQ6** results echo some of the results on Vasilescu et al. [74](mentioned in the *Related Work* section); i.e. PC members are not just gatekeepers culling sub-standard work. Rather, their actions suggest they are promoting something akin to their own research agenda.

**RQ7: How to focus a research career?**
Finally, we ask what find of research career leads to most impact (where "impact" is measured in terms of number of citations). Figure 10 compares all authors to those with 1%, 10,%, 20% most number of citations. Figure 10 shows how many topics are covered by the papers of those four groups of authors. The overall patterns is quite clear: publishing in

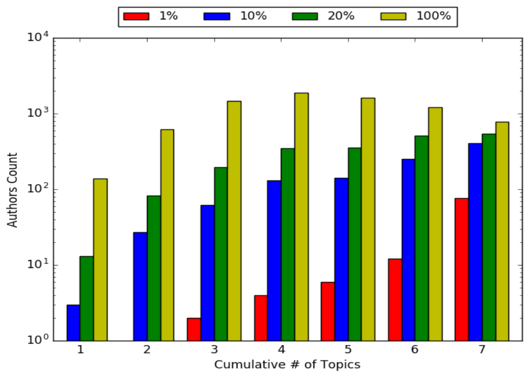


Fig. 10: Represents number of authors(on log scale) exploring different number of topics for the top 20%,10%,1% authors based on citation counts. For example, 85% of the top 1% authors explore all the 7 topics and less than 2% explore 3 different topics.

more topics in SE is rewarded by more citations.

From this, we offer the following advice to junior faculty: do not over-specialize. The field of SE tends to most reward researchers who explore multiple research directions.

## V. Conclusions, Implications, Future Work

Topic modeling is an insightful method for studying a large text corpus. In this analysis, we have explored over 9000 papers from the last 20 years of software engineering.

### A. Results and Surprises

Some of our results echo some prior work. Like Cai & Card [13], we find that SE conferences are focused on a relatively small number of topics (just the seven reported in Table II). Also, like Vasilescu et al. [74] we find that the program committee of a conference has a very large impact on the conference; specifically, topics not supported by the PC are generally not accepted to a conference.

Otherwise, this study leads to some very surprising results such as the dramatic usurpation of Topic5 (Modeling) by Topic4 (Defect Analysis), as seen in Figure 5. Also, we had thought MSR was something of an outlier conference. Based on these results, it now seems to be hovering near WCRE and ICPC (see Figure 6).

Another surprise was the degree of overlap between conferences. Informally, we were aware of some overlap between ASE, ICSE and FSE. But we did not expect such large extent of that overlap, and that it dates back 20 years (see Figure 6, Figure 7 and Figure 8).

### B. Implications

*1) Implications for Research Agendas:* There is much that is actionable in these results. For example, Figure 5 might be used to guide strategic choices such as faculty hiring and what research agendas faculty might choose to follow for several years to come. Further, Figure 6 could guide researchers' tactical choices about where to submit their latest paper.

*2) Implications for Conference Organization:* This analysis found which conferences significantly overlapped other conferences and what topics most select for particular conferences (see Figure 6). Those result could be used to:

- Rewrite conference call for papers such that when two conferences are closely aligned, each individual conference focuses on the topic(s) that distinguish between them.
- Create new conferences to specialize in areas not well covered elsewhere.

Yet another approach might be to merge conferences that have a significant topic overlap. We would suggest that conference steering committees run a topic analysis every few years, then consider (possibly) merging with their nearest neighboring conference.

In discussion with colleagues, two frequently asked questions is (a) " why merge?" and (b) "why not let conferences evolve naturally?". To this we reply that:

- SE conferences are merging all the time:
  - e.g. ESEM = IEEE Metric + ISESE(In 2007)
  - e.g. SANER = CSMR + WCRE(In 2014)
- The conference calendar is already overcrowded and it is hard to find gaps for new events. For example, in the early years of the PROMISE conference were characterized as a "chase" for such a gap. Further, this crowded conference cycle had detrimental effects on academic events; e.g. ASE 2012 has 60% of its usual size due to an unanticipated clash with a call for papers by another conference.
- It is not clear that conferences are "naturally evolving". The topics seen at ASE and FSE and ICSE have been in lockstep for the last twenty years– which is not what we would expect if each conference was evolving separately. Also, in a result that echoes prior work by Cai & Card [13], we have found that SE conferences focus on a very narrow range of topics. Such lack-of-diversity is not consistent with "natural evolution".

For a proposal on how to extend the range of topics explored at conferences, see the "New Idea" discussion later in the paper.

*3) Implications for Program Committees:* As to the program committee results of Figure 9, this suggests that there is much to be gained by a fine-grained analysis of the particular topic biases of PC members.

When selecting a PC, it might be prudent to do a topic analysis and:

- Accept less PC members who explore the same topics;
- Accept more PC members that explore different topics.

Also, after papers are submitted, program committee chairs could use topic modeling to recognize *outlier submissions*; i.e. those papers that do not overlap with the papers of the PC. At which point, they could:

- Advise authors that they should consider withdrawing their paper and submit elsewhere)
- Or they could recruit new PC members to cover the topic gaps.

*4) A New Idea about "New Idea" Tracks:* If conferences want to encourage more topic evolution at their forum, the following strategy might be useful.

Most large conferences now run a "New Idea" tracks. That track could be recruited to handle outlier submissions, as follows:

- The "new ideas" program committee could be formed as usual, but might call it "new topics".
- Papers for the "new topics" track are not submitted by authors but rather by PC chairs when they recognize papers submitted to the main conference that fall outside of the topic set of the PC members.
- The authors of those "outlier" papers could then be given a choice to either (a) leave their paper in the main track (and risk rejection) or (b) withdraw the paper (so they can submit it elsewhere) or (c) have the paper transferred to the "new topics" track.
- Change the acceptance criteria for the "new topics" track: select for technical competency and distance to the topics of the PC in the main track.
- Finally, have a policy to always invite the authors of the better "new topics" papers to next year's PC.

### C. Future Work

Having completed this analysis of Vasilescu et al. corpus, we have been looking into other sources of data. The next steps in this research would be to consider how our results change across other samples:

- One sample of conferences not explored here are those devoted to security or programming languages (specfically: POPL, PLDI, OOPSLA/SPASH). Our pre-experimental intuition is that each of these are a sub-field onto themselves but it would be interesting to test that intuition.
- Several prominent conferences do not appear in Table I and it would be useful to add them in (specfically: ISSTA, RE, ESEM, ISSRE).
- Another important sample to be explored are the senior journals in our field (specifically: TSE, TOSEM, JSS, IST, EMSE, SPE, IEEE Software, Software & Systems Modeling).
- Apart from conferences and journals, other insightful topic modelings might include an exploration of the text of funding bodies (e.g. 20 years of funding calls for software resaerch from DoD, NSF. NASA or their equivalent in other countries). By comparing trends in topics from the funding bodies to those from conferences+journals, we could check the alignment between research funding and published research results.

## Appendix: Reproduction Package

All the scripts and data used in this analysis will be made available on-line, once this work has been accepted by some peer-reviewed venue.


## Acknowledgments

We would like to acknowledge B.Vasilescu et al. for providing a well documented relational database of 11 SE conferences, its program committees and authors. We would also like to thank J.Tang et al. for harnessing the abstracts from the ACM portal.


## References


[1] A. Agrawal, T. Menzies, and W. Fu. What is wrong with topic modeling? (and How to Fix it using Search-based SE). Submitted to ICSE'17. Available from http://tiny.cc/LDADE.

[2] A. Al-Zubidy and J. C. Carver. Review of systematic literature review tools. *University Of Alabama Technical Report*, 2014.

[3] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Learning operational requirements from goal models. In *Proceedings of the 31st international conference on software engineering*, pages 265–275. IEEE Computer Society, 2009.

[4] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370. ACM, 2006.

[5] J. Aranda and G. Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proceedings of the 31st International Conference on Software Engineering*, pages 298–308. IEEE Computer Society, 2009.

[6] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 95–104. ACM, 2010.

[7] D. Batory, J. N. Sarvela, and A. Rauschmayer. Scaling step-wise refinement. *IEEE Transactions on Software Engineering*, 30(6):355–371, 2004.

[8] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 368–377. IEEE, 1998.

[9] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, pages 125–134. IEEE, 2010.

[10] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu. Fair and balanced?: bias in bug-fix datasets. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 121–130. ACM, 2009.

[11] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu. Latent social structure in open source projects. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 24–35. ACM, 2008.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[13] K.-Y. Cai and D. Card. An analysis of research topics in software engineering–2006. *Journal of Systems and Software*, 81(6):1051–1058, 2008.

[14] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. Model checking lots of systems: efficient verification of temporal properties in software product lines. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 335–344. ACM, 2010.

[15] N. Coulter, I. Monarch, and S. Konda. Software engineering as seen through its research literature: A study in co-word analysis. *Journal of the American Society for Information Science*, 49(13):1206–1223, 1998.

[16] D. Dig, J. Marrero, and M. D. Ernst. Refactoring sequential java code for concurrency via concurrent libraries. In *Proceedings of the 31st International Conference on Software Engineering*, pages 397–407. IEEE Computer Society, 2009.

[17] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *Proceedings of the 31st International Conference on Software Engineering*, pages 111–121. IEEE Computer Society, 2009.

[18] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):99–123, 2001.

[19] J. M. Fernandes. Authorship trends in software engineering. *Scientometrics*, 101(1):257–271, 2014.

[20] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*, pages 23–32. IEEE, 2003.

[21] H. Gall, K. Hajek, and M. Jazayeri. Detection of logical coupling based on product release history. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 190–198. IEEE, 1998.

[22] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 582–591. IEEE Press, 2013.

[23] V. Garousi. A bibliometric analysis of the turkish software engineering research community. *Scientometrics*, 105(1):23–49, 2015.

[24] V. Garousi and M. V. Mäntylä. Citations, research topics and active countries in software engineering: A bibliometrics study. *Computer Science Review*, 19:56–77, 2016.

[25] V. Garousi and T. Varma. A bibliometric assessment of canadian software engineering scholars and institutions (1996-2006). *Computer and Information Science*, 3(2):19, 2010.

[26] R. L. Glass and T. Y. Chen. An assessment of systems and software engineering scholars and institutions (1999–2003). *Journal of Systems and Software*, 76(1):91–97, 2005.

[27] R. L. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software technology*, 44(8):491–506, 2002.

[28] C. Gould, Z. Su, and P. Devanbu. Static checking of dynamically generated queries in database applications. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 645–654. IEEE, 2004.

[29] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[30] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162. IEEE, 2014.

[31] S. Hangal and M. S. Lam. Tracking down software bugs using automatic anomaly detection. In *Proceedings of the 24th international conference on Software engineering*, pages 291–301. ACM, 2002.

[32] A. E. Hassan. Predicting faults using the complexity of code changes. In *Proceedings of the 31st International Conference on Software Engineering*, pages 78–88. IEEE Computer Society, 2009.

[33] H. Hemmati, Z. Fang, and M. V. Mantyla. Prioritizing manual test cases in traditional and rapid release environments. In *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, pages 1–10. IEEE, 2015.

[34] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu. On the naturalness of software. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 837–847. IEEE, 2012.

[35] A. Hoonlor, B. K. Szymanski, and M. J. Zaki. Trends in computer science research. *Communications of the ACM*, 56(10):74–83, 2013.

[36] J. A. Jones, M. J. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In *Proceedings of the 24th international conference on Software engineering*, pages 467–477. ACM, 2002.

[37] A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst. Automatic creation of sql injection and cross-site scripting attacks. In *2009 IEEE 31st International Conference on Software Engineering*, pages 199–209. IEEE, 2009.

[38] M. Kim and D. Notkin. Discovering and representing systematic code changes. In *Proceedings of the 31st International Conference on Software Engineering*, pages 309–319. IEEE Computer Society, 2009.

[39] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering - A systematic literature review, 2009.

[40] J. Krinke. Identifying similar code with program dependence graphs. In *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, pages 301–309. IEEE, 2001.

[41] J. Liu, D. Batory, and C. Lengauer. Feature oriented refactoring of legacy applications. In *Proceedings of the 28th international conference on Software engineering*, pages 112–121. ACM, 2006.

[42] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang. Improving trace accuracy through data-driven configuration and composition of tracing features. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 378–388. ACM, 2013.

[43] C. Marshall and P. Brereton. Tools to support systematic literature reviews in software engineering: A mapping study. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 296–299. IEEE, 2013.

[44] A. K. Massey, J. Eisenstein, A. I. Antón, and P. P. Swire. Automated text mining for requirements analysis of policy documents. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 4–13. IEEE, 2013.

[45] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[46] A. Mockus and J. D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proceedings of the 24th international conference on software engineering*, pages 503–512. ACM, 2002.

[47] E. Murphy-Hill, C. Parnin, and A. P. Black. How we refactor, and how we know it. *IEEE Transactions on Software Engineering*, 38(1):5–18, 2012.

[48] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *Proceedings of the 28th international conference on Software engineering*, pages 452–461. ACM, 2006.

[49] S. I. Nikolenko, S. Koltcov, and O. Koltsova. Topic modelling for qualitative studies. *Journal of Information Science*, page 0165551515617393, 2015.

[50] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 68–71. IEEE, 2010.

[51] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 522–531. IEEE Press, 2013.

[52] M. Ponweiser. Latent dirichlet allocation in r. *Institute for Statistics and Mathematics*, 2012.

[53] K. Prete, N. Rachatasumrit, N. Sudan, and M. Kim. Template-based reconstruction of complex refactorings. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[54] S. Rao and A. Kak. Retrieval from software libraries for bug localization: a comparative study of generic and composite text models. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pages 43–52. ACM, 2011.

[55] A. Restificar and S. Ananiadou. Inferring appropriate eligibility criteria in clinical trial protocols without labeled data. In *Proceedings of the ACM sixth international workshop on Data and text mining in biomedical informatics*, pages 21–28. ACM, 2012.

[56] G. Robles, J. M. González-Barahona, C. Cervigón, A. Capiluppi, and D. Izquierdo-Cortázar. Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 222–231. ACM, 2014.

[57] R. Santelices, J. A. Jones, Y. Yu, and M. J. Harrold. Lightweight fault-localization using multiple coverage types. In *Proceedings of the 31st International Conference on Software Engineering*, pages 56–66. IEEE Computer Society, 2009.

[58] K. Sen, D. Marinov, and G. Agha. Cute: a concolic unit testing engine for c. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 263–272. ACM, 2005.

[59] Simple h-index estimation. http://shine.icomp.ufam.edu.br/index.php. Accessed: February 2013.

[60] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? In *ACM sigsoft software engineering notes*, volume 30, pages 1–5. ACM, 2005.

[61] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[62] X. Sun, B. Li, H. Leung, B. Li, and Y. Li. Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks. *Information and Software Technology*, 66:1–12, 2015.

[63] X. Sun, X. Liu, B. Li, Y. Duan, H. Yang, and J. Hu. Exploring topic models in software engineering data analysis: A survey. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 357–362. IEEE, 2016.

[64] T. Systä, M. Harsu, and K. Koskimies. Inbreeding in software engineering conferences, 2012.

[65] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

[66] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein. Studying software evolution using topic models. *Science of Computer Programming*, 80:457–479, 2014.

[67] S. W. Thomas, H. Hemmati, A. E. Hassan, and D. Blostein. Static test case prioritization using topic models. *Empirical Software Engineering*, 19(1):182–212, 2014.

[68] S. Thummalapenta and T. Xie. Mining exception-handling rules as sequence association rules. In *Proceedings of the 31st International Conference on Software Engineering*, pages 496–506. IEEE Computer Society, 2009.

[69] S. Thummalapenta, T. Xie, N. Tillmann, J. De Halleux, and W. Schulte. Mseqgen: Object-oriented unit-test generation via mining source code. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 193–202. ACM, 2009.

[70] K. Tian, M. Revelle, and D. Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 163–166. IEEE, 2009.

[71] C. Treude and M.-A. Storey. How tagging helps bridge the gap between social and technical aspects in software development. In *Proceedings of the 31st International Conference on Software Engineering*, pages 12–22. IEEE Computer Society, 2009.

[72] G. Tsafnat, P. Glasziou, M. K. Choong, A. Dunn, F. Galgani, and E. Coiera. Systematic review automation technologies. *Systematic reviews*, 3(1):1, 2014.

[73] B. Vasilescu, A. Serebrenik, and T. Mens. A historical dataset of software engineering conferences. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 373–376. IEEE Press, 2013.

[74] B. Vasilescu, A. Serebrenik, T. Mens, M. G. van den Brand, and E. Pek. How healthy are software engineering conferences? *Science of Computer Programming*, 89:251–272, 2014.

[75] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda. Model checking programs. *Automated Software Engineering*, 10(2):203–232, 2003.

[76] G. Wassermann and Z. Su. Static detection of cross-site scripting vulnerabilities. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 171–180. IEEE, 2008.

[77] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *Proceedings of the 31st International Conference on Software Engineering*, pages 364–374. IEEE Computer Society, 2009.

[78] C. Wohlin. An analysis of the most cited articles in software engineering journals-2000. *Information and Software Technology*, 49(1):2–11, 2007.

[79] T. Wolf, A. Schroter, D. Damian, and T. Nguyen. Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st*

*International Conference on Software Engineering*, pages 1–11. IEEE Computer Society, 2009.
[80] G. Yang and B. Lee. Predicting bug severity by utilizing topic model and bug report meta-field. *KIISE Transactions on Computing Practices*, 21(9):616–621, 2015.
[81] F. Yu, M. Alkhalaf, and T. Bultan. Patching vulnerabilities with sanitization synthesis. In *Proceedings of the 33rd International Conference on software engineering*, pages 251–260. ACM, 2011.
[82] Y. Zhang, M. Harman, Y. Jia, and F. Sarro. Inferring test models from kates bug reports using multi-objective search. In *Search-Based Software Engineering*, pages 301–307. Springer, 2015.