

Reviving JPlag

**5 Jahre Open-Source-Modernisierung des weltweit führenden
Code-Plagiatsdetektors aus Karlsruhe**

Timur Sağlam und Sebastian Hahner



STELL DIR VOR...

JPlag is a system that finds similarities among multiple sets of source code files. This way it can detect software plagiarism. It does not merely compare bytes of text, but is aware of programming language syntax and program structure and hence is robust against many kinds of attempts to disguise similarities between plagiarized files. JPlag currently supports Java, C#, C, C++, and natural language text.

JPlag is typically used to detect and thus discourage the unallowed copying of student exercise programs in programming education. But in principle it can also be used to detect stolen software parts among large amounts of source text or modules that have been duplicated (and only slightly modified). JPlag has already played a part in several intellectual property cases where it has been successfully used by expert witnesses.

JPlag has a powerful graphical interface for presenting its results. See our [example](#).

Matches for 132207 & 792145	132207 (93%)	792145 (93%)	Tokens
93%	Jumpbox.java(33-177)	Jumpbox.java(9-154)	143
INDEX - HELP	Jumpbox.java(194-214)	Jumpbox.java(168-198)	
	Jumpbox.java(216-334)	Jumpbox.java(200-327)	
	Jumpbox.java(345-354)	Jumpbox.java(337-352)	
	Jumpbox.java(391-443)	Jumpbox.java(374-426)	

```

public void writeIndexBegin(HTMLFile f, String title) {
    writeHTMLHeader(f, title);
    f.println("<BODY BGCOLOR=#ffffff LINK");
    f.println("<TABLE ALIGN=center CELLPAD");
    f.println("<TR VALIGN=middle ALIGN=center");
    f.println("<TD>Hi>BIG" + title + "

    public static void usage() {
        System.out.print(Program.name_Long
            + ", Copyright (c) 2004-2017 KIT - IPD Tichy
            + "Usage: JPlag [ options ] <root-dir> [-c <code>]
            + " <root-dir>          The root-directory tha
            + "options are:\n"

        if (program.get_title() != null) {
            f.println("<TR BGCOLOR=#aaaaaaff>
                + "</BIG></BIG><TD><BIG>
            if (program.get_original_dir() != null)
                f.println("<TR BGCOLOR=#aaaaaaff VALIGN=top><TD> + "<BIG> + msg.getString("Report.Directory")
                    + "< </BIG></BIG><TD><BIG><CODE> + program.get_original_dir()
                    + (program.get_sub_dir() == null ? "" : File.separator + "" + File.separator + program.get_sub_dir())
                    + "< </CODE></BIG></TD></TR>");
        } else {
            if (this.program.get_original_dir() == null)
                f.println("<TR BGCOLOR=#aaaaaaff VALIGN=top><TD> + "<BIG><BIG> + msg.getString("Report.Directory")
                    + "< </BIG></BIG><TD><BIG><BIG><CODE> + msg.getString("Report.Not available") + "< </CODE></BIG></BIG></TD></TR>");
        }
    }
}

```

```
public static void usage() {
    System.out.print(Program.name_Long
        + ", Copyright (c) 2004-2017 KIT - IPD Tichy, Guido Malpohl, and others.\n"
        + "Usage: JPlag [ options ] <root-dir> [-c file1 file2 ...]\n"
        + "    <root-dir>          The root-directory that contains all submissions.\n\n"
        + "options are:\n"
```

```
Submission A, B, tmp;
if (subA.struct.size() > subB.struct.size()) {
    A = subB; B = subA;
} else {
    A = subB; B = subA;
}
```

```
case 's': // hidden Option
    this.language = new jplag.javax.Language(null); // WARNING!!!!!! BOMB
    this.min_token_match = this.language.min_token_match();
    this.suffixes = this.language.suffixes();
    is.verbose_quiet = true;
    is.exp = true;
    break;
```

```

inner:    for (int i = 1; i <= elemsB[0]; i++) { // elemsB[0] contains the length of the Array
            int y = elemsB[i];
            if (B[y].marked || maxmatch > lengthB - y) continue;

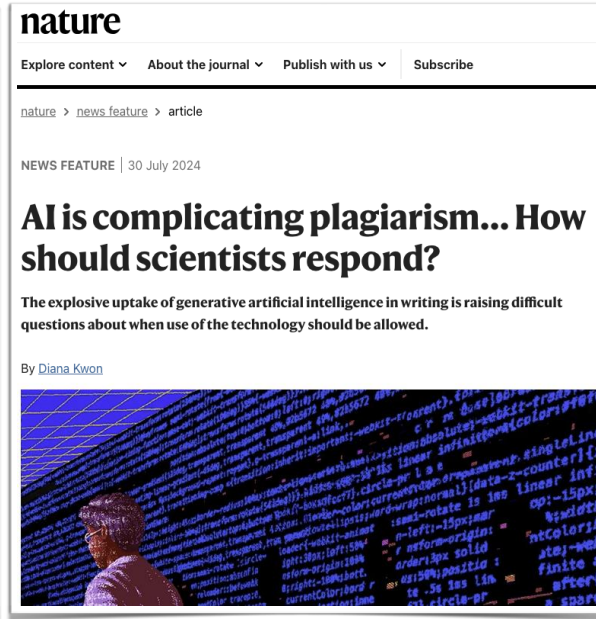
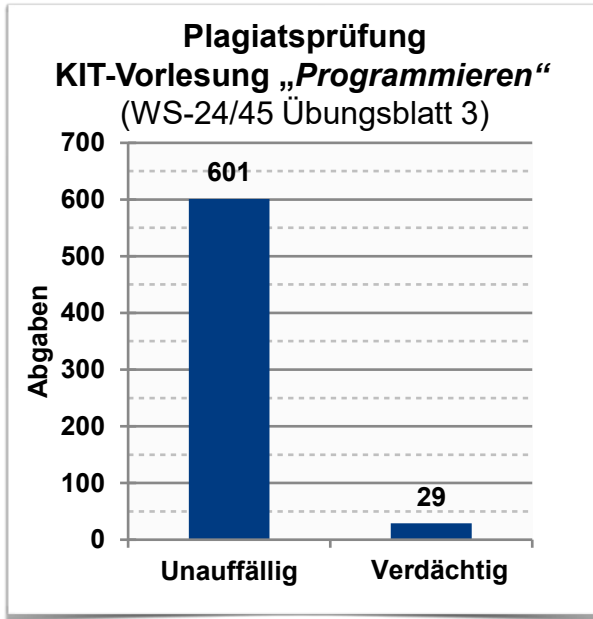
            int j,hx,hy;
            for (j = maxmatch - 1; j >= 0; j--) { // begins comparison from behind
                if (A[hx = x + j].type != B[hy = y + j].type || A[hx].marked || B[hy].marked)
                    continue inner;
            }
            // expand match
            j = maxmatch;
            while(A[hx = x + j].type == B[hy = y + j].type && !A[hx].marked && !B[hy].marked)
                j++;

            if (j != maxmatch) { // new biggest match? -> delete current smaller

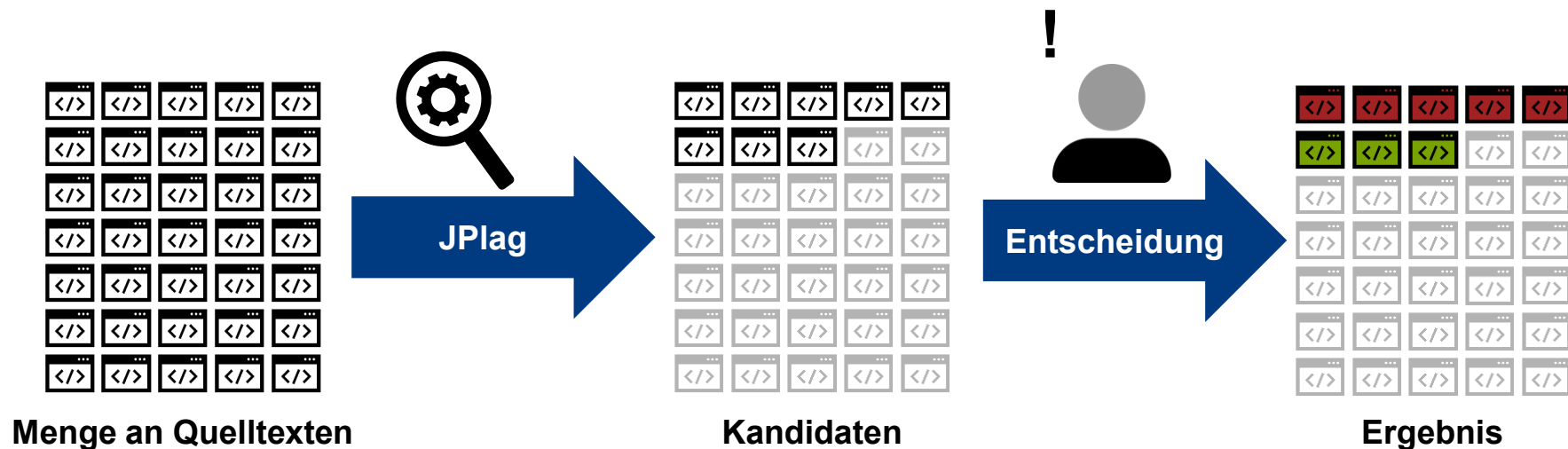
```

```
public void create_hashes(Structure s, int hashLength, boolean makeTable) {
    // Hier wird die obere Grenze der Hash-Laenge festgelegt.
    // Sie ist bestimmt durch die Bitzahl des 'int' Datentyps und der Anzahl
    // der Token.
    if (hashLength < 1) hashLength = 1;
    hashLength = (hashLength < 26 ? hashLength : 25);
}
```

Plagiarismus in der Informatikbildung



TL;DR: JPlag



Herausforderung Software-Plagiarismus

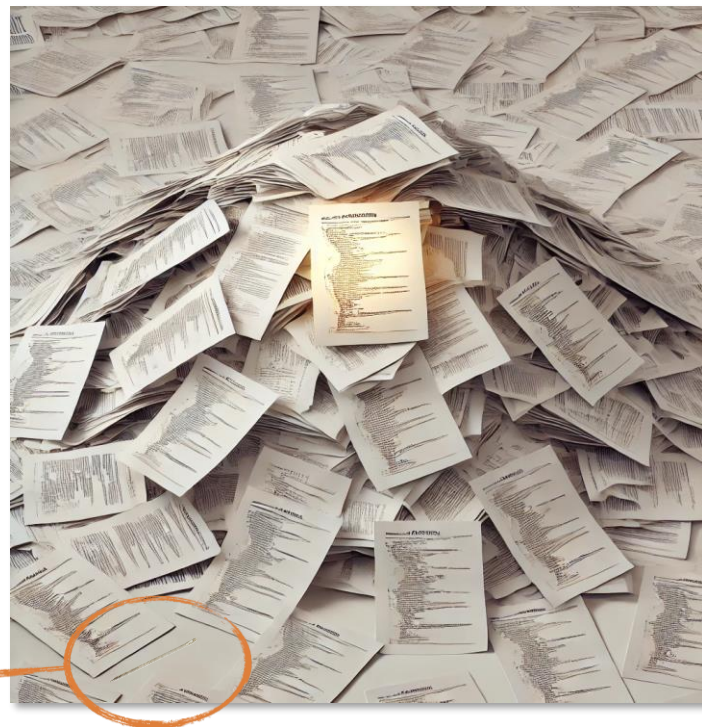
Es ist unbekannt, ...

- ... ob überhaupt Plagiate vorliegen.
- ... welche Programmabschnitte übereinstimmen.

Plagiatserkennung ist ein **Skalierungsproblem!**

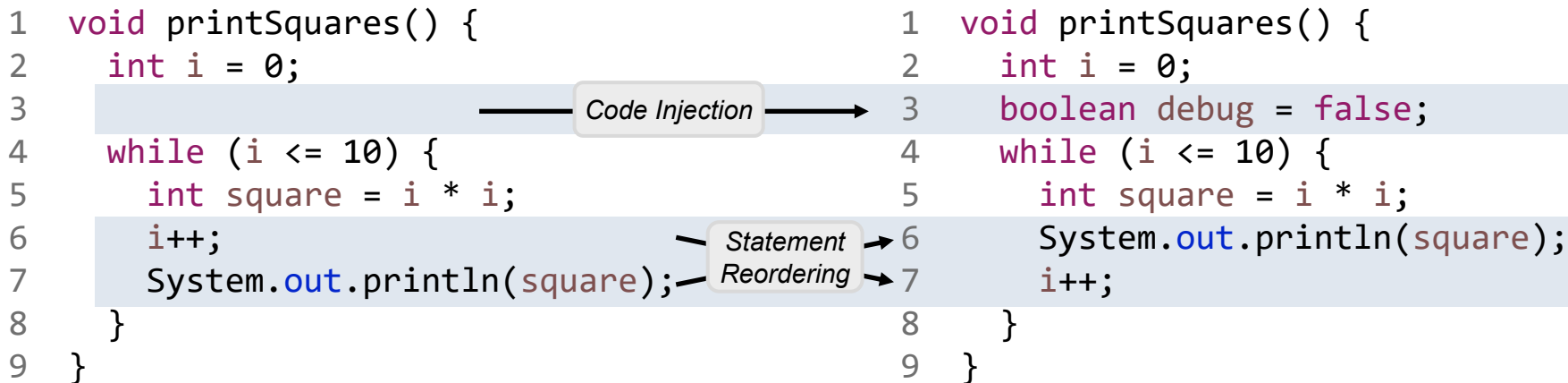
- 30 Studierende $\Rightarrow$ 435 paarweise Vergleiche
- 200 Studierende $\Rightarrow$ 19.900 paarweise Vergleiche
- 630 Studierende $\Rightarrow$ 198.135 paarweise Vergleiche

*Suche nach
der Nadel im
Heuhaufen*



Obfuscation Attacks

Welche strukturellen Änderungen erschweren die Erkennung?

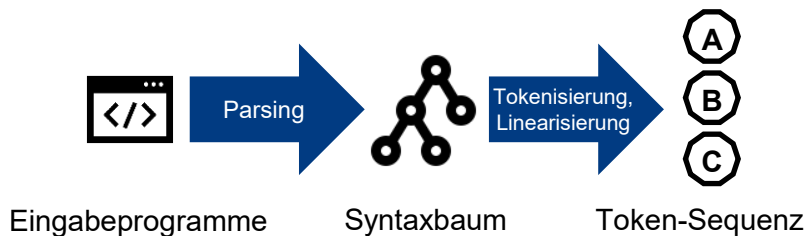


Semantisch neutrale Transformation

- Einfügen von toten Anweisungen
- Umordnung zweier unabhängiger Anweisungen

⇒ leicht automatisierbar

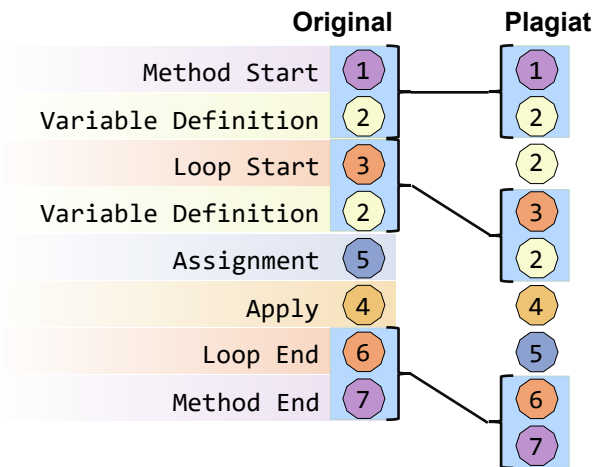
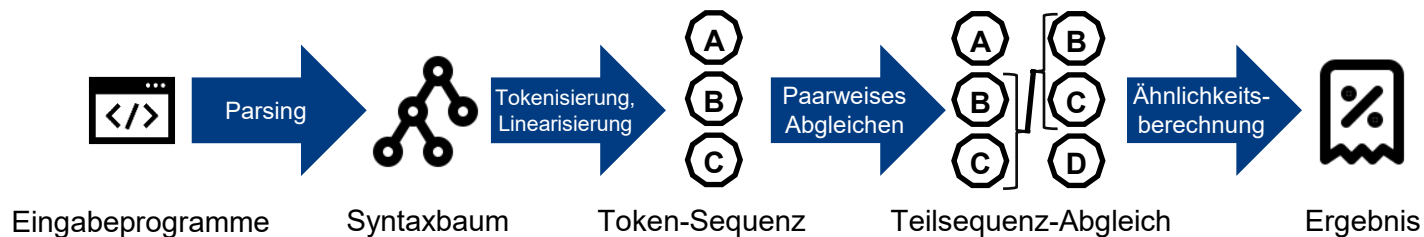
Wie funktioniert JPlag?



Token sind eine *Abstraktion* von Programmcode:

1	<code>void</code>		Code-Plagiatsdetektoren vergleichen die Programmstruktur.	
2	<code>i</code>			
3	<code>boolean debug = false;</code>			(2) Variable Definition
4	<code>while (i <= 10) {</code>			(3) Loop Start
5	<code>int square = i * i;</code>			(2) Variable Definition
6	<code>System.out.println(square);</code>			(4) Apply
7	<code>i++;</code>			(5) Assignment
8	<code>}</code>			(6) Loop End
9	<code>}</code>			(7) Method End

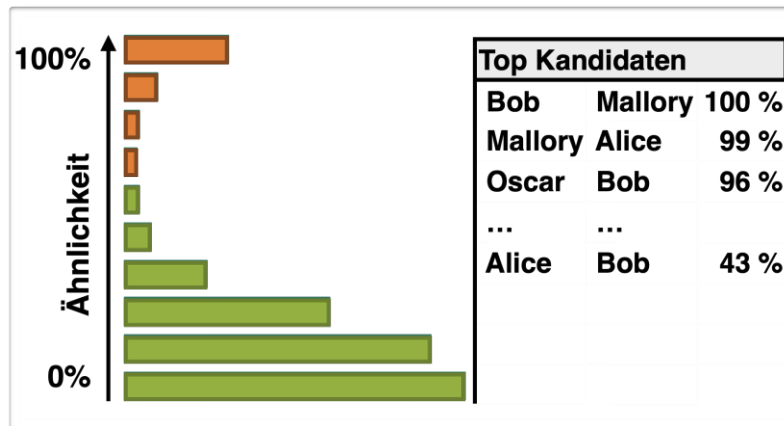
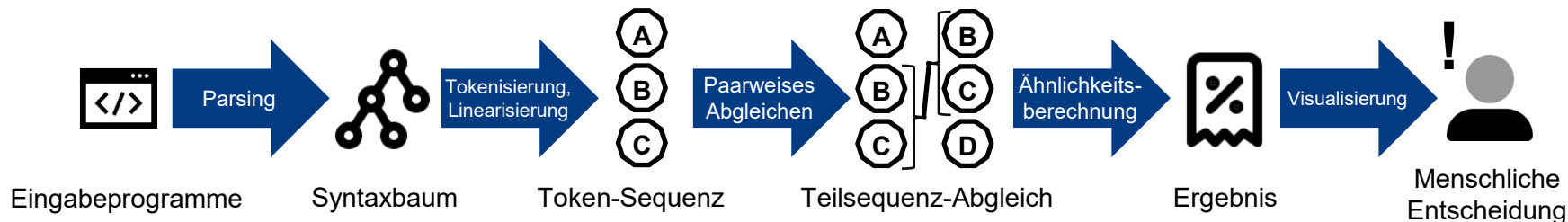
Wie funktioniert JPlag?



⇒ 71 % Ähnlichkeit

(Vereinfachte Darstellung)

Wie funktioniert JPlag?



! Ein Plagiatsdetektor ermittelt nur verdächtige Kandidaten.

What is JPlag

JPlag is a system that finds similarities among multiple sets of source code files. This way it can detect software plagiarism. It does not merely compare bytes of text, but is aware of programming language syntax and program structure and hence is robust against many kinds of attempts to disguise similarities between plagiarized files. JPlag currently supports Java, C#, C, C++, and natural language text.

JPlag is typically used to detect and thus discourage the unallowed copying of student exercise programs in programming education. But in principle it can also be used to detect stolen software parts among large amounts of source text or modules that have been duplicated (and only slightly modified). JPlag has already played a part in several intellectual property cases where it has been successfully used by expert witnesses.

JPlag has a powerful graphical interface for presenting its results. See our [example](#).

Matches for 132207 & 792145

93%

INDEX - HELP

132207 (93%)	792145 (93%)	Tokens
Jumpbox.java(33-177)	Jumpbox.java(9-154)	143
Jumpbox.java(184-214)	Jumpbox.java(168-198)	
Jumpbox.java(216-343)	Jumpbox.java(200-327)	
Jumpbox.java(345-354)	Jumpbox.java(337-352)	
Jumpbox.java(391-443)	Jumpbox.java(374-428)	

```

// System.err.println("paint()");
// Use update() to display the offscreen buffer.
update(g);

/**
 * Update Canvas
 */
void updateCanvas ()
{
    offDimension = dim;
    offImage = createImage(dim.width, dim.height);
    offGraphics = offImage.getGraphics();
    offGraphics.setColor(Color.white);
    offGraphics.fillRect(0, 0, dim.width, dim.height);
    offGraphics.setColor(Color.black);
    offGraphics.drawRect(0, 0, dim.width, dim.height);
    offGraphics.drawRect(0, 0, dim.width, 0);
    drawRTBBoxes();
    drawJumpBox();
}

/**
 * Repaints canvas if it was modified.
 */
synchronized public void update (Graphics g) {
    // System.err.println("update()");

    Dimension dim = getSize();

    // Is the offscreen buffer still valid?
    if ((offGraphics == null)
        || (dim.width != offDimension.width)
        || (dim.height != offDimension.height)) {
        // Repaint it
        updateCanvas ();
    }

    // Copy the offscreen buffer into the game area
    g.drawImage(offImage, 0, 0, this);
}

/**
 * Handle mouse drags.
 */
public void mouseDragged(MouseEvent e) {
    mouseMoved(e);
}

```

```

public void writeIndexBegin(HTMLFile f, String title) {
    writeHTMLHeader(f, title);
    f.println("<BODY BGCOLOR=#ffffff LINK
    f.println("<TABLE ALIGN=center CELLP
    f.println("<TR VALIGN=middle ALIGN=ce
    f.println("<TD><H1><BIG>" + title + "

    if (program.getTitle() != null) {
        f.println("<TR BGCOLOR=#aaaaafff
        + "<BIG></BIG><TD><BIG>

    if (program.getOriginalDir() != null)
        f.println("<TR BGCOLOR=#aaaaafff VALIGN=top><TD>" + "<BIG>" + msg.getString("Report.Directory")
        + "<BIG></BIG><TD><BIG><CODE>" + program.getOriginalDir()
        + (program.getSubDir() == null ? "" : File.separator + " " + File.separator + program.getSubDir())
        + "<CODE></BIG></TD></TR>");
    } else {
        if (this.program.getOriginalDir() == null)
            f.println("<TR BGCOLOR=#aaaaafff VALIGN=top><TD>" + "<BIG><BIG>" + msg.getString("Report.Directory")
            + "<BIG></BIG><TD><BIG><BIG><CODE>" + msg.getString("Report.Not_available") + "<CODE></BIG></BIG><TD></TR>");
    }
}

```

```

public static void usage() {
    System.out.print(Program.name_Long
        + ", Copyright (c) 2004-2017 KIT - IPD Tichy, Guido Malpohl, and others.\n"
        + "Usage: JPlag [ options ] <root-dir> [-c file1 file2 ...]\n"
        + "    <root-dir>      The root-directory that contains all submissions.\n\n"
        + "options are:\n"

```

```

case 's': // hidden Option
    this.language = new jplag.javafx.Language(null); // WARNING!!!!BOMB
    this.min_token_match = this.language.min_token_match();
    this.suffixes = this.language.suffixes();
    this.verbose_quiet = true;
    this.exp = true;
    break;

```

```

inner:
    for (int i = 1; i <= elemsB[0]; i++) { // elemsB[0] contains the length of the Array
        int y = elemsB[i];
        if (B[y].marked || maxmatch > lengthB - y) continue;

        int j, hx, hy;
        for (j = maxmatch - 1; j >= 0; j--) { // begins comparison from behind
            if (A[hx = x + j].type != B[hy = y + j].type || A[hx].marked || B[hy].marked)
                continue inner;
        }
        // expand match
        j = maxmatch;
        while (A[hx = x + j].type == B[hy = y + j].type && !A[hx].marked && !B[hy].marked)
            j++;

        if (j != maxmatch) { // new biggest match? -> delete current smaller
            // delete current smaller
        }
    }

```

```

public void create_hashes(Structure s, int hashLength, boolean makeTable) {
    // Hier wird die obere Grenze der Hash-Laenge festgelegt.
    // Sie ist bestimmt durch die Bitzahl des 'int' Datentyps und der Anzahl
    // der Token.
    if (hashLength < 1) hashLength = 1;
    hashLength = (hashLength < 26 ? hashLength : 25);
}

```

Ausgangssituation

- JPlag kommt aus den 90ern vom Lehrstuhl Prof. Walter F. Tichy, KIT, Karlsruhe
- Funktionsfähig, aber veraltet, viele Code Smells, Monolith, weder Tests noch CI/CD
- Keine Dokumentation außer Paper, kein Wissen über Code Base vorhanden
- Feature-Druck: Die TU München möchte JPlag in ihre Lernplattform integrieren
- JPlag hat sich in den letzten 20 Jahren weit verbreitet und wird weltweit eingesetzt

IPD - Lehrstuhl em. Programmiersysteme

Lehrstuhl em. Prof. em. Dr. Walter F. Tichy

```
on
jplag.javax.Language(null); // WARNING!!!! BOMB
= this.language.min_token_match();
.language.suffixes();
true;
```

Resubmission to J. of Universal Computer Science, November 28, 2001
<http://www.jucs.org/>

Finding plagiarisms among a set of programs with JPlag

Lutz Prechelt, Guido Malpohl, Michael Philippson
Universität Karlsruhe, Germany
prechelt,malpohl,philipp@ira.uka.de

Jedem Anfang...
KIT-Fakultät für Informatik
Prof. Walter Tichy
grüßt seine Nachf.

Java API for JPlag #89 +5,338 -35,342

Open krusche wants to merge 93 commits into jplag:master from lsintum:develop

World map showing JPlag usage locations (blue pins).

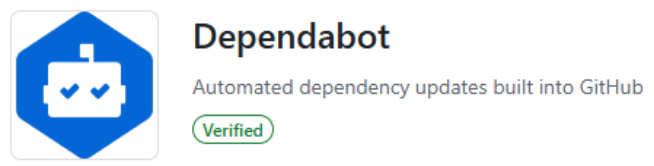
Erste Schritte

Housekeeping

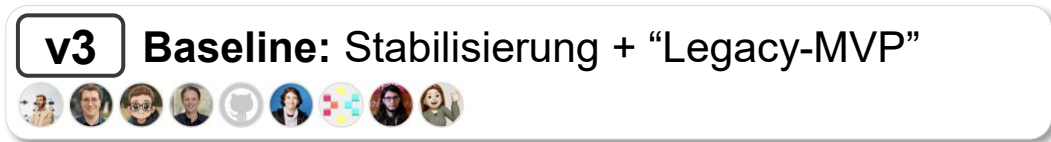
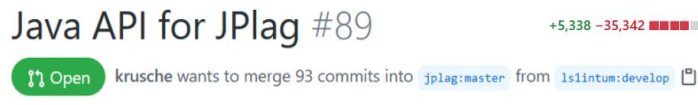
- Erstmal Dependabot hinzufügen ☺
- Minimale Tests
- Code aufräumen, Logging + CLI, toten Code löschen, ...
- Open-Source: Issues reorganisieren und labeln, uralte Pull Requests schließen, ...

Stabilisierungs-Release

- Ein Jahr aufräumen und Kontrolle gewinnen
- Weniger Features als zuvor, aber: Bessere API, CLI, Bugfixes



```
public static void usage() {  
    System.out.print(Program.name_Long  
        + ", Copyright (c) 2004-2017 KIT - IPD Tichy, Guido Malpohl, and others.\n"  
        + "Usage: JPlag [ options ] <root-dir> [-c file1 file2 ...]\n"  
        + " <root-dir>      The root-directory that contains all submissions.\n\n"  
        + "options are:\n"
```



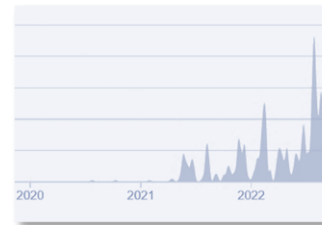
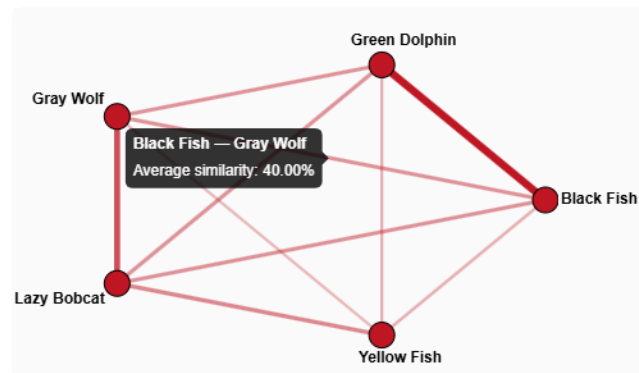
Neue Vision und ein wachsendes Team

Vision für die Zukunft von JPlag

- Vollständig neue, web-basierte UI
- Mehr Features, z.B. Gruppenplagiarismus
- Umfangreiche End-2-End Tests, Qualität+++
- Aktives gepflegtes GitHub-Projekt mit Community, Docs, Releases, ...

Team vergrößern

- Mehr Kollegen begeistern 😊
- Studentische Entwickler, Praktika, ...



v4

Rebuild: Moderne Architektur + Report Viewer





Comparison: Orange Dog - Purple Antelope

d Average Similarity: 85.32%
 Similarity Orange Dog: 86.81%
Similarity Purple Antelope: 83.89%

Matches: pa-java, pa-java: 78
pa-java, pa-java: 37
pa-java, pa-java: 10

File Sorting:
 Alphabetical
Match Coverage
Match Count
Match Size

Files of Orange Dog:

144 total tokens
Collapse All

```

1  Orange Dog/pa.java
2  import java.util.*;
3
4  class util {
5      Scanner kb;
6      util(Scanner kb){this.kb = kb;}
7      int N;
8      int count;
9      boolean isPrime;
10
11      IntegerInteger error = new LinkedIntInteger();
12      IntegerInteger Integer map = new HashMapInteger,Integer();
13      ArrayListIntegerInteger tag1 = new ArrayListInteger,Integer();
14      ArrayListIntegerInteger tag2 = new ArrayListInteger,Integer();
15
16      if(map.containsKey(k)) return map.get(k);
17
18      add(new LinkedIntInteger());
19      IntegerInteger Integer map = new HashMapInteger,Integer();
20      int i = map.size();
21      return map.get(i);
22  }
23
24  void read() {
25      N = kb.nextInt();
26      map.clear();
27      tag1.clear();
28      for(int i = 1; i <= N; i++)
29          add(i);
30      int i = kb.nextInt();
31      int c = find.nextInt();

```

Files of Purple Antelope:

149 total tokens
Collapse All

```

1  Purple Antelope/pa.java
2  import java.util.*;
3
4  class util {
5      Scanner kb;
6      util(Scanner kb){this.kb = kb;}
7      int N;
8      int count;
9      boolean isPrime;
10
11      IntegerInteger error = new LinkedIntInteger();
12      IntegerInteger Integer map = new HashMapInteger,Integer();
13      ArrayListIntegerInteger tag1 = new ArrayListInteger,Integer();
14      ArrayListIntegerInteger tag2 = new ArrayListInteger,Integer();
15
16      if(map.containsKey(k)) return map.get(k);
17
18      add(new LinkedIntInteger());
19      IntegerInteger Integer map = new HashMapInteger,Integer();
20      int i = map.size();
21      return map.get(i);
22  }
23
24  void read() {
25      N = kb.nextInt();
26      map.clear();
27      tag1.clear();
28      for(int i = 1; i <= N; i++)
29          add(i);
30      int i = kb.nextInt();
31      int c = find.nextInt();

```

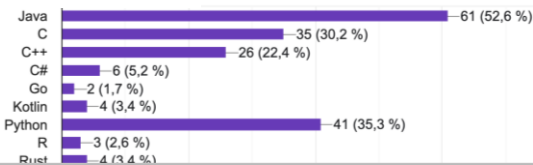

Resilienz gegen Obfuscation Attacks

Neue Angriffsmöglichkeiten

- Oktober 2020: Paper demonstriert Angriff auf JPlag durch Einfügen von Dead Code
- November 2022: ChatGPT wird angekündigt
- ICSE 2024
 - Vorstellung von Abwehrmechanismen
 - Unmittelbar Research-to-Production
- Start der JPlag User Survey

What language modules do you use?

116 Antworten



Detecting Automatic Software Plagiarism via Token Sequence Normalization

Timur Sağlam, Moritz Brödel, Larissa Schmid, Sebastian Hahner
firstname.lastname@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

v5

Expand: Verbesserte Resilienz + UX



TL;DR: 30 Jahre JPlag



1996 ○ Initiale Entwicklung

2015 ○ Umzug auf GitHub

Ende 2020 ○ **Projekt Übernahme**

v3 Baseline: Stabilisierung + “Legacy-MVP”



2022 ○ **v4 Rebuild:** Moderne Architektur + Report Viewer



2023 ○ **v5 Expand:** Verbesserte Resilienz + UX



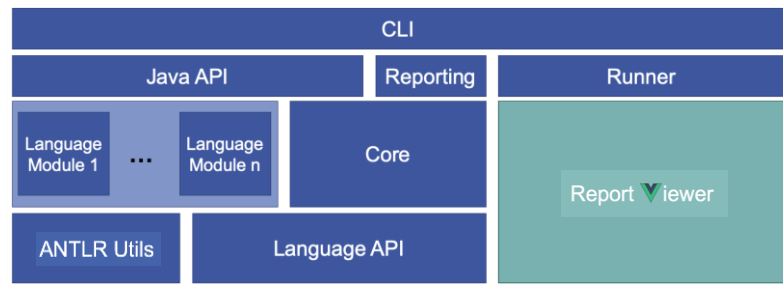
2024 ○ **v6 Polish:** Integrierte Plattform + bessere Analyse



JPlag Heute

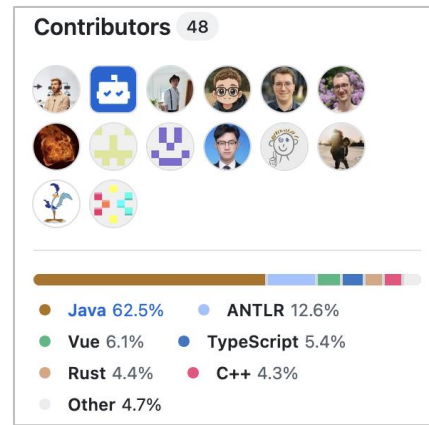
Modernisierte Architektur

- Web-UI in Java-Anwendung integriert
- API für Drittanbieter-Integration
- Einheitliches Sprachmodul-System
- Hohe Performance durch Parallelisierung
- Always-Up-To-Date Web-Demo



Best-Practice Software Engineering

- Umfassende Tests: Unit/Integration/E2E/Language (~84% Coverage)
- Moderne CI/CD-Pipeline (build/lint/scan/test/doc/deploy/release)
- Documentation-as-Code
- Schnelle und verlässliche Release-Prozesse
- Aktive, internationale Open-Source-Community

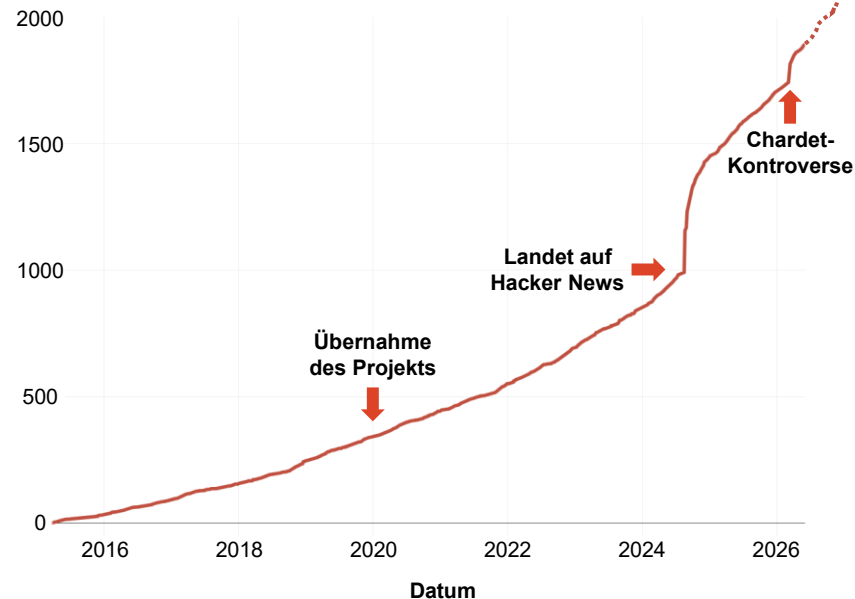


JPlag in Zahlen

Stetiges Wachstum der Community:

Metrik	2020	Heute
★ Stars	~300	~1.900
🔗 Commits	~100	~6.000
🔗 Pull Requests	19	2.560
⚠️ Issues	81	380 (54 offen)
👥 Contributors	12	48

GitHub-Stars



JPlag Morgen

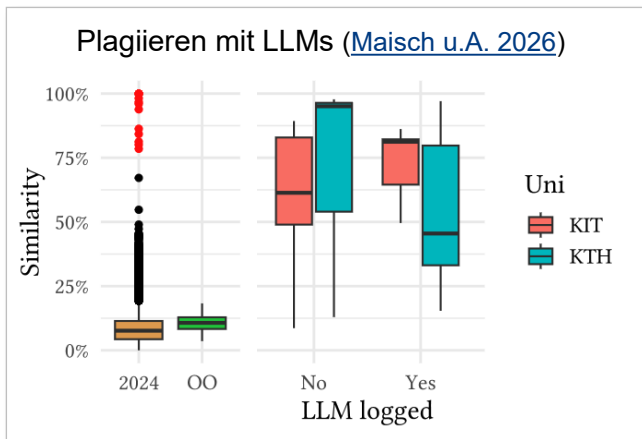
Projekt Lead

- Ab Ende 2025 an nächste Generation übergeben
- Im Hintergrund als Maintainer gelegentlich aktiv

JPlag entwickelt sich weiter

- **Engineering:** Erkennungsqualität und Explainability
- **Forschung:** Plagiarismus im Zeitalter von KI
 - KI-basierte Obfuscation verstehen und verhindern
 - KI-generierter Code: Strukturell ähnlicher als gedacht

 Robin Maisch Project Lead Karlsruhe Institute of Technology ID 0009-0003-4546-9048	 Nils Niehues Maintainer Karlsruhe Institute of Technology ID 0009-0006-4295-7232
 Alexander Vogt Student Developer (Frontend) Karlsruhe Institute of Technology	 Alexander Milster Student Developer (Backend) Karlsruhe Institute of Technology



Zusammenfassung

1996 ○ Initiale Entwicklung

2015 ○ Umzug auf GitHub

Ende 2020 ○ Projekt Übernahme

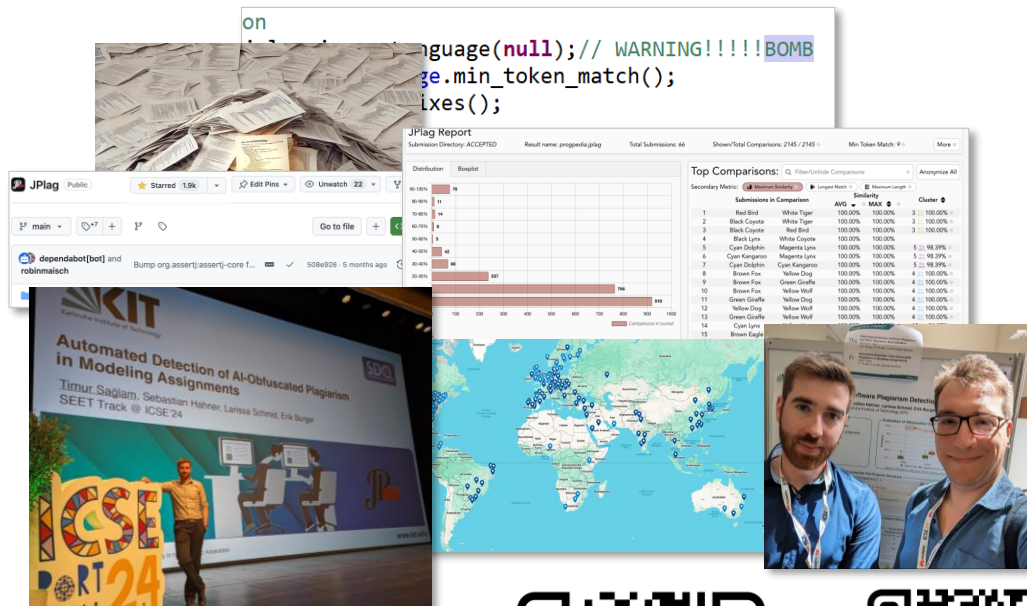
2022 ○ v3 Baseline

2023 ○ v4 Rebuild

2024 ○ v5 Expand

2025 ○ v6 Polish

Ende 2025 ○ Abgabe Project-Lead



Mehr zu JPlag...



Vortrag bewerten...

Weiterführende Informationen

Relevante Links

- JPlag auf GitHub: <https://www.github.com/jplag/JPlag>
- JPlag Online Demo: <https://demo.jplag.de/>
- Helmholtz RSD: <https://helmholtz.software/software/jplag>

Ausgewählte Veröffentlichungen

- [1] R. Maisch, et al., „Same Same But Different: Preventing Refactoring Attacks on Software Plagiarism Detection”, *ICSE*, IEEE/ACM, 2026.
- [2] R. Maisch, “Too Hot To Handle: The Effects of Temperature on AI-Generated Code Used to Cheat in Programming Assignments”, *ICSE-C*, IEEE/ACM, 2026.
- [3] T. Sağlam, et al., “Mitigating Obfuscation Attacks on Software Plagiarism Detectors via Subsequence Merging”, *CSEE&T*, IEEE/ACM, 2025.
- [4] T. Sağlam, et al., “Detecting Automatic Software Plagiarism via Token Sequence Normalization”, *ICSE*, IEEE/ACM, 2024.
- [5] T. Sağlam, et al., “Automated Detection of AI-Obfuscated Plagiarism in Modeling Assignments”, *ICSE-SEET*, IEEE/ACM, 2024.
- [6] T. Sağlam, et al., “Obfuscation-Resilient Software Plagiarism Detection with JPlag”, *ICSE-C*, IEEE/ACM, 2024.
- [7] T. Sağlam, et al., “How Students Plagiarize Modeling Assignments”, *MODELS-C*, ACM/IEEE, 2023.