Bit-Tech (Binary Digital - Technology)
Vol.7, No.2, Desember 2024
Available online at: http://jurnal.kdi.or.id/index.php/bt

Deteksi Code Smell dengan Pendekatan Machine Learning: Analisis Bibliometrik

Ellysha Dwiyanthi Kusuma[1)*]

[1)]*Fakutas Sains dan Teknologi, Universitas Buddhi Dharma*
Jl. Imam Bonjol No. 41 Karawaci, Tangerang, Indonesia
[1)]ellyshad_k@yahoo.com

Article history:

Received 21 Nov 2024;
Revised 28 Nov 20xx;
Accepted 28 Nov 2024;
Available online 27 Des 2024

Keywords:

Analisis Bibliometrik
Code Smell Detection
Machine Learning
Nvivo
Perangkat Lunak

Abstrak

Dalam pengembangan perangkat lunak, aspek efisiensi dan keberlanjutan sangat penting. Teknik *refactoring* digunakan untuk mengurangi dampak dari *code smell* yang merupakan bentuk *technical debt* akibat implementasi *source code* yang tidak mematuhi prinsip-prinsip rekayasa perangkat lunak. Tetapi proses *refactoring* membutuhkan biaya yang banyak. Oleh karena itu, *code smell* perlu dideteksi untuk mencegah dampak buruknya terhadap kualitas perangkat lunak. Dalam penelitian ini, pencarian jurnal dilakukan dengan *tools Publish or Perish* dengan kata kunci "*Code Smell Detection*". Terkumpul 200 jurnal yang terindeks Scopus, namun hanya 69 yang tersedia secara terbuka (*open access*). Analisis bibliometrik dalam penelitian ini menyajikan tinjauan literatur mengenai deteksi *code smell* dengan pendekatan *machine learning*. Penelitian ini memberikan wawasan mengenai kontribusi domain deteksi *code smell*. Hasil analisis menunjukkan bahwa institusi dari India menjadi penyumbang terbanyak penelitian dalam domain ini, diikuti oleh RRC dan Amerika Serikat. Penelitian ini banyak dilakukan pada tahun 2021, namun tren dalam domain ini mengalami penurunan pada tahun berikutnya. Dampak yang diberikan penelitian ini berupa potensi pengembangan alat deteksi code smell otomatis berbasis machine learning yang dapat mempercepat proses identifikasi masalah kode dalam proses pengembangan perangkat lunak. Dengan teknologi ini, pengembang dapat meningkatkan efisiensi dalam menemukan code smell sehingga dapat mengurangi biaya pemeliharaan perangkat lunak. Jenis publikasi yang paling banyak adalah dalam bentuk artikel sebanyak 40 publikasi. Melalui pembentukan *word cloud* dengan software *NVivo*, kata *code*, *smells* dan *software* adalah kata yang sering dipakai. Lalu untuk kata kunci yang terdeteksi melalui *software VOSviewer*, kata-kata *code smell detection*, *detection*, *code*, *machine*, dan *method* memiliki keterkaitan satu sama lain.

I. PENDAHULUAN

Dalam pembuatan sebuah perangkat lunak, perangkat lunak yang efisien dan mudah untuk dikelola merupakan suatu keharusan. Selama pemeliharaan, terkadang perangkat lunak tersebut perlu dikembangkan oleh pengembang untuk menambahkan fitur baru yang diperlukan, meningkatkan fitur yang sudah ada dan memperbaiki *bug* [1]. Namun, seringkali pengembang perangkat lunak diharuskan melakukan pengembangan di bawah tenggat waktu yang ketat, sehingga pengembang tidak memiliki cukup waktu untuk menemukan solusi desain yang baik sebelum mengimplementasikan pengembangan mereka demi tercapainya ketepatan waktu. Pengembangan seperti ini memiliki dampak yang dinamakan *technical debt*, dimana dampak ini akan menghasilkan biaya tinggi di masa depan karena mengorbankan kualitas kode di tahap awal pengembangan perangkat lunak [1][2].

Code smell merupakan salah satu bentuk *technical debt* yang serius yaitu suatu akibat yang ditimbulkan karena implementasi *source code* yang tidak mengikuti prinsip-prinsip rekayasa perangkat lunak dalam pengembangan perangkat lunak [3]. Pengurangan dampak dari *code smell* dapat dilakukan dengan menerapkan proses *refactoring*. *Refactoring* dilakukan karena bermanfaat untuk membangun kembali struktur internal perangkat lunak tanpa mengubah perilaku eksternalnya. Proses *refactoring* melibatkan tiga langkah, yaitu mengidentifikasi adanya *code smell*, menerapkan operasi *refactoring* yang sesuai untuk memperbaikinya, dan memastikan keberlanjutan fungsi dari sistem [4].

* Corresponding author

ISSN 2622-2728 (online) 2622-271X (print)
doi: 10.32877/bt.v7i2.1822

Pendeteksian *code smell* menjadi sangat penting untuk dilakukan dalam pengembangan perangkat lunak agar dapat meningkatkan kualitas perangkat lunak dan mengurangi resiko kesalahan dan kegagalan dalam perangkat lunak [5]. Selain itu deteksi *code smell* secara dini dapat membantu untuk mengevaluasi kualitas dari perangkat lunak [3] serta meningkatkan efisiensi dalam pengembangan perangkat lunak yang dilakukan. Pendeteksian awal terhadap *code smell* memiliki peran yang sangat penting dalam mengurangi biaya proses *refactoring*. Oleh karena itu, dengan melakukan pendeteksian *code smell* secara otomatis menjadi suatu kebutuhan karena metode manual cenderung rumit untuk dilakukan dan akan memakan waktu yang tidak sedikit [4][6].

Penelitian yang telah dilakukan sebelumnya menyebutkan bahwa metode *machine learning* sudah terbukti menjadi pendekatan yang dapat dilakukan oleh para pengembang perangkat lunak untuk meningkatkan kualitas perangkat lunak dan memungkinan bagi para pengembang untuk melakukan identifikasi pola masalah dalam kode yang telah mereka buat. [6]

Dalam penelitian ini akan dilakukan analisis bibliometrik untuk menganalisis secara kuantitatif terhadap penelitian yang berhubungan dengan pendeteksian *code smell* menggunakan *machine learning*. Analisis bibliometrik terhadap pendeteksian *code smell* penting untuk dilakukan agar dapat mengetahui tren terkini dalam domain ini dan memberikan wawasan mengenai seberapa banyak penelitian yang telah dilakukan tentang deteksi *code smell*. Ini akan membantu para peneliti dan pengembang perangkat lunak dalam memilih teknik yang tepat dalam mendeteksi *code smell*.

Pendekatan bibliometrik dalam penelitian ini memberikan kontribusi karena membantu memetakan tren penelitian, mengidentifikasi peneliti dan institusi yang banyak berperan, serta menilai pengaruh dan dampak penelitian. Dengan pemahaman berupa analisis yang lebih sistematis dan terstruktur terhadap literatur yang ada dalam bidang ini, penelitian ini bertujuan agar peneliti dapat membuat keputusan yang lebih tepat dalam merencanakan penelitian mereka di bidang ini. Dengan menggunakan metode bibliometrik, peneliti dapat memperoleh wawasan yang lebih mendalam mengenai tren penelitian mengenai deteksi *code smell* dan kolaborasi antar peneliti. Dengan penelitian ini diharapkan dapat meningkatkan kolaborasi antara akademisi dan praktisi industri perangkat lunak dalam mengembangkan alat untuk melakukan pendeteksian *code smell* secara otomatis dalam lingkungan pengembangan perangkat lunak yang sesungguhnya.

II. TINJAUAN PUSTAKA

Penelitian yang dilakukan sebelumnya [7] menggunakan pendekatan tinjauan konvensional mengenai *code smell*. Tinjauan tersebut juga dilakukan oleh peneliti yang lain [8]. Penelitian yang lain melakukan tinjauan terhadap *code smell* yang dinamakan tinjauan naratif dimana pemetaan biblometrik yang dihasilkan dapat memvisualisasikan konten publikasi penelitian dalam bentuk lanskap ilmiah [9]. Penelitian ini pun akan menggunakan analisis bibliometrik untuk menganalisis tren dalam pendeteksian code smell.

A. Code Smell

Code smell diperkenalkan oleh Kent Beck [10] dan populer dalam konteks *refactoring* oleh Martin Fowler [11]. *Code smell* adalah istilah yang digunakan dalam pengembangan perangkat lunak merujuk pada tanda atau indikasi bahwa kode memiliki potensi masalah atau desain yang buruk [12]. Meskipun kode tersebut mungkin berfungsi dengan benar, akan tetapi hal ini menunjukkan bahwa ada sesuatu yang bisa menyebabkan masalah di masa depan, seperti kesulitan dalam perawatan, pemahaman, atau pengembangan lebih lanjut [13]. Penyebab *code smell* bisa terjadi karena beberapa faktor, seperti desain awal yang buruk atau tidak konsisten, kurangnya perhatian terhadap prinsip desain seperti SOLID, adanya tekanan untuk mengirimkan produk dengan cepat tanpa memperhatikan kualitas kode, dan kurangnya *factoring* dalam proses pengembangan [14].

Beberapa contoh *code smell* yaitu adanya kode yang sama atau mirip yang muncul di beberapa tempat dalam sebuah *project* sehingga meningkatkan resiko inkonsistensi jika ada perubahan. Contoh berikutnya nama metode atau fungsi yang terlalu panjang membuat sulit untuk dibaca, dipahami atau diuji. Belum lagi misalnya metode atau fungsi dalam satu *class* terlalu banyak mengakses data dari class lain yang menunjukkan ketergantungan yang kuat. Lalu adanya sebuah *large class* yang melanggar prinsip *Single Responsibility* karena mempunyai tanggung jawab yang terlalu luas. Terdapat sebuah *dead code* [15] atau kode yang tidak relevan atau tidak pernah digunakan lagi. Contoh lainnya adalah *god object* atau *god class* [14] yang berarti sebuah objek atau kelas melakukan terlalu banyak tugas. Kemudian penambahan komentar yang berlebihan dapat dicegah dengan memberikan nama variabel atau fungsi yang lebih deskriptif. Penggunaan angka atau string tanpa konteks membuat kode sulit dipahami. Bisa juga terdapat perubahan kecil pada satu bagian kode membutuhkan banyak perubahan di berbagai tempat atau disebut dengan *shotgun surgery* [16].

Solusi yang dapat dilakukan untuk memperbaiki *code smell* dengan cara *refactoring*, yaitu sebuah teknik yang digunakan untuk memperbaiki desain kode tanpa mengubah perilaku eksternalnya. Menerapkan prinsip desain yang baik seperti SOLID [17], DRY (*Don't Repeat Yourself*) [18], dan KISS (*Keep It Simple, Stupid*) [19]. Melakukan *review* kode secara berkala dengan melibatkan pengembang lain dalam memastikan kualitas kode tetap tinggi. Dan menggunakan alat otomatisasi seperti *SonarQube* [20] untuk mendeteksi *code smell*

secara otomatis. Pemahaman akan tentang *code smell* dan cara penanganannya, diharapkan pengembang dapat memastikan kode tetap bersih, mudah dibaca, dipahami dan dikelola dalam waktu jangka yang panjang.

B. Bibliometrik

Bibliometrik adalah metode analisis kuantitatif terhadap dokumen ilmiah, seperti artikel jurnal, buku, ataupun makalah konferensi [21]. Tujuannya adalah mengukur produktivitas penelitian melalui cara mengindentifikasi jumlah publikasi seorang peneliti, institusi, ataupun negara. Lalu untuk menilai dampak publikasi dengan menggunakan metrik seperti banyaknya jumlah sitasi untuk menilai pengaruh sebuah artikel atau penulis. Selanjutnya untuk melihat pola kolaborasi dan hubungan antar peneliti atau antar institusi dalam menghasilkan sebuah publikasi. Kemudian untuk memetakan ilmu pengetahuan dengan menganalisis hubungan antar bidang studi untuk memahami bagaimana ide-ide baru muncul dan berkembang. Dan yang terakhir sebagai evaluasi jurnal dalam menentukan tingkat pengaruh sebuah jurnal melalui metrik seperti *Impact Factor* [22].

Beberapa indikator yang sering digunakan dalam metode bibliometrik yaitu jumlah publikasi seorang penulis, kelompok, atau institusi dalam tingkat produktivitasnya. Lalu jumlah sitasi yang menandakan seberapa sering karya ilmiah dirujuk oleh peneliti lain. Kemudian *H-Index* atau Indeks H yang merupakan indeks dalam mengukur produktivitas dan dampak karya ilmiah seorang peneliti berdasarkan jumlah publikasi dan sitasi. Selanjutnya ada *Impact Factor* yaitu mengukur reputasi jurnal berdasarkan jumlah rata-rata sitasi artikel yang diterbitkan dalam jurnal tersebut. Adanya *co-authorship analysis* yaitu kegiatan analisa kolaborasi antar penulis berdasarkan publikasi bersama. Terakhir *keyword co-occurrence* adalah cara identifikasi hubungan antar topik dengan melihat kata kunci yang sering muncul bersamaan.

Teknik-teknik dalam metode bibliometrik adalah *citation analysis*, *co-citation analysis*, *bibliographic coupling*, dan *network analysis*. *Citation analysis* adalah menganalisa hubungan antar dokumen berdasarkan pola sitasi. *Co-citation analysis* merupakan cara mengukur seberapa sering dua dokumen dirujuk bersama dalam publikasi lain. *Bibliographic coupling* yaitu sebuah analisa hubungan antar dua dokumen yang merujuk pada referensi yang sama. *Network analysis*, membuat jaringan hubungan antar penulis, institusi, ataupun kata kunci.

Pengaplikasian metode bibliometrik adalah untuk evaluasi penelitian yang sering digunakan oleh universitas atau lembaga penelitian untuk menilai produktivitas dan dampak ilmiah. Untuk membantu membuat kebijakan dalam memahami tren penelitian dan menentukan prioritas pendanaan. Lalu untuk manajemen jurnal yaitu menentukan strategi penerbitan jurnal ilmiah. Dan sebagai pemetaan pengetahuan dalam identifikasi tren dan pola sebuah perkembangan di bidang-bidang tertentu. Metode bibliometrik biasanya didukung oleh perangkat lunak seperti *NVivo*, *VOSviewer*, *CiteSpace*, atau *Gephi* untuk membantu memvisualisasikan hasil analisis, seperti peta kolaborasi atau jaringan kata kunci.

C. Publish or Perish

Software Publish or Perish [23] dikembangkan oleh Anne-Wil Harzing dan dirancang untuk membantu peneliti, akademisi, dan mahasiswa dalam menganalisis publikasi ilmiah dan mengevaluasi kinerja akademik berdasarkan data publikasi dan sitasi. Fitur utama yang dapat digunakan adalah pencarian di berbagai database publikasi ilmiah, seperti *Google Scholar*, *Microsoft Academic*, CrossRef, PubMed, Scopus dan *Web of Science*. Fitur lainnya sebagai analisis bibliometrik yang memberikan berbagai metrik kinerja ilmiah, termasuk jumlah publikasi atau artikel yang ditemukan, jumlah sitasi yang diterima oleh semua publikasi penulis, H-*index* yaitu mengukur produktivitas dan dampak penelitian, G-*index* mengutamakan publikasi yang lebih banyak disitasi, Hc-*index* untuk menghitung H-*index* dengan mempertimbangkan usia publikasi, *Impact Per Paper* (IPP) merupakan rata-rata jumlah sitasi per artikel, dan *Citations Per Year* yaitu menghitung jumlah sitasi rata-rata per tahun. Fitur lainnya adalah *export data* dimana data hasil analisis dapat diekspor untuk digunakan dalam analisis lebih lanjut menggunakan perangkat lunak lainnya seperti *Excel*.

Cara kerja *Publish or Perish* yang pertama adalah memasukkan kata kunci atau nama penulis atau kombinasi lainnya dengan memasukkan *range* tahun, lalu *Publish or Perish* akan mengambil data dari database yang dipilih seperti *Google Scholar*. Data yang diperoleh akan diolah menjadi metrik bibliometrik yang relevan dan ditampilkan dalam format tabel. Lalu pengguna dapat mengekspor data untuk digunakan lebih lanjut.

Publish or Perish mempunyai kelebihan yaitu gratis, tidak berbayar, memiliki antarmuka yang mudah digunaakan oleh pengguna, mendukung pencarian di berbagai database sehingga lebih fleksibel, dan memberikan analisa yang komprehensif berupa metrik yang mendalam untuk menilai kinerja penelitian. Selain kelebihan, *Publish or Perish* juga memiliki kekurangan yaitu sebagian besar data berasal dari *Google Scholar*, memungkinan adanya duplikasi atau data yang kurang akurat, keterbatasan menganalisa lebih lanjut dan lebih kompleks sehingga data perlu diekspor dan diolah dengan *software* lainnya, dan pembaruan data bergantung pada *database* yang diakses sehingga data-data tersebut bukanlah data *real time*.

Penggunaan Publish or Perish sangat cocok dan bermanfaat dalam dunia akademik, untuk dosen, peneliti, mahasiswa, institusi akademik. Mendukung pengelolaan publikasi ilmiah, meneliti tren di bidang tertentu, menilai produktivitas staf, dan mengevaluasi kinerja dalam publikasi.

D. NVivo

NVivo [24] adalah sebuah *software* analisis data kualitatif yang dirancang untuk membantu peneliti dalam mengatur, menganalisis, dan menemukan pengetahuan dari data non-numerik atau tidak terstruktur seperti wawancara, survei terbuka, dokumen, video, atau media sosial. *NVivo* juga dapat digunakan dalam analisis bibliometrik, khususnya untuk analisa yang berbasis teks dan tematik, seperti memetakan pola atau tema dari publikasi ilmiah. *Nvivo* berguna untuk menganalisis isi dari artikel, laporan, atau publikasi untuk mengidentifikasi tema, pola atau tren dalam teks. Lalu memungkinkan pengguna membuat kode pada data teks, misal abstrak atau isi artikel, untuk menemukan topik atau kategori yang sering muncul. Membantu menemukan kata kunci atau frasa yang sering digunakan dalam kumpulan dokumen serta membantu mengidentifikasi hubungan antar topik melalui analisis tematik. Sebagai analisis metadata dalam arti mengelompokkan dokumen berdasarkan atribut seperti penulis, tahun, atau jurnal serta menganalisis hubungan antara atribut-atribut tersebut. *Nvivo* juga dapat memberikan visualisasi berupa peta kata (*word cloud*), matriks hubungan (*relationship matrix*), atau grafik hierarki tema.

Kelebihan yang ditawarkan yaitu fleksibilitas dalam mendukung segala jenis format data, termasuk data teks, audio, video, dan media digital. Memiliki kemampuan analisis yang mendalam untuk menemukan hubungan dan pola yang tidak terlihat dalam data mentah. Cocok untuk proyek tim besar yang melibatkan banyak data dalam kolaborasi dan organisasi. Lalu kemudahan integrasi dengan alat lain untuk analisa lebih lanjut dengan ekspor data. Kekurangan *Nvivo* adalah kurangnya kemudahan dipahami oleh pengguna baru karena antarmuka terasa begitu kompleks dan membutuhkan waktu untuk menguasai. Biaya lisensi cukup tinggi terutama untuk institusi kecil ataupun perorangan. Dan membutuhkan sumber daya komputer dengan spesifikasi tinggi untuk menangani proyek yang besar.

E. VOSviewer

VOSviewer [25] adalah perangkat lunak yang dirancang untuk membuat dan memvisualisasikan jaringan bibliometrik, seperti hubungan antara publikasi, kolaborasi antar penulis dan mencari peta kata kunci. *VOSviewer* menampilkan elemen-elemen dalam bentuk peta berbasis *node* (simpul) dan *edge* (garis penghubung). Ukuran node mencerminkan bobot elemen seperti frekuensi kata atau jumlah sitasi. *VOSviewer* juga dapat menganalisis *cluster* dengan mengelompokkan elemen yang memiliki hubungan erat ke dalam *cluster* berdasarkan algoritma VOS (*Visualization of Similarities*). Data yang diproses bisa dari berbagai sumber bibliometrik seperti *Scopus*, *Web of Science*, dan *PubMed*. *VOSviewer* dapat diperoleh secara gratis dan diunduh dari situs resminya.

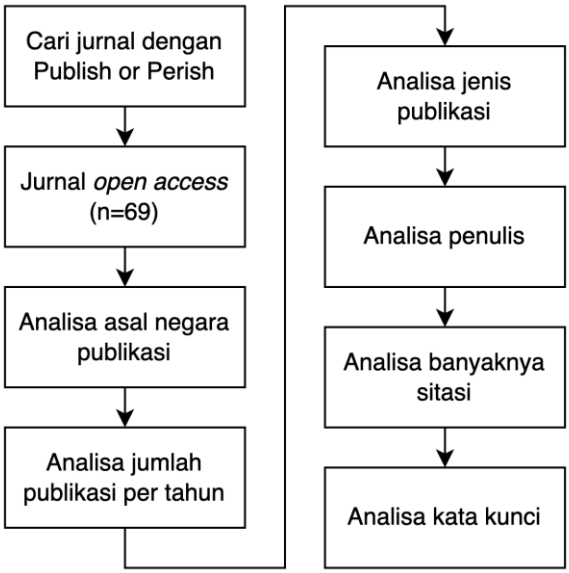
F. Word Cloud

Word cloud [26] adalah representasi visual dari kata-kata yang digunakan dalam sebuah teks atau kumpulan teks, dimana ukuran besar kecilnya setiap kata mencerminkan frekuensi atau pentingnya kata tersebut dalam data yang dianalisis. *Word cloud* sering digunakan untuk mengidentifikasi kata-kata kunci atau tema utama dalam data dengan cepat. Tujuan *word cloud* adalah penyajian informasi secara cepat, membantu pengguna memahami topik utama atau tren dari data dalam waktu singkat. Sering digunakan dalam penelitian kualitatif untuk menganalisis tema dalam data survei, ulasan pelanggan, atau tanggapan terbuka. Sebagai visualisasi hasil analisis teks agar lebih menarik dan mudah dipahami.

III. METODE

Metode pencarian jurnal yang digunakan dalam penelitian ini adalah dengan memanfaatkan *software Publish or Perish*. Kata kunci yang digunakan untuk mencari jurnal ini adalah "*Code Smell Detection*" dengan rentang waktu dari tahun 2021 hingga tahun 2024. Jurnal yang terkumpul sebanyak 200 jurnal terindeks Scopus dan sudah disitasi sebanyak 1.521 kali, dan rata-rata sitasi per tahun adalah 507 kali. Namun, dari 200 jurnal yang terkumpul, hanya 69 jurnal yang tersedia secara *open access*. Dan dari 69 jurnal yang *open access* tersebut akan dilakukan berbagai macam analisa yaitu analisa asal negara publikasi, analisa jumlah publikasi per tahun, analisa jenis publikasi, analisa penulis yang sering membuat jurnal dengan topik yang sama, analisa seberapa banyak jurnal tersebut disitasi oleh jurnal lain dan analisa kata kunci yang banyak dipakai. Tahapan penyeleksian dan analisa jurnal yang dilakukan digambarkan pada Gambar 1.

Analisa jumlah publikasi per tahun, analisa jenis publikasi dan analisa banyaknya sitasi dibantu oleh software *Publish or Perish*. Kemudian analisa penulis akan menggunakan *software VOSviewer*. Dan penggunaan *software NVivo* 15 untuk analisa kata kunci beserta pembuatan *word cloud*, dilanjutkan dengan mencari hubungan antar kata kunci dengan *VOSviewer*.



Gambar 1 Tahapan Penyeleksian beserta Analisa Jurnal

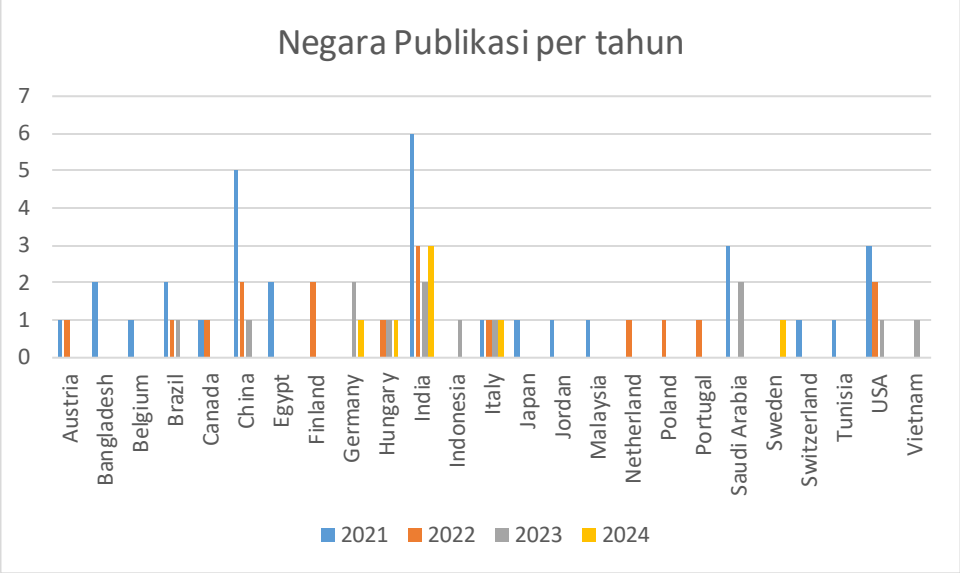
IV. HASIL

Tabel 1 menjelaskan bahwa institusi yang paling banyak melakukan penelitian dengan topik deteksi *code smell* adalah dari India, dimana institusi tersebut menghasilkan jurnal sebanyak 14 jurnal dari tahun 2021-2024. Kemudian disusul oleh institusi yang berasal dari China dengan menghasilkan 8 jurnal. Lalu ada institusi dari negara Amerika Serikat yang mengeluarkan 6 jurnal, Saudi Arabia sebanyak 5 jurnal, dan untuk Brazil dan Italia menghasilkan masing-masing 4 jurnal. Negara Jerman dan Hungaria dengan masing-masing 3 jurnal. Lalu negara Austria, Bangladesh, Kanada, Mesir, dan Finlandia membuat 2 jurnal. Sisanya seperti negara Belgia, Indonesia, Jepang, Yordania, Malaysia, Belanda, Polandia, Portugal, Swedia, Swiss, Tunisia dan Vietnam menghasilkan 1 jurnal. Dengan adanya data tersebut, dapat disimpulkan bahwa institusi yang produktif dalam melakukan penelitian tentang *code smell* adalah berasal dari India.

TABEL 1
INSTITUSI NEGARA YANG MENGHASILKAN JURNAL

No	Negara	2021	2022	2023	2024	Total
1	India	6	3	2	3	14
2	RRC	5	2	1	0	8
3	Amerika Serikat	3	2	1	0	6
4	Saudi Arabia	3	0	2	0	5
5	Brazil	2	1	1	0	4
6	Italia	1	1	1	1	4
7	Jerman	0	0	2	1	3
8	Hungaria	0	1	1	1	3
9	Austria	1	1	0	0	2
10	Bangladesh	2	0	0	0	2
11	Kanada	1	1	0	0	2
12	Mesir	2	0	0	0	2
13	Finlandia	0	2	0	0	2
14	Belgia	1	0	0	0	1
15	Indonesia	0	0	1	0	1
16	Jepang	1	0	0	0	1
17	Yordania	1	0	0	0	1
18	Malaysia	1	0	0	0	1
19	Belanda	0	1	0	0	1
20	Polandia	0	1	0	0	1
21	Portugal	0	1	0	0	1
22	Swedia	0	0	0	1	1
23	Swiss	1	0	0	0	1
24	Tunisia	1	0	0	0	1
25	Vietnam	0	0	1	0	1

Selama rentang tahun 2021 hingga 2024, tiap negara dipisah berdasarkan tahun pada Gambar 2. Dapat dilihat institusi negara dari India dan RRC memiliki jurnal paling tinggi selama tahun 2021.



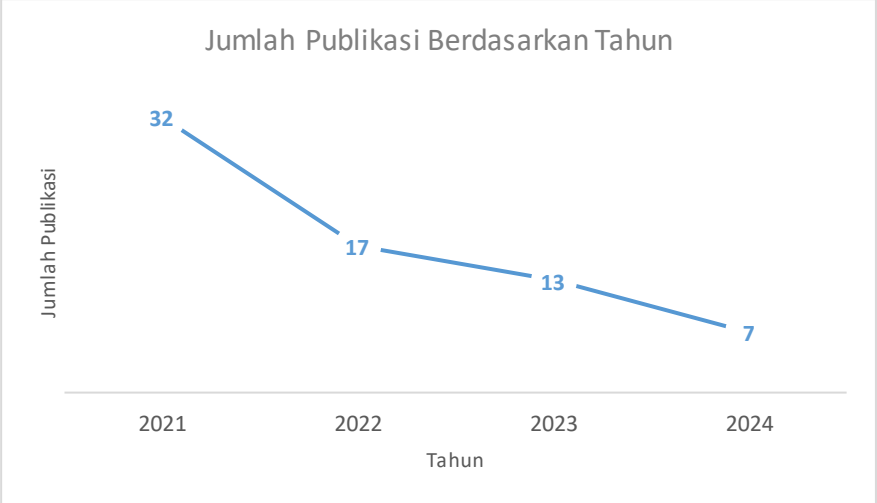
Gambar 2 Publikasi Penelitian dari Berbagai Negara

Analisa jenis publikasi dibagi menjadi 3 jenis, yaitu artikel, *conference paper* dan *review*. Dari 69 jurnal *open access* yang berupa artikel sebanyak 40 jurnal, 24 jurnal berupa *conference paper* dan sisanya sejumlah 5 merupakan publikasi *review*, seperti yang ada pada Tabel 2.

TABEL 2
JENIS PUBLIKASI

No	Jenis Publikasi	Total
1	Artikel	40
2	*Conference Paper*	24
3	*Review*	5

Pada Gambar 3, dapat dilihat bahwa penelitian mengenai deteksi *code smell* mengalami penurunan setelah tahun 2021. Pada tahun 2021, jumlah penelitian yang dilakukan masih sebanyak 32 jurnal. Tetapi pada tahun 2022, tren penelitian ini mulai mengalami kemerosotan sehingga hanya sejumlah 17 jurnal. Kemudian, penurunan tersebut masih terjadi di tahun 2023 yang hanya menghasilkan 13 jurnal. Dan di tahun 2024 juga hanya menghasilkan 7 jurnal.



Gambar 3 Jumlah Publikasi Berdasarkan Tahun

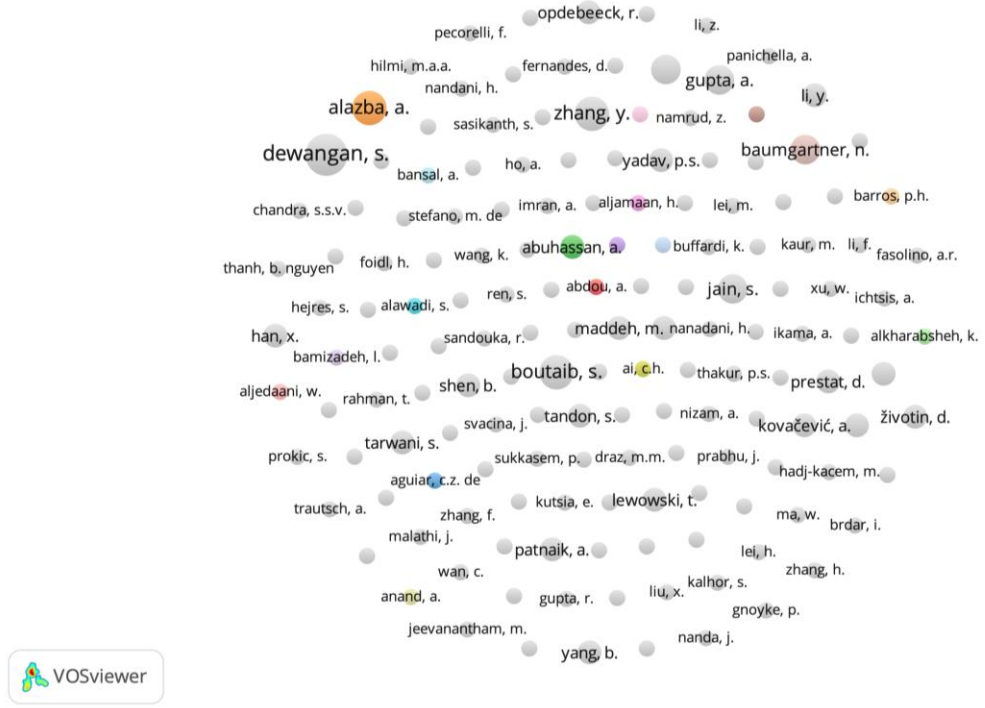
Untuk mencari data penulis yang sering membuat publikasi berkaitan dengan *code smell detection* menggunakan *software VOSviewer*. Didapat sebanyak 7 penulis yang membuat publikasi lebih dari 1 jurnal. Penulis yang paling sering membuat publikasi adalah Nasraldeen Alnor Adam Khleel sebanyak 3 jurnal.

Berikutnya Bo Yang, Mohamed Maddeh, Nils Baumgartner, Pravin Singh Yadav, Seema Dewangan, dan Shivani Jain dengan masing-masing sebanyak 2 jurnal dan dapat dilihat pada Tabel 3.

TABEL 3
PENULIS YANG SERING MEMBUAT PUBLIKASI

No	Penulis	Total jurnal yang dibuat
1	Nasraldeen Alnor Adam Khleel	3
2	Bo Yang	2
3	Mohamed Maddeh	2
4	Nils Baumgartner	2
5	Pravin Singh Yadav	2
6	Seema Dewangan	2
7	Shivani Jain	2

Gambar 4 divisualisasikan secara *network* tentang semua penulis oleh *software VOSviewer*. Hasil menunjukkan tidak ada keterkaitan hubungan antar institusi satu sama lain atau adanya kolaborasi antar penulis dalam membuat publikasi secara bersama-sama.



Gambar 4 Visualisasi *Network* Kolaborasi Antar Penulis

Selanjutnya analisa banyaknya sebuah jurnal disitasi oleh jurnal lainnya, dapat dilihat pada Tabel 4. Tabel 4 menampilkan 5 jurnal yang paling sering dipakai dalam daftar pustaka pada jurnal peneliti lain.

TABEL 4
JURNAL YANG SERING DIPAKAI

No	Nama Jurnal	Tahun	Banyaknya sitasi	Banyaknya sitasi per tahun
1	*Deep learning based code smell detection* [27]	2021	63	21
2	*Code smell detection by deep direct-learning and transfer-learning* [28]	2021	61	20,33
3	*Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?* [29]	2021	50	16,67
4	*Code smell detection using feature selection and stacking ensemble: An empirical investigation* [30]	2021	42	14
5	*Improving performance with hybrid feature selection and ensemble machine learning techniques for code smell detection* [31]	2021	39	13

Software NVivo 15 digunakan dalam penelitian ini untuk mengidentifikasikan kata kunci yang paling sering muncul dalam judul dan abstrak penelitian. Pencarian yang dipakai memakai filter minimum panjang kata 3 huruf dan 1000 daftar kata kunci. Kata kunci yang paling sering dipakai dalam 69 jurnal ini adalah kata *code*

dengan jumlah 9.199 dan bobot presentase sebesar 1,49%, diikuti kata *smells* dengan jumlah 5.447 dan bobot presentase 0,88%. Lalu kata kunci tertinggi ketiga adalah kata *software* dengan jumlah 5.115 dan bobot presentase 0,84%. Kata kunci berikutnya yaitu ada kata *detection*, *data*, *test*, *class*, *method*, *learning* dan *study*. Pada tabel 5, menampilkan tentang 10 kata kunci tertinggi yang sering dipakai beserta panjang kata (*length*), jumlah (*count*), dan bobot presentase.

TABEL 5
KATA KUNCI YANG SERING DIPAKAI

No	Kata Kunci	Length	Count	Bobot Presentase (%)
1	Code	4	9.199	1,49
2	Smells	6	5.447	0,88
3	Software	8	5.155	0,84
4	Detection	9	4.417	0,72
5	Data	4	3.694	0,60
6	Test	4	3.276	0,53
7	Class	5	3.253	0,53
8	Method	6	2.885	0,47
9	Learning	8	2.772	0,45
10	Study	5	2.271	0,37

Pada Gambar 5, kata kunci yang telah ditemukan divisualisasikan dalam bentuk *word cloud*. Kata kunci *code*, *smells* dan *software* memiliki ukuran yang besar dan ditempatkan di tengah grafik dikarenakan memiliki bobot presentase yang lebih besar di antara kata kunci-kata kunci yang lainnya.

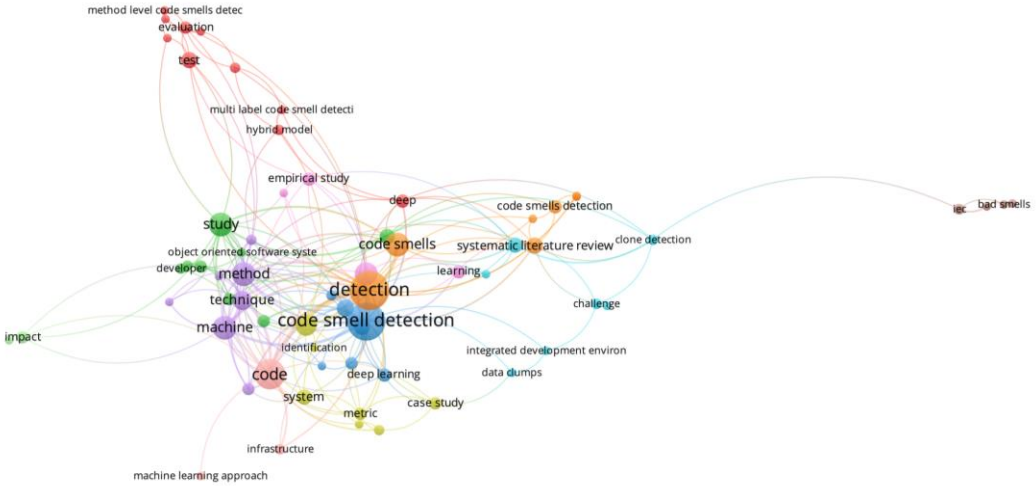
Gambar 5 *Word Cloud* Berdasarkan Kata Kunci

Kata kunci *code* sering dipakai di jurnal yang ditulis oleh Zhenhao Li [32] sebanyak 456 kata. Lalu untuk kata kunci *smells* sering digunakan di jurnal yang dibuat oleh Fabio Palomba [29] dengan *reference* sebanyak 284 kali dan kata kunci tertinggi ketiga yaitu *software* sering dipakai oleh jurnal dengan penulis yang bernama Amjad AbuHassan [33] sebanyak 374 kata. Berikut pada Tabel 6 ditampilkan kata kunci beserta nama jurnal, banyaknya kata yang dipakai (*reference*) dan *coverage* dalam presentase.

TABEL 6
KATA KUNCI, NAMA JURNAL, *REFERENCE*, DAN *COVERAGE*

No	Kata Kunci	Nama jurnal	References	Coverage (%)
1	Code	*Studying Duplicate Logging Statements and Their Relationships With Code Clones* [32]	456	0,84
2	Smells	*Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?* [29]	283	0,62
3	Software	*Software smell detection techniques: A systematic literature review* [33]	374	1,00

Gambar 6 memperlihatkan sebuah visualisasi *network* dalam mencari keterkaitan kata kunci berdasarkan judul pada *software VOSviewer*. Terdapat 66 kata dan diklasifikasikan ke dalam 11 cluster yang berbeda. Kata-kata yang memiliki skala besar seperti *code smell detection, detection, code, machine* dan *method* memiliki keterkaitan satu sama lain antar semua jurnal.



Gambar 6 Visualisasi *Network* dari Kata Kunci

V. PEMBAHASAN

Penelitian ini dapat memberikan pemahaman mendalam mengenai kontribusi institusi dalam domain deteksi *code smell*. Dari hasil analisa, tergambar dengan jelas bahwa institusi yang berasal dari India yang paling banyak mempublikasikan penelitian tentang deteksi *code smell* dengan total 14 jurnal. Kemudian disusul oleh institusi yang berasal dari RRC sebanyak 8 jurnal dan Amerika Serikat dengan 6 jurnal. Walaupun penelitian mengenai deteksi *code smell* banyak dilakukan pada tahun 2021, tetapi tren mengenai domain ini mengalami penurunan pada tahun 2022 sampai tahun 2024. Jenis publikasi yang paling banyak adalah dalam bentuk artikel sebanyak 40 jurnal, lalu 24 jurnal adalah *conference paper*, dan 5 publikasi *review*.

Penelitian ini juga dapat memberikan wawasan bagi pengembang perangkat lunak dalam mengimplementasikan sistem deteksi *code smell* dalam proyek besar dan kompleks, di mana *review* kode secara manual menjadi tidak efisien untuk dilakukan.

Penulis yang sering membuat publikasi tentang deteksi code smell adalah Nasraldeen Alnor Adam Khleel sebanyak 3 jurnal. Dan dari semua penulis tidak memiliki keterkaitan sama sekali untuk membuat kolaborasi sebuah publikasi secara bersama-sama. Hasil analisa jurnal yang paling banyak dipakai sebagai referensi oleh jurnal lain adalah yang berjudul *Deep learning based code smell detection* sebanyak 63 kali sitasi. Apabila melihat *word cloud* yang terbentuk, terdapat kata *code, smells* dan *software* yang sering dipakai. Melalui VOSviewer, kata *code smell detection, detection, code, machine* dan *method* memiliki keterkaitan satu sama lain antar semua jurnal.

Penelitian ini hanya mencakup data yang diambil dari basis data terbuka Scopus sehingga tidak mencakup seluruh literatur yang relevan terutama artikel yang lebih baru mengenai deteksi *code smell*. Keterbatasan ini berpotensi mengurangi representasi lengkap dari tren penelitian yang ada. Meskipun penelitian ini memberikan wawasan tentang tren dan kontribusi penelitian, pemahaman yang lebih mendalam tentang kualitas penelitian memerlukan analisis lebih lanjut menggunakan metode kualitatif.

VI. KESIMPULAN

Penelitian ini menemukan bahwa institusi dari India mendominasi publikasi di bidang pendeteksian *code smell*. Hal ini mencerminkan kontribusi besar serta menunjukkan tingginya minat dari negara India dalam melakukan penelitian tentang pendeteksian *code smell*. Meskipun ada lonjakan minat penelitian, namun dari penelitian ini ditemukan adanya penurunan publikasi jurnal penelitian mengenai topik ini setelah tahun 2021. Penurunan ini dapat disebabkan beberapa faktor seperti matangnya metode *machine learning* dalam mendeteksi *code smell*, serta peralihan fokus para peneliti ke topik baru yang lebih relevan.

Pada penelitian analisis bibliometrik yang membahas karya-karya ilmiah terkait deteksi *code smell* dengan penerapan pendekatan *machine learning*, diharapkan pemahaman mendalam terhadap literatur ini dapat memberikan pandangan yang lebih jelas terkait tren penelitian maupun kontribusi peneliti atau institusi dalam domain ini. Temuan dari penelitian ini dapat dijadikan sebagai dasar untuk merencanakan penelitian mendatang

terkait deteksi *code smell*. Selain itu, penelitian ini dapat membantu pengembang perangkat lunak untuk memilih dan menerapkan alat deteksi *code* dengan memanfaatkan teknologi *machine learning*.

Saran untuk penelitian di masa depan dapat memperluas penggunaan teknik selain *machine learning* yaitu dengan menggunakan teknik *deep learning* untuk meningkatkan akurasi dan efektivitas dalam mendeteksi *code smell* terutama dalam konteks kode yang besar dan kompleks.

DAFTAR PUSTAKA

[1] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, "Machine learning techniques for code smell detection: A systematic literature review and meta-analysis," *Information and Software Technology*, vol. 108, pp. 115–138, Apr. 2019, doi: 10.1016/j.infsof.2018.12.009.

[2] M. De Stefano, F. Pecorelli, F. Palomba, and A. De Lucia, "Comparing within- and cross-project machine learning algorithms for code smell detection," in *Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution*, Athens Greece: ACM, Aug. 2021, pp. 1–6. doi: 10.1145/3472674.3473978.

[3] S. Dewangan, R. S. Rao, A. Mishra, and M. Gupta, "Code Smell Detection Using Ensemble Machine Learning Algorithms," *Applied Sciences*, vol. 12, no. 20, p. 10321, Oct. 2022, doi: 10.3390/app122010321.

[4] A. Ho, A. M. T. Bui, P. T. Nguyen, and A. Di Salle, "Fusion of deep convolutional and LSTM recurrent neural networks for automated detection of code smells," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, Oulu Finland: ACM, Jun. 2023, pp. 229–234. doi: 10.1145/3593434.3593476.

[5] M. Y. Mhawish and M. Gupta, "Predicting Code Smells and Analysis of Predictions: Using Machine Learning Techniques and Software Metrics," *J. Comput. Sci. Technol.*, vol. 35, no. 6, pp. 1428–1445, Nov. 2020, doi: 10.1007/s11390-020-0323-7.

[6] X. Yu, F. Li, K. Zou, J. Keung, S. Feng, and Y. Xiao, "On the Relative Value of Imbalanced Learning for Code Smell Detection," Jan. 10, 2023. doi: 10.22541/au.167338512.23766841/v1.

[7] T. Sharma and D. Spinellis, "A survey on software smells," *Journal of Systems and Software*, vol. 138, pp. 158–173, Apr. 2018, doi: 10.1016/j.jss.2017.12.034.

[8] P. Kokol, M. Zorman, G. Zlahtic, and B. Žlahti\vc, "Code smells," *ArXiv*, vol. abs/1802.06063, 2018, [Online]. Available: https://api.semanticscholar.org/CorpusID:3377287

[9] P. Kokol and M. Kokol, "Code smells: A Synthetic Narrative Review".

[10] K. Beck, *Kent Beck's guide to better Smalltalk: a sorted collection*, vol. 14. Cambridge University Press, 1999.

[11] M. Fowler, "Refactoring.," presented at the TOOLS (34), 2000, p. 437.

[12] T. Lewowski and L. Madeyski, "How far are we from reproducible research on code smell detection? A systematic literature review," *Information and Software Technology*, vol. 144, p. 106783, Apr. 2022, doi: 10.1016/j.infsof.2021.106783.

[13] J. P. dos Reis, F. B. e Abreu, G. de F. Carneiro, and C. Anslow, "Code smells detection and visualization: A systematic literature review," *Arch Computat Methods Eng*, vol. 29, no. 1, pp. 47–94, Jan. 2022, doi: 10.1007/s11831-021-09566-x.

[14] S. Dewangan, R. S. Rao, A. Mishra, and M. Gupta, "A Novel Approach for Code Smell Detection: An Empirical Study," *IEEE Access*, vol. 9, pp. 162869–162883, 2021, doi: 10.1109/ACCESS.2021.3133810.

[15] S. Lu, N. Duan, H. Han, D. Guo, S. Hwang, and A. Svyatkovskiy, "ReACC: A Retrieval-Augmented Code Completion Framework," Mar. 15, 2022, *arXiv*: arXiv:2203.07722. Accessed: Nov. 18, 2024. [Online]. Available: http://arxiv.org/abs/2203.07722

[16] D. Životin, "Shotgun Surgery Bad Smell Detection in IEC 61499".

[17] H. Singh and S. I. Hassan, "Effect of SOLID Design Principles on Quality of Software: An Empirical Assessment," vol. 6, no. 4, 2015.

[18] S. Kelly and R. Pohjonen, "Worst Practices for Domain-Specific Modeling," *IEEE Softw.*, vol. 26, no. 4, pp. 22–29, Jul. 2009, doi: 10.1109/MS.2009.109.

[19] A. Bing, "KISS: Keep It Simple, Stupid!".

[20] "Code Quality Tool & Secure Analysis with SonarQube." Accessed: Nov. 18, 2024. [Online]. Available: https://www.sonarsource.com/products/sonarqube/

[21] R. N. Broadus, "Toward a definition of 'bibliometrics,'" *Scientometrics*, vol. 12, no. 5–6, pp. 373–379, Nov. 1987, doi: 10.1007/BF02016680.

[22] M. S. Kavic and R. M. Satava, "Scientific Literature and Evaluation Metrics: Impact Factor, Usage Metrics, and Altmetrics," *JSLS*, vol. 25, no. 3, p. e2021.00010, 2021, doi: 10.4293/JSLS.2021.00010.

[23] A.-W. Harzing -, "Publish or Perish," Harzing.com. Accessed: Nov. 18, 2024. [Online]. Available: https://harzing.com/resources/publish-or-perish

[24] "NVivo Leading Qualitative Data Analysis Software (QDAS) by Lumivero," Lumivero. Accessed: Nov. 18, 2024. [Online]. Available: https://lumivero.com/products/nvivo/

[25] N. J. van Eck and L. Waltman, "VOSviewer Manual".
[26] J. Wang, J. Zhao, S. Guo, C. North, and N. Ramakrishnan, "ReCloud: Semantics-Based Word Cloud Visualization of User Reviews," in *Graphics Interface 2014*, 1st ed., P. G. Kry and A. Bunt, Eds., A K Peters/CRC Press, 2020, pp. 151–158. doi: 10.1201/9781003059325-20.
[27] H. Liu, J. Jin, Z. Xu, Y. Bu, Y. Zou, and L. Zhang, "Deep Learning Based Code Smell Detection," *IIEEE Trans. Software Eng.*, pp. 1–1, 2021, doi: 10.1109/TSE.2019.2936376.
[28] T. Sharma, V. Efstathiou, P. Louridas, and D. Spinellis, "Code smell detection by deep direct-learning and transfer-learning," *Journal of Systems and Software*, vol. 176, p. 110936, Jun. 2021, doi: 10.1016/j.jss.2021.110936.
[29] F. Palomba, D. Andrew Tamburri, F. Arcelli Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik, "Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?," *IIEEE Trans. Software Eng.*, vol. 47, no. 1, pp. 108–129, Jan. 2021, doi: 10.1109/TSE.2018.2883603.
[30] A. Alazba and H. Aljamaan, "Code smell detection using feature selection and stacking ensemble: An empirical investigation," *Information and Software Technology*, vol. 138, p. 106648, Oct. 2021, doi: 10.1016/j.infsof.2021.106648.
[31] S. Jain and A. Saha, "Improving performance with hybrid feature selection and ensemble machine learning techniques for code smell detection," *Science of Computer Programming*, vol. 212, p. 102713, Dec. 2021, doi: 10.1016/j.scico.2021.102713.
[32] Z. Li and W. Shang, "Studying Duplicate Logging Statements and Their Relationships with Code Clones".
[33] A. AbuHassan, M. Alshayeb, and L. Ghouti, "Software smell detection techniques: A systematic literature review," *J Software Evolu Process*, vol. 33, no. 3, p. e2320, Mar. 2021, doi: 10.1002/smr.2320.