
Wie kann KI Qualitätsprobleme in bestehender Software zielgerichtet beheben?

Dr. Benjamin Hummel



Was können LLMs heute und wo geht die Reise hin?



Wie kann KI bei der Softwareentwicklung am besten unterstützen?

FORBES > LEADERSHIP > CAREERS

AI Writes Over 25% Of Code At Google—What Does The Future Look Like For Software Engineers?

Jack Kelly Senior Contributor 

Jack Kelly covers career growth, job market and workplace trends.

Follow

  0

Nov 1, 2024, 06:30am EDT

Introducing Devin, the first AI software engineer

March 12, 2024 • by Scott Wu

Setting a new state of the art on the SWE-bench coding benchmark. Meet Devin, the world's first fully autonomous AI software engineer.

Wir backen erstmal kleinere Brötchen

Statische Analyse

10k Regeln

Regelwerk

100k – 10M Zeilen

```
public FindingResolutionResult getAIFindingResolution(  
    @Parameter(description = "The name of the configured AI engine to use.") @Param  
    @Parameter(description = "The id of the finding to find a resolution for.") @Param  
    @Parameter(description = "The timestamp parameter description") @QueryParam("TIMESTAMP")  
    throws StorageException, ServiceCallException {  
    TrackedFinding finding = retrieveTrackedFinding(findingId, commit);  
    if (!(Finding.getLocation() instanceof TextRegionLocation)) {  
        throw new BadRequestException("Method is only supported for findings with text  
    }  
    TokenElementInfo element = retrieveTokenElementInfo(commit, finding);  
    AIFindingResolutionEngine resolutionEngine = new AIFindingResolutionEngine(  
        (prompt, observeabilityMonitor) -> serviceInfo.getEngineProvider().comp  
        EAIFindingResolutionPromptGenerator.DEFAULT, observeabilityMonitor, null);  
    try {  
        String newElementContent = resolutionEngine.suggestResolution(finding, element  
            getFindingTypeDescription(finding));  
        return findAndConvertResolution(element, newElementContent);  
    } catch (FindingResolutionException e) {  
        return FindingResolutionResult.error(e.getMessage());  
    }  
}  
@Private static @Nonnull FindingResolutionResult trimAndConvertResolution(TokenElement  
    String newElementContent) {  
    List<String> oldLines = StringUtils.splitLinesAsList(element.getText());  
    List<String> newLines = StringUtils.splitLinesAsList(newElementContent);  
    int firstLine = 0;  
    while (firstLine < oldLines.size() && firstLine < newLines.size()  
        && oldLines.get(firstLine).equals(newLines.get(firstLine))) {  
        firstLine ++ 1;  
    }  
    int oldLastLine = oldLines.size() - 1;  
    int newLastLine = newLines.size() - 1;  
    while (oldLastLine >= 0 && newLastLine >= 0 && oldLines.get(oldLastLine).equals(  
        oldLastLine -- 1;  
    }  
}
```

Scanner
AST Parser
Datenfluss
Kontrollfluss
Suffix Trees
Machine Learning

10k – 10M Findings

Findings

Unused Code
Naming Issues
Klone
Speicherfehler
Fehlende
Kommentare

Was mache ich jetzt mit den ganzen Findings?

- Baselineing + änderungsgetriebenes Aufräumen
- Annotation von Pull-Requests
- Real-Time Feedback in der IDE

Auf gar keinen Fall: Alles (von Hand) Aufräumen

Quick Fix

```
2
3 import java.util.List;
4 import java.util.Set;
5
6 import jakarta.ws.rs.BadRequestException;
7 import jakarta.ws.rs.GET;
8 import jakarta.ws.rs.NotFoundException;
9 import jakarta.ws.rs.Path;
10 import jakarta.ws.rs.PathParam;
11 import jakarta.ws.rs.QueryParam;
12
13 import org.checkerframework.checker.nullness.qual.NonNull;
14 import org.checkerframework.checker.nullness.qual.Nullable;
15 import org.conqat.engine.commons.findings.location.TextRegionLocation;
16 import org.conqat.engine.commons.findings.location.TextRegionLocation;
```

Wo helfen Quick Fixes?

```
import jakarta.ws.rs.NotFoundException;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.ApplicationPath;
import
import
import org.checkerframework.checker.nullness.qual.NonNull;
import org.checkerframework.checker.nullness.qual.Nullable;
```

Unused import statement

Remove unused imports Alt+Shift+Enter More actions... Alt+Enter



**Flächendeckendes Aufräumen
der Codebasis?**

Warum haben wir nicht für alles einen Quick Fix?

- Quick Fixes sind nicht einfach zu schreiben
- Oft hängt die Lösungsvariante auch vom Kontext ab
- Viele Regeln \Rightarrow viele Quick Fixes
- Oft reicht reines Codeverständnis nicht aus; man braucht Sprachverständnis

```
/**  
 * Trims and converts the resolution of a given token element by comparing it with new element content.  
 * It calculates the differences between the old and new content, identifies the differing lines,  
 * and then returns a result containing the partial content and diff information.  
 *  
 * @param element The token element information that contains the original text content.  
 * @param newElementContent The new text content to compare against the original element.  
 * @return A FindingResolutionResult containing the processed old and new partial contents, their respective  
 */  
private static @NonNull FindingResolutionResult trimAndConvertResolution(TokenElementInfo element, Usage  
    String newElementContent) {
```

**Kann das nicht alles "dieses AI" für
mich machen?**

Warum sind Quick Fixes ein guter Startpunkt?

- Klar umrissenes Problem
- Relevante Hilfe bei der Entwicklung
- Überschaubarer Lösungsraum (nicht gleich "komplette Codebasis")

Code



Snippet



Prompt



LLM



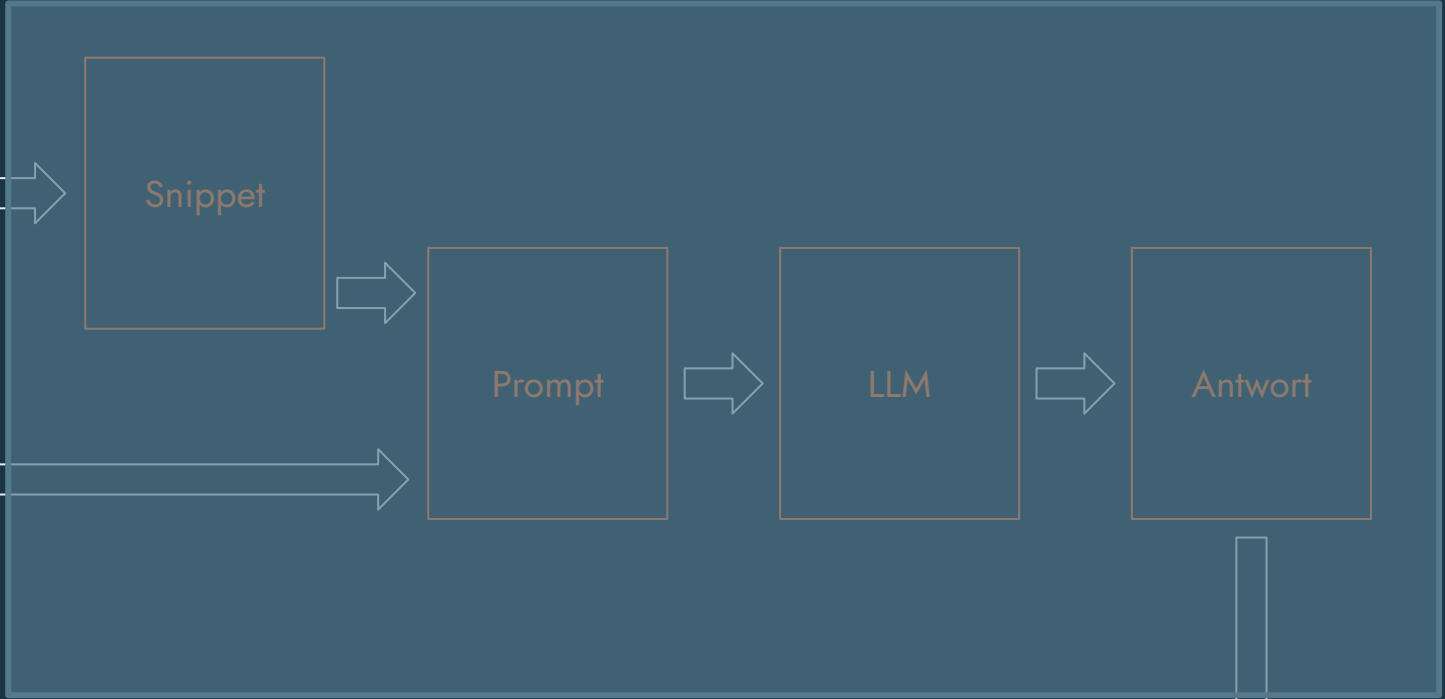
Antwort

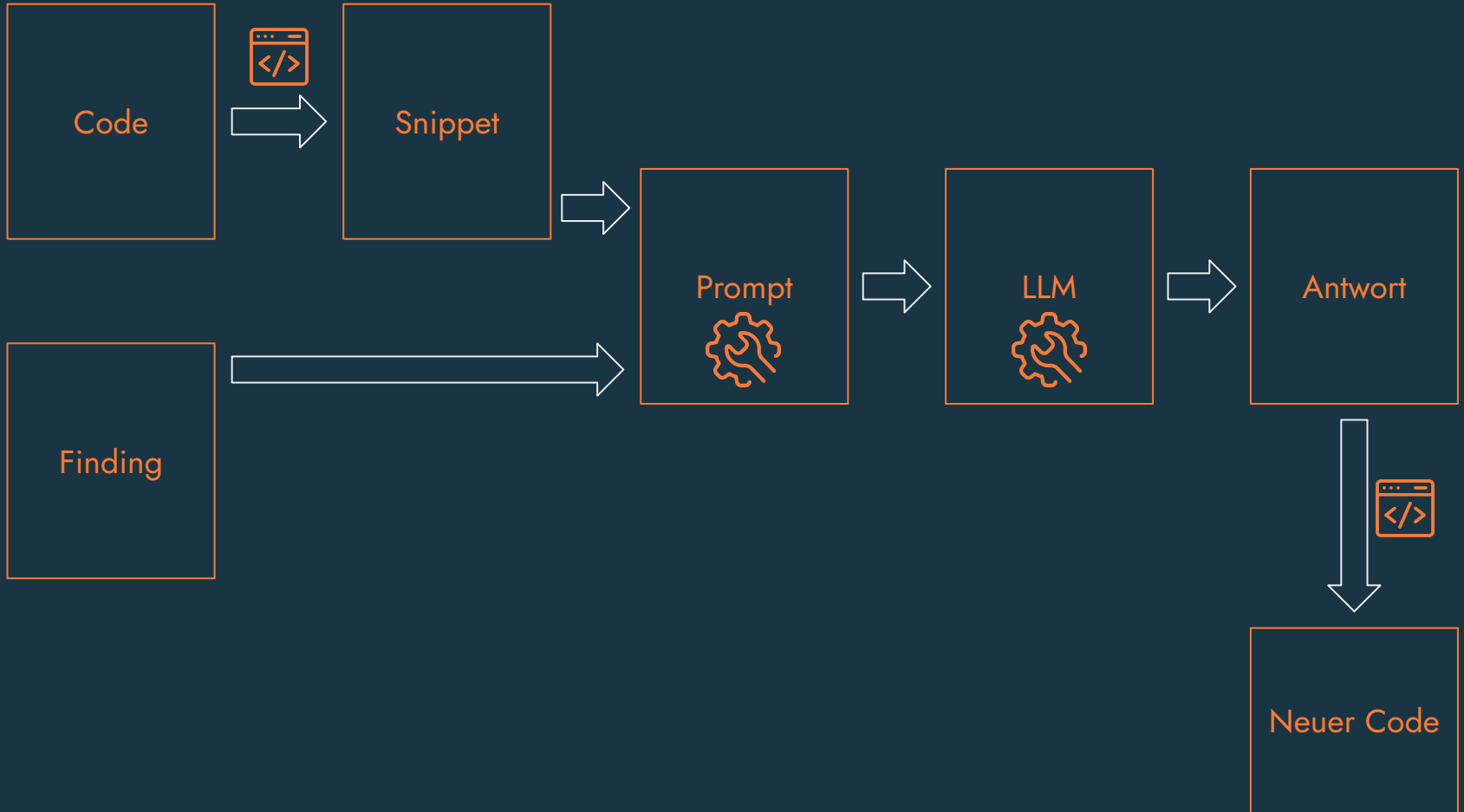


Finding



Neuer Code





The Good

```
/**
 * Generates js files and writes them to the typedefs folder.
 */
public static void generateTsDataClasses(File typedefsFolder) throws JsBuildException {
    typedefsFolder.mkdirs();

    List<File> filesToDelete = FileSystemUtils.listFilesRecursively(typedefsFolder);
    List<File> updatedFiles = new ArrayList<>();
    for (Class<? extends TsClassGeneratorBase> subclass : ClassIndex.getSubclasses(TsClassGen
        try {
            Constructor<? extends TsClassGeneratorBase> constructor = subclass.getCon
            TsClassGeneratorBase generator = constructor.newInstance(typedefsFolder);
            generator.generateFiles();
        } catch (IOException e) {
            throw new JsBuildException(subclass.getName() + " has no public constructor f
        } catch (IOException e) {
            throw new JsBuildException(e.getMessage(), e);
        }
    }
    filesToDelete.removeAll(updatedFiles);
    // Remove old files as Java types might e.g. have changed their name.
    for (File file : filesToDelete) {
        file.delete();
    }
}
```

Do something with the `boolean` returned by `delete`

```
    // Remove old files as Java types might e.g. have changed their name.
    for (File file : filesToDelete) {
        if (!file.delete()) {
            throw new JsBuildException("Failed to delete file: " + file.getPath());
        }
    }
}
```

```
package com.teamscale.index.repository.git.common;

import org.conqat.lib.commons.js_export.ExportToTypeScript;

/**
 * Stores the constant name values for the Voting and Detailed Line Comments
 * Connector Options
 */
@ExportToTypeScript
@SuppressWarnings("unused")
public enum EVotingConnectorOption {

    FINDINGS_INTEGRATION_OPTION(
        VoteSupportingGitRepositoryConnectorDescriptorBase.ENABLE_FINDINGS_INTEGRATION_OPTION_NAME),

    TEST_GAP_INTEGRATION_OPTION(
        VoteSupportingGitRepositoryConnectorDescriptorBase.ENABLE_TEST_GAP_INTEGRATION_OPTION_NAME),

    COMMIT_ALERTS_INTEGRATION_OPTION(
        VoteSupportingGitRepositoryConnectorDescriptorBase.ENABLE_COMMIT_ALERTS_INTEGRATION_OPTION_NAME);

    private final String optionName;

    EVotingConnectorOption(String optionName) {
        this.optionName = optionName;
    }

    public String getOptionName() {
        return optionName;
    }
}

@ExportToTypeScript
// Suppress 'unused' warning because the class is used only in TypeScript
@SuppressWarnings("unused")
public enum EVotingConnectorOption {
```

Undocumented SuppressWarnings


```
→ /** ←  
→ * Encodes a byte array as a hex string following the method described here: ←  
→ * http://stackoverflow.com/questions/9655181/convert-from-byte-array-to-hex ←  
→ * string-in-java ←  
→ */ ←  
public static String encodeAsHex(byte[] data) { ←  
    char[] hexChars = new char[data.length * 2]; ←  
    for (int j = 0; j < data.length; j++) { ←
```

Comment should not contain `//`

The AI model decided that the finding is a false positive.

```

*/
private Optional<ClassInfo> extractClassType(TypeSignature typeSignature, ClassInfo containingClass) {
    if (typeSignature instanceof ClassRefTypeSignature classRefTypeSignature) {
        ClassInfo classInfo = classRefTypeSignature.getClassInfo();
        if (classInfo == null) {
            String className = classRefTypeSignature.getFullyQualifiedClassName();
            reportClassNotFound(className, containingClass);
            return Optional.empty();
        }
        if (TYPES_WITH_VALUE_TYPE_IN_FIRST_TYPE_PARAMETER.contains(className)) {
            List<TypeArgument> typeArguments = classRefTypeSignature.getTypeArguments();
            if (typeArguments.isEmpty()) {
                System.err.println("No type arguments found in " + classRefTypeSignature);
                return Optional.empty();
            }
            return extractClassType(typeArguments.get(0).getTypeSignature(), containingClass);
        } else if (TYPES_WITH_VALUE_TYPE_IN_SECOND_TYPE_PARAMETER.contains(className)) {
            List<TypeArgument> typeArguments = classRefTypeSignature.getTypeArguments();
            if (typeArguments.isEmpty()) {
                System.err.println("No type arguments found in " + classRefTypeSignature);
                return Optional.empty();
            }
            // we ignore the key type and use only the value type of the map.
            return extractClassType(typeArguments.get(1).getTypeSignature(), containingClass);
        } else if (typeSignature instanceof TypeParameterSignature typeParameterSignature) {
            return extractClassType(typeParameterSignature.getNestedType(), containingClass);
        } else if (typeSignature instanceof TypeVariableSignature typeVariableSignature) {
            return extractClassType(typeVariableSignature.getNestedType(), containingClass);
        } else if (typeSignature instanceof ArrayTypeSignature arrayTypeSignature) {
            return extractClassType(arrayTypeSignature.getNestedType(), containingClass);
        }
        return Optional.empty();
    }
}

/**
 * Handles cases where we did not find a class that is used.
 * <p>
 * Adds it to {@link #wideTypeMessages} if it is {@link Object}. Ignores it if
 * it is one of the "expected" standard java types ("java.lang.String", etc.).
 * Otherwise, prints it on {@link System#err}.
 * <p>
 * In particular in backup indexes, it can be dangerous to use third-party
 * classes because we can't control how they evolve (and if they are evolving in
 * a serialization-compatible way).
 */
private void reportClassNotFound(String className, ClassInfo containingClass) {
    if ("java.lang.Object".equals(className)) {
        wideTypeMessages.add(className);
    } else {
        System.err.println("Wide type " + className + " stored in Index or Index-Value class");
    }
}

if (className.startsWith("java.lang.") || className.startsWith("java.util.")
    || className.startsWith("java.awt.Color") || className.startsWith("java.awt.Dimension") || className.startsWith("java.awt.Font") || className.startsWith("java.awt.Graphics") || className.startsWith("java.awt.Graphics2D") || className.startsWith("java.awt.Image") || className.startsWith("java.awt.RenderingHints") || className.startsWith("java.awt.Toolkit") || className.startsWith("java.awt.geom.AffineTransform") || className.startsWith("java.awt.geom.Area") || className.startsWith("java.awt.geom.Ellipse2D") || className.startsWith("java.awt.geom.GeneralPath") || className.startsWith("java.awt.geom.Point2D") || className.startsWith("java.awt.geom.Rectangle2D") || className.startsWith("java.awt.geom.RoundRectangle2D") || className.startsWith("java.awt.image.BufferedImage") || className.startsWith("java.awt.image.DataBuffer") || className.startsWith("java.awt.image.DataBufferInt") || className.startsWith("java.awt.image.DataBufferLong") || className.startsWith("java.awt.image.DataBufferShort") || className.startsWith("java.awt.image.DataBufferUShort") || className.startsWith("java.awt.image.DataBufferByte") || className.startsWith("java.awt.image.DataBufferShort") || className.startsWith("java.awt.image.DataBufferUShort") || className.startsWith("java.awt.image.DataBufferByte") || className.startsWith("java.awt.image.DataBufferShort") || className.startsWith("java.awt.image.DataBufferUShort") || className.startsWith("java.awt.image.DataBufferByte")) {
    return;
}

return;

```

```

} else {
    buildTrace(next, tracingLinks, seen, buffer, childrenPrefix + "  ", children);
}
}

/**
 * Extracts the "core" type from the given signature. For example, if the
 * signature is <code>LinkedList<CommitDescriptor></code>
 */
private Optional<ClassInfo> extractClassType(TypeSignature typeSignature, ClassInfo containingClass) {
    if (typeSignature instanceof ClassRefTypeSignature classRefTypeSignature) {
        return extractFromClassRefTypeSignature(classRefTypeSignature, containingClass);
    } else if (typeSignature instanceof TypeVariableSignature typeVariableSignature) {
        return extractFromTypeVariableSignature(typeVariableSignature, containingClass);
    } else if (typeSignature instanceof ArrayTypeSignature arrayTypeSignature) {
        return extractFromArrayTypeSignature(arrayTypeSignature, containingClass);
    }
    return Optional.empty();
}

private Optional<ClassInfo> extractFromClassRefTypeSignature(ClassRefTypeSignature classRefTypeSignature, ClassInfo classInfo) {
    ClassInfo classInfo = classRefTypeSignature.getClassInfo();
    if (classInfo == null) {
        String className = classRefTypeSignature.getFullyQualifiedClassName();
        reportClassNotFound(className, containingClass);
        return Optional.empty();
    }
    return classInfo;
}

private Optional<ClassInfo> extractFromTypeArguments(ClassRefTypeSignature classRefTypeSignature, List<TypeArgument> typeArguments) {
    if (typeArguments.isEmpty()) {
        System.err.println("No type arguments found in " + classRefTypeSignature);
        return Optional.empty();
    }
    return extractClassType(typeArguments.get(index).getTypeSignature(), containingClass);
}

private Optional<ClassInfo> extractFromTypeVariableSignature(TypeVariableSignature typeVariableSignature, TypeParameter resolved) {
    TypeParameter resolved = typeVariableSignature.resolve();
    return extractClassType(resolved.getClassBound(), containingClass);
}

private Optional<ClassInfo> extractFromArrayTypeSignature(ArrayTypeSignature arrayTypeSignature, TypeSignature signature) {
    while (signature instanceof ArrayTypeSignature) {
        signature = ((ArrayTypeSignature) signature).getNestedType();
    }
    return extractClassType(signature, containingClass);
}

```

Violation of method length threshold (source lines of code) of 30: 36

The Bad

```
const line = ' name?: string;';  
const styleRegions: StyleRegion[] = [];  
let sourceFormatterContext: SourceFormatterContext | null = null;
```

Constant `styleRegions` violates naming conventions.
Should be one of `([A-Z][_A-Z0-9]*)|(E[A-Z][a-zA-Z]+)|([A-Z][a-zA-Z0-9]+Context)`

```
const line = ' name?: string;';  
let styleRegions: StyleRegion[] = [];  
let sourceFormatterContext: SourceFormatterContext | null = null;
```

```
@Override
public int hashCode() {
    final int prime = 31;
    int result = super.hashCode();
    result = prime * result + (int) (lastChangeTimestamp ^ (lastChangeTimestamp >>> 32));
    return result;
}
```

Warum nicht `Objects.hash()` ?

```
/** {@inheritDoc} */
@Override
public int hashCode() {
    return Objects.hash(super.hashCode(), lastChangeTimestamp);
}
```

```
public @interface ExternalServiceTest {  
    /** The URI of the target server (http(s)://hostname or hostname:port). */  
    String value();  
}
```

Comment should not contain `//`

```
public @interface ExternalServiceTest {  
    /** The URI of the target server (http(s):&#47;&#47;hostname or hostname:port). */  
    String value();  
}
```

```
    }  
    recList.push(square);  
    recCentersY.push(selectorRecY + this.selectorRecHeight! / 2);  
    selectorRecY += this.selectorRecHeight! * 2; // Improved line  
}  
return recCentersY;
```

```
/** Number of line where exception occurred (1-based). */  
private final int lineNumber; // Made final  
  
/** Number of column where exception occurred (1-based). */  
private final int columnNumber; // Made final
```

```
/** Constructor. */
```

The Ugly


```
/// <summary>
/// Initializes a new instance of the <see cref="Opt
/// </summary>
/// <param name="item">The item to be edited.</param
# ←
protected OptionsEditorControlBase(T item)
{
    this.Item = item;
}

/// <inheritdoc/>
```

```
* shallow parser.
*/
@IndexValueClass(containedInBackup = true)
<Code>
public class MethodMatchRegion extends M
```

```
87     public static class ClassWithCollections {
88
89         List<String> stringList;
90
91         Integer[] numberArray;
92
93         List<KnownType> listWithKnownType;
94
95         List<UnknownType> listWithObjects;
96
97         @Nullable
98         List<Integer> nullableArray;
99     }
100
```

Interface comment missing (line 89).

```
87     /** A class that contains various collections */
88     public static class ClassWithCollections {
89
90         List<String> stringList;
91
92         Integer[] numberArray;
93
94         List<KnownType> listWithKnownType;
95
96         List<UnknownType> listWithObjects;
97
98         @Nullable
99         List<Integer> nullableArray;
100     }
101
```

```
// Remove old files as Java types might e.g. have changed their name.  
for (File file : filesToDelete) {  
    file.delete();  
}
```

Do something with the `boolean` returned by `delete`

```
// Remove old files as Java types might e.g. have changed their name.  
for (File file : filesToDelete) {  
    if (!file.delete()) {  
        // Handle the failure to delete the file, e.g., log a warning or throw an exception  
        System.err.println("Failed to delete file: " + file.getAbsolutePath());  
    }  
}
```

```
/// <inheritdoc/>
public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
{
    switch (value as EAssessment?)
    {
        case EAssessment.RED:
            return Resources.Assessment_Red;
        case EAssessment.YELLOW:
            return Resources.Assessment_Yellow;
        default:
            return string.Empty;
    }
}
```

The value of parameter `culture` is never used

```
/// <inheritdoc/>
public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
{
    switch (value as EAssessment?)
    {
        case EAssessment.RED:
            return Resources.ResourceManager.GetString("Assessment_Red", culture);
        case EAssessment.YELLOW:
            return Resources.ResourceManager.GetString("Assessment_Yellow", culture);
        default:
            return string.Empty;
    }
}
```

ResourceManager gibt es nicht!

Was heißt das denn jetzt?

Das sind ja wieder nur Anekdoten

Benchmarking

Eigene Codebasis

~ 1,6 MLOC

Java
TypeScript
C#

~ 8.600 Findings



Random Sample

100 Findings



Quick Fix Script

Verschiedene LLMs
Verschiedene Prompts

Manuelle Bewertung



Good

Kann man direkt
übernehmen

Acceptable

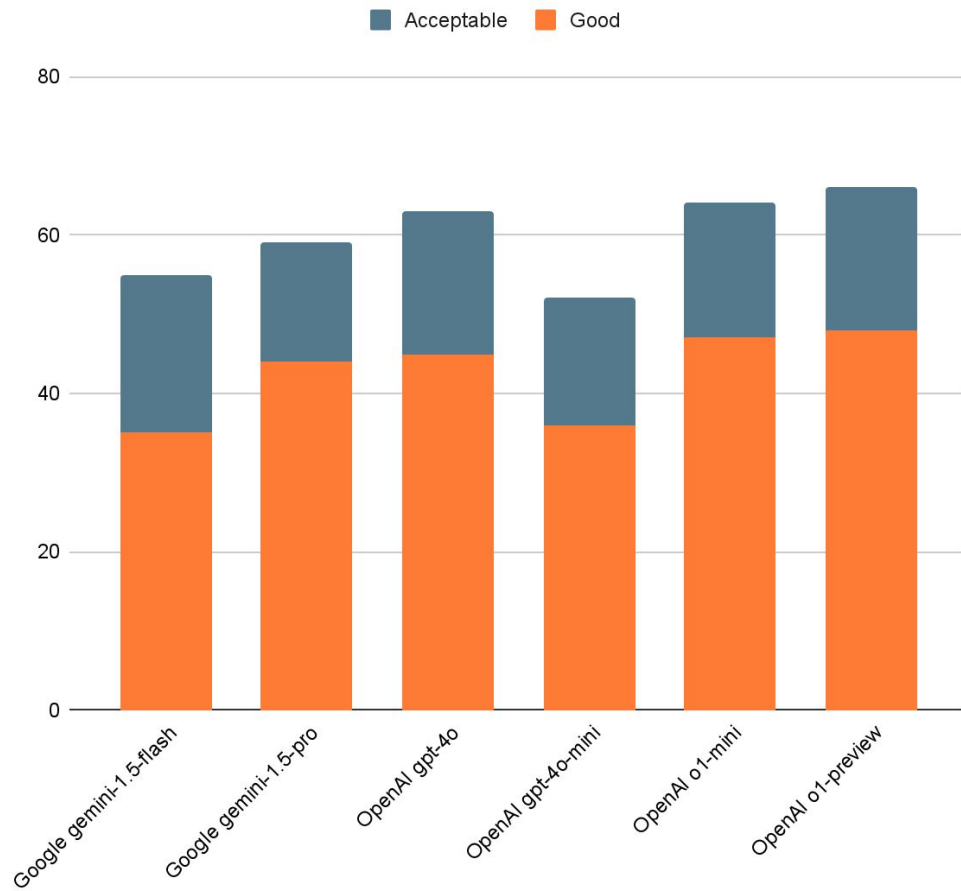
Guter Ansatz, würde man
i.d.R. nachbearbeiten

Unsolved

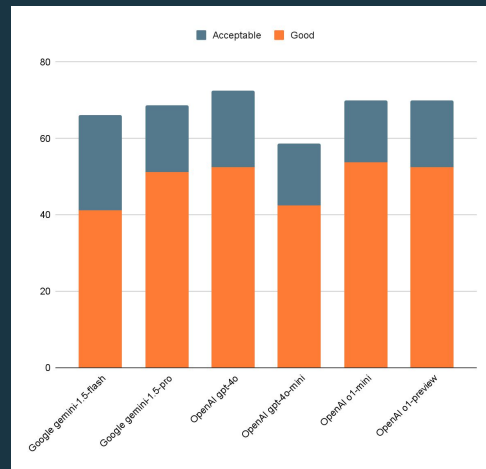
Das LLM sagt, dass es
keine Lösung findet

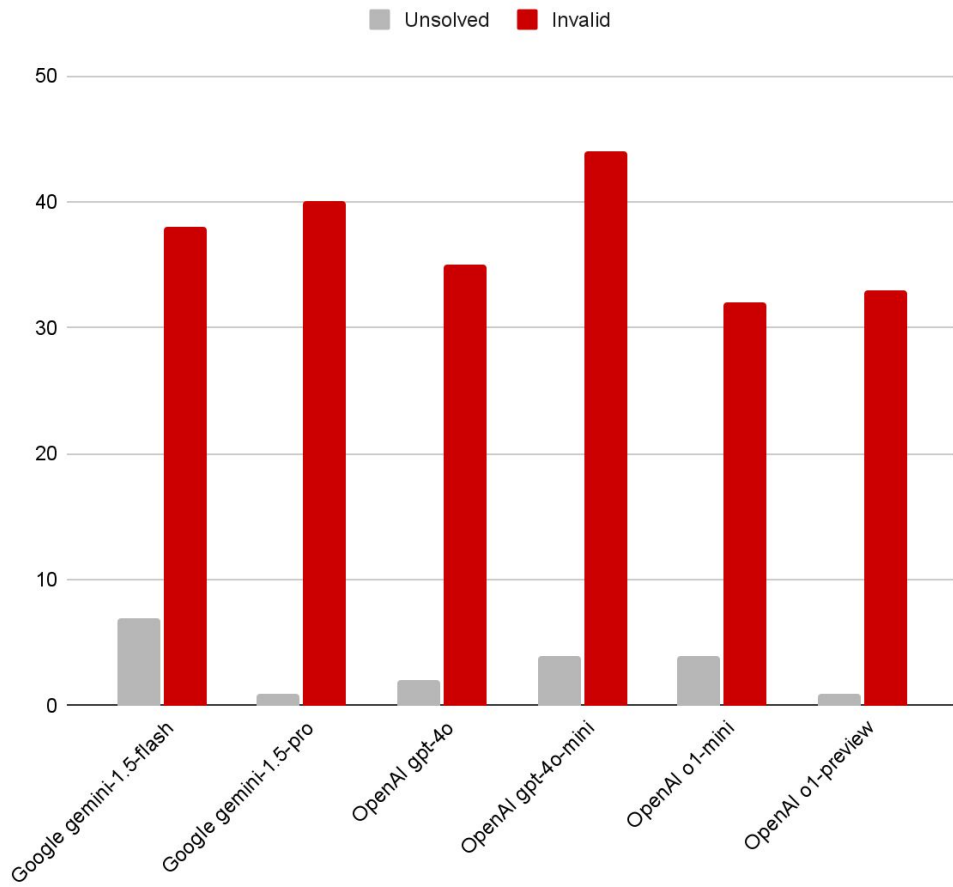
Invalid

Keine gültige
Antwort/Lösung

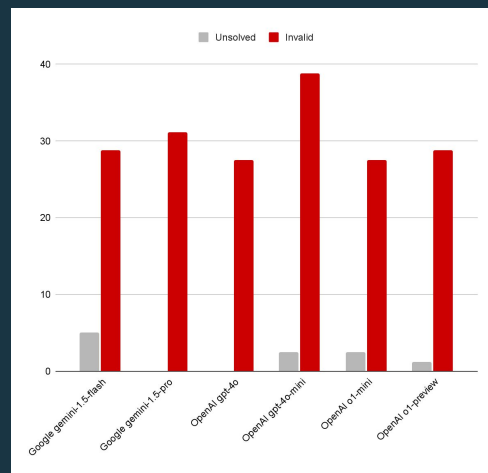


Ohne False Positives

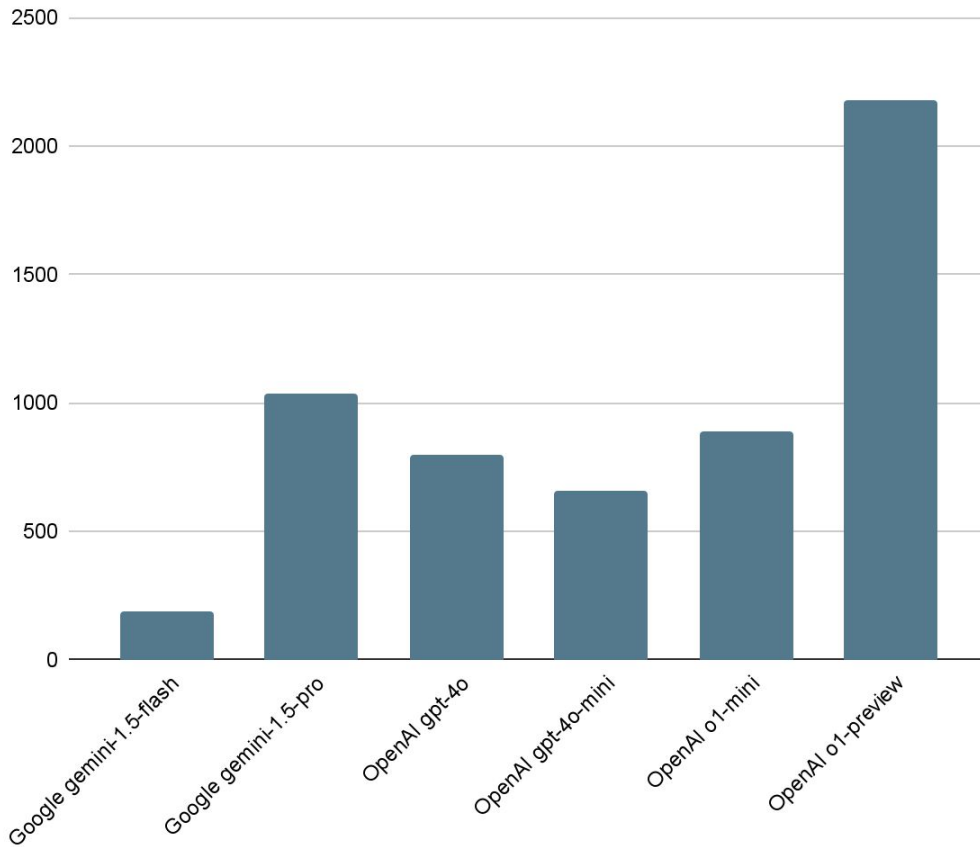




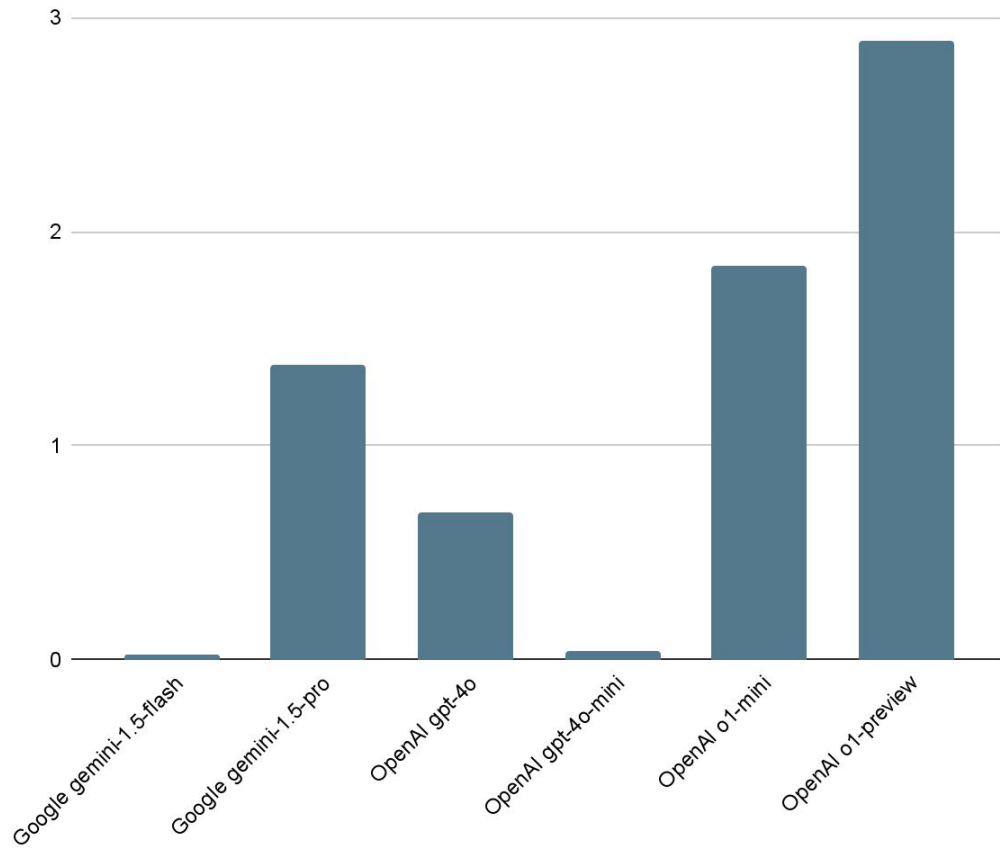
Ohne False Positives



Time (sec)



Cost (US-\$)



Fazit

Funktionieren LLM-basierte Quick Fixes?



**Flächendeckendes Aufräumen
der Codebasis?**

Ja, ca. $\frac{2}{3}$ der Fixes sind nützlich

Nein, ca. $\frac{1}{3}$ ist unbrauchbar

Was können wir noch besser machen?

- Auf bessere Modelle warten ← Wie sicher sind wir, dass die kommen?
- Selbst ein LLM trainieren ← Für “normale” Unternehmen nicht machbar
- Fine-tune Learning ← Braucht viele Daten und macht LLM-Updates schwieriger
- RAG ← Braucht auch viele Daten, aber kann man inkrementell ausbauen

Dr. Benjamin Hummel

 [linkedin.com/in/benjamin-hummel-cqse](https://www.linkedin.com/in/benjamin-hummel-cqse)

 hummel@cqse.eu

