# Bachelorarbeit

## Nutzung von Informationsquellen in der Softwaresicherheit

**Tim Hansen**
tihansen@uni-bremen.de
Mat-Nr: 400064

Erstgutachter: Prof. Dr. Rainer Koschke
Zweitgutachter: Dr. Karsten Sohr

Wirtschaftsinformatik
Universität Bremen
14. Januar 2021

# Vorwort

Ein besonderer Dank für die Unterstützung, das Feedback und die ständige Mithilfe an dieser Bachelorarbeit gebührt Anika Behrens, die mir als Betreuerin zur Seite gestanden hat und maßgeblich an der Findung der Fragestellung dieser Arbeit beteiligt war, ebenso wie Martin Schröer, der mich mit umfangreichem Feedback und nahezu endloser Geduld ebenfalls als Betreuer begleitet hat.

Weiterhin möchte ich mich bei Falko Galperin und Hugo Damer bedanken, welche die Entwicklung des **Mimesis**-Plugins voran treiben und mir geholfen haben, den Code zu verstehen, Probleme zu lösen und Tipps zur Optimierung gebracht haben (mit Antwortzeiten, die zu jeder Uhrzeit im Millisekundenbereich lagen).

Außerdem möchte ich mich bei den Gutachtern bedanken, die ihre Zeit in diese Arbeit investieren.

## Erklärung

Ich versichere, diese Bachelorarbeit selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, wurden unter Angabe der jeweiligen Quelle als solche kenntlich gemacht.

# Inhaltsverzeichnis

## Überblick

Im Rahmen dieser Arbeit soll unter Nutzung und Erweiterung der Möglichkeiten des **Mimesis**-Plugins eine Studie durchgeführt werden, in welcher das Verhalten von Softwareentwicklern bei der Nutzung von Dokumentation zur Lösung eines Problems beobachtet und analysiert wird.

Zunächst wird die Ausgangslage für diese Arbeit dargelegt und erklärt, auf welchen bereits bestehenden Arbeiten sie aufbaut (Kapitel 1). Darauf folgend werden Fragen formuliert, die im späteren Verlauf erneut aufgegriffen werden (Kapitel 4). Da das **Mimesis**-Plugin und die Erweiterung um eine Komponente, welche das Verhalten eines Entwicklers im Browser aufzeichnet, eine zentrale Rolle spielen, werden das Plugin und das Konzept für die Erweiterung vorgestellt (Kapitel 5). Nachfolgend wird die Implementierung der Erweiterung erläutert (Kapitel 6). Um die im Rahmen dieser Arbeit zu untersuchenden Forschungsfragen zu betrachten, wurde eine Studie durchgeführt, welche in Kapitel „Studie“ 7 detailliert beschrieben wird. Die Ergebnisse (Kapitel 8) werden dargestellt und diskutiert (Kapitel 9). Abschließend gibt es ein Fazit (Kapitel 10).

# 1 Motivation

Die Dokumentation von Schnittstellen und Entwurfsmustern erscheint in der Softwareentwicklung als eines der wichtigsten Elemente, die Programmierern zur Verfügung stehen, um Anforderungen umzusetzen oder generelles Wissen zu erwerben. Eine gute Dokumentation erleichtert den Einstieg in ein neues Gebiet erheblich und eine schlechte oder nicht vorhandene Dokumentation kann auch dafür sorgen, dass eine Technologie falsch oder gar nicht verwendet oder umgesetzt wird. [7] Das Anfertigen einer Dokumentation ist dabei keinesfalls nur als das Erstellen kleiner Erklärungen für die spätere Pflege einer Softwarelösung zu betrachten, sondern erfordert mitunter einen umfangreichen Aufwand [23].

Besonders im Bereich sicherheitsrelevanter Anforderungen an eine Software scheint eine gute Dokumentation von enormer Wichtigkeit, da die falsche oder unvollständige Umsetzung von Sicherheitsmechanismen zu erheblichen wirtschaftlichen oder persönlichen Schäden führen kann [36]. Es kann angenommen werden, dass Dokumentation häufig von Fehlern belastet ist [37] oder nicht die Informationen enthält, welche der Entwickler gesucht hat [19] und auch scheinbare Überfülle an Dokumentation kann diesen Umstand nicht ausgleichen [34]. Darüber hinaus bleibt zu beachten, dass manche Informationen von Entwicklern nicht als nutzbar identifiziert [13] oder durch eine zu hohe Sachkenntnis eine Art Tunnelblick für Informationen entsteht [8].

Nicht nur offizielle Dokumentation wird dazu genutzt, den Umgang mit einer Softwarelösung oder die Struktur eines Musters zu erlernen, häufig wird mit Hilfe von Suchmaschinen auch Wissen aus externen Quellen bezogen [26] – so entsteht jedoch die Möglichkeit, dass eine Lösung umgesetzt wird, die nicht im Sinne der eigentlichen Aufgabe ist [18].

# 2 Bestehende Arbeiten

Es wurde in vergangenen Arbeiten festgestellt, dass Entwickler dazu neigen, Werkzeuge[13], Informationen [2] und Abläufe [28] zu ignorieren, die spezifisch dafür entwickelt wurden, bei der Informationssuche und -verarbeitung zu unterstützen. Während die Sicherheit von Softwarelösungen mit einer immer höheren Wichtigkeit betrachtet wird, so kann im Gesamtverlauf der letzten Jahre ein deutlicher Anstieg von Sicherheitsproblemen festgestellt werden [34]. [15] Oliveira et al. stellten in ihrer Arbeit „It's the psychology stupid“ fest, dass die von ihnen untersuchten Softwareentwickler während ihrer Arbeit am Code nicht selbstständig auf Sicherheitsaspekte achten [28]. Vielmehr konzentrieren sich die Lösungsansätze auf die Einbringung der funktionalen Anforderungen [4] [14].

Besondere Beachtung sollte auch die Feststellung finden, dass gerade sehr erfahrene Entwickler anfälliger dafür sind, Sicherheitslücken im Programmcode zu hinterlassen, da sie unter einer „Expertenblindheit“ leiden, bei der sie sich an bereits bekannte Muster halten und neue Informationen nicht in ihre Lösungen einbringen [8]. Außerdem wurde mehrfach beobachtet, dass Entwickler spezifische Informationen zu Sicherheitsaspekten sogar schlicht ignorieren, unabhängig davon, wie versiert sie in der Entwicklung von Programmcode sind [28] [2] [31] [13].

Softwaresicherheit ist ein Thema von höchster Bedeutung, [22] denn wenn sie unzureichend umgesetzt wird, entstehen Risiken, die mit fortschreitender Digitalisierung immer präsenter werden und auch schwerwiegende Auswirkungen auf das öffentliche Leben haben können [27].

Andere Arbeiten enthalten bereits Informationen darüber, dass die Art und Weise sowie die Intensität der Nutzung von Dokumentation unterschiedlich zwischen den verschiedenen Phasen der Softwareentwicklung ist [31]. So wird die Dokumentation während der initialen Implementierungsphase einer Software intensiver genutzt als zur Maintenancephase, in welcher sie gepflegt wird [11]. Des Weiteren verändert der agile Arbeitsansatz den Blickwinkel auf die Dokumentation und sorgt dafür, dass diese als notwendiges Nebenprodukt zu betrachten ist, nicht jedoch einen Hauptteil der Aufmerksamkeit gewidmet bekommen sollte [33]. Dennoch gehören zu den wichtigsten Qualitätsmerkmalen für Softwaredokumentation Vollständigkeit, Beständigkeit und Verfügbarkeit, wobei die Kosten im Hintergrund stehen [3] [39] und das Angebot der Werkzeuge, welche verwendet werden können um diese Ansprüche zu erfüllen stetig weiter wächst, was die Verfügbarkeit und Interaktivität der Dokumentation verbessert [38].

Die verschiedenen Aussagen zu der Notwendigkeit von Softwaredokumentation spiegeln die Probleme damit wider, da die allgemeine Meinung gegenüber Dokumentation vor allem aus Unmut und Unzufriedenheit besteht, da diese als unvollständig und die Qualität meistens als unzureichend bewertet wird, wie unter anderem Garousi et al. feststellen konnten [10] [17]. Diese allgemein scheinbar mangelhafte Qualität von Dokumentation ist jedoch nicht nur auf die Möglichkeiten und Ressourcen zurückzuführen, sondern auch in der mangelnden Bereitschaft von Softwareentwicklern Quellcode zu dokumentieren zu begründen [1]. Auch ist wenig beleuchtet, welche Fragen Entwickler stellen, wenn sie eine Programmieraufgabe lösen, wie diese durch die Dokumentation beantwortet werden können [32] [16] und wie sie in einer unbekannten Softwareumgebung nach Informationen suchen [12]. Es konnte durch Umfragen festgestellt werden, dass Softwareentwickler bestimmten Bereichen der Dokumentation eine höhere Relevanz zuschreiben als anderen und es demnach wichtiger empfinden, dass Quellcode und Funktionen mit Kommentaren versehen werden, als dass das Gesamtsystem dokumentiert wird [33].

# 3 Ausgangslage

Um die Nutzung von Dokumentationen durch Softwareentwickler besser verstehen zu können, soll eine Möglichkeit geschaffen werden, mit Hilfe derer eine Aufzeichnung und Analyse des Verhaltens möglich wird. Um dieses Ziel zu erreichen, soll das an der Universität Bremen entwickelte Tool **Mimesis** verwendet und erweitert werden, dass die Nutzung von Quellen während der Programmierung mitgeschnitten werden kann. Das Tool **Mimesis** wurde zu dem Zwecke entworfen, die Arbeitsweise und das Verhalten von Entwicklern während der Implementierung von Softwareanforderungen aufzuzeichnen, sodass es anschließend analysiert werden kann. Ein besonderer Fokus soll dabei auf die Nutzung von Quellen während der Umsetzung von sicherheitsrelevanten Anforderungen gelegt werden, da kaum Arbeiten mit Blick auf diesen spezifischen Aspekt existieren. Durch die Auswertung dieser Aufzeichnungen sollen Rückschlüsse darüber gezogen werden können, wie sich die Nutzung der Dokumentation auf die Qualität und Sicherheit von Softwarelösungen auswirken.

# 4 Fragestellung

Um ein besseres Verständnis für die Verwendung von Dokumentation während der Umsetzung von Anforderungen an eine Software und die Gründe für einen erfolgreichen oder unzureichenden Wissensgewinn durch Dokumentation zu erfassen, soll das Verhalten in Bezug auf die Nutzung offizieller und externer Quellen untersucht werden.

**F1: Welchen Einfluss haben Dokumentation und externe Quellen auf die Softwarequalität und wie werden sie während der Entwicklung einer Software oder der Umsetzung einer Softwareanforderung verwendet?**

Diese Frage umschreibt den großen Blickpunkt auf den Einfluss von Softwaredokumentation auf die Qualität von Software [29] und die Art und Weise der Verwendung durch Softwareentwickler [18]. Die Fragestellung ist breit gefächert und zeigt in die generelle Richtung, welche untersucht und konkretisiert werden soll. Hierbei sollen Muster beobachtet und Vermutungen aufgestellt werden. Eine vollständige Beantwortung des gesamten Fragebereichs wird nicht angestrebt, sondern vielmehr durch den spezifischen Ansatz im Bereich der sicherheitsrelevanten Anforderungen eine Lösung geschaffen, welche die Beantwortung dieser weiterführenden Frage ermöglicht. Die Fragestellung bietet den offenen Einstiegspunkt, der in einem konkreten Teilgebiet untersucht werden soll.

**F2: Zu welchen Zeitpunkten greift ein Entwickler auf externe Quellen zurück, um Wissen zu erlangen und in welcher Form wird nach Quellen gesucht?**

Hierbei wird nicht nur auf den zeitlichen Aspekt der Entwicklung geblickt, sondern auch auf ein „Wann“ im Sinne des Fortschritts der Bearbeitung, unabhängig zu dem absoluten Zeitpunkt. Können spezifische Aspekte herausgestellt werden, die den Nutzer dazu bringen, auf externes Wissen zuzugreifen? Ein erwartetes Ergebnis hierbei wäre, dass der Nutzer nach Analyse der Fragestellung zunächst eine Suche nach Arbeitsansätzen durchführt.

Weiterhin ist in dieser Frage auch festgehalten, ob die genutzten Quellen für den vorgesehenen Zweck geeignet waren [18] und inwiefern sie sich unterscheiden, wenn ein genereller Überblick geschaffen oder ein konkretes Problem gelöst werden soll.

**F3: Welche Informationen werden zur Bewältigung sicherheitsrelevanter Aspekte in der Programmierung benötigt und aus welchen Quellen werden sie beschafft?**

Mit dieser Fragestellung wird spezifisch darauf abgezielt, Sicherheit in der Softwareentwicklung und die benötigten Informationen in Verbindung zu setzen. Das Fehlen von Wissen während der Implementierung könnte zu einer Sicherheitslücke oder einem allgemein nicht sicheren System führen. Es wird also untersucht, wie Softwareentwickler versuchen, alle benötigten Informationen zu erhalten und ob diese Suche von den verwendeten Quellen unterstützt wird.

**F4: Können Quellen in sicherheitsrelevanten Anwendungsgebieten als vertrauenswürdig eingestuft werden? Welche Risiken bestehen?**

Im Zuge dieser Fragestellung soll aus einem anderen Blickwinkel auf den Ansatz der Arbeit geblickt werden. Während Quellen den Entwickler bei seiner Recherche allgemein unterstützen, so wäre es denkbar, dass eine Quelle auch gezielt für eine nicht optimale Umsetzung im Sinne von Softwaresicherheit sorgen könnte [15]. Hierbei soll betrachtet werden, was mögliche Gründe für solch eine Situation wären, welche Folgen daraus entstehen und wie ein Entwickler darauf reagiert.

# 5 Konzept

Um die Nutzung von Quellen durch Programmierer untersuchen zu können, wird das Verhalten während der Implementierung mitgeschnitten. Den Einstiegspunkt für dieses Vorhaben liefert das **eclipse**-Plugin **Mimesis**. Das Plugin wurde an der Universität Bremen entwickelt, um die Verwendung des **eclipse-IDE** während der Programmierung zu untersuchen. Entsprechend der Anforderungen von **eclipse** wurde das Plugin in Java implementiert.

Während einer Aufzeichnung durch das Plugin kann eine Reihe von Events aufgezeichnet werden, die auftreten, wenn der Nutzer mit dem **eclipse-IDE** interagiert. Unter anderem Events wie diese:

| Events |
| --- |
| **CodeChangeEvent**<br>Wird aufgezeichnet, wenn Code verändert wird |
| **CodeCompletionEvent**<br>Wird aufgezeichnet, wenn die automatische Codevervollständigung genutzt wird |
| **DebugEvent**<br>Wird aufgezeichnet, wenn in den Debug-Modus gewechselt wird |
| **FileEvent**<br>Wird aufgezeichnet, wenn eine Datei erstellt, gelöscht, geöffnet, verschoben oder auf andere Art und Weise verändert wird |
| **LaunchEvent**<br>Wird aufgezeichnet, wenn die Anwendung ausgeführt wird |
| **PerspectiveEvent**<br>Wird aufgezeichnet, wenn die Perspektive, also die Ansicht (Test, Debug, etc.) gewechselt wird |

Es werden weitere Events aufgezeichnet, die den Arbeitsfluss des Anwenders innerhalb der **eclipse-IDE** noch detaillierter darstellen können, wie zum Beispiel verschiedene Maus-Events, die für diese Arbeit jedoch keinen Mehrwert liefern.

Zu den bereits vorhandenen Events wird nun im Zuge dieser Arbeit ein weiteres Event hinzugefügt, mit dessen Hilfe es ermöglicht wird, die Interaktion eines Entwicklers mit dem in **eclipse** integrierten Browser aufzuzeichnen. Durch die Integration dieses neuen Events sollen nach erfolgter Aufzeichnung Rückschlüsse auf die Nutzung der Quellen und die Auswirkungen auf den entwickelten Code gezogen werden können. Jedes aufgezeichnete Event wird mit einer eindeutigen ID versehen und in der Aufzeichnungsdatei hinterlegt.

Das **Mimesis**-Plugin stellt eine Visualisierung der aufgetretenen Events zur Verfügung, wie in Abbildung 1 zu sehen.

Jeder Punkt bildet ein aufgetretenes **Event** ab. Auf der Y-Achse ist die bisher verstrichene Zeit eingetragen und auf der X-Achse lässt sich nachvollziehen, in welcher Zeile im Quellcode gearbeitet wird. Es besteht bereits die Möglichkeit, bestimmte Dokumentationen zu untersuchen, sofern diese als Dateien mitgeliefert werden, da das Plugin ausschließlich die Verwendung der Dateien untersucht, die sich im derzeit geöffneten Projekt befinden. Da für diese Arbeit insbesondere interessant ist, wie die Verwendung von externen Quellen abläuft, soll das Plugin so erweitert werden, dass das Mitschneiden des internen **eclipse**-Browser ermöglicht wird.

Um alle teilnehmenden Personen unter den gleichen Umständen untersuchen zu können, beschränkt sich die Implementierung auf den internen Browser von **eclipse**. Der **eclipse** interne Browser ist eine sehr minimalistische Variante, so ist nur das simple Browsen einer Webseite, ohne Verwendungen von etwaigen Addons, Shortcuts oder Erweiterungen möglich, wodurch eine eindeutige Auswertung der Ergebnisse erhofft wird. Durch die Aufzeichnung der Verwendung des Browsers in Echtzeit entsteht die Möglichkeit, das Verhalten des Entwicklers während der Verwendung der **eclipse-IDE** genau zu analysieren. Es kann nachvollzogen werden, zu welchen Zeitpunkten der Entwickler auf externe Quellen zugegriffen hat, welche Änderungen nach dem Konsultieren der Quellen entstanden sind und wie sich diese auf die Umsetzung der Anforderung ausgewirkt haben [18].

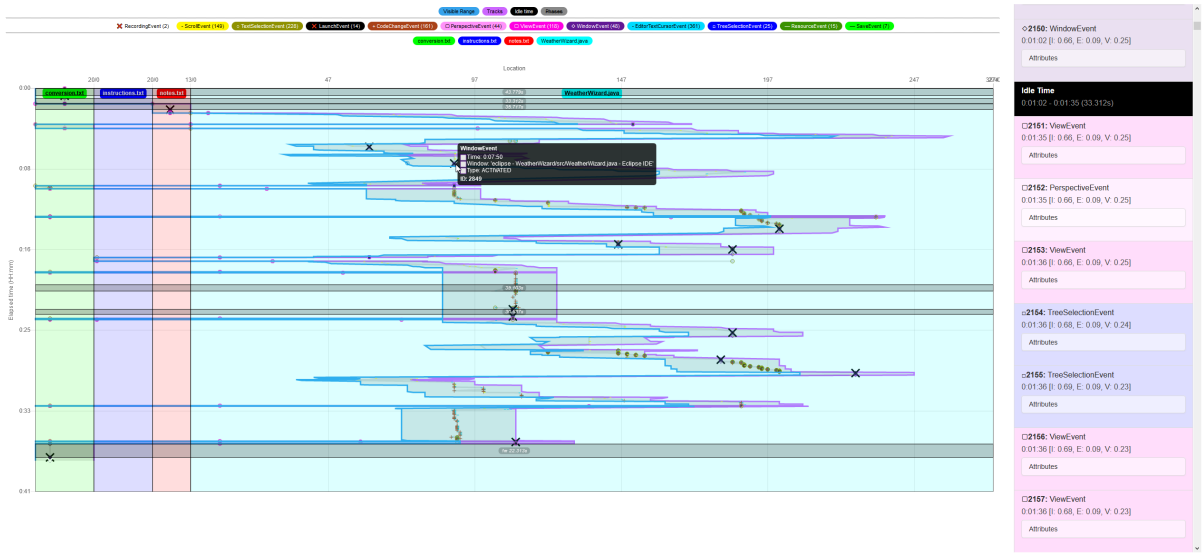


Abbildung 1: Visualisierung von **Events** in **Mimesis**

Da die Verwendung nicht nur auf dafür entworfenen Systemen stattfinden soll, sondern auch Fernaufzeichnungen durchgeführt werden sollen, bietet die Verwendung des internen Browsers auch einen maßgeblichen Datenschutzaspekt, da so andere auf dem System installierte Browser nicht überprüft werden müssen.

# 6 Implementierung

**Hinweis zum Quellcode**

Der Programmcode befindet sich im Verzeichnis **Quellcode**.

Der Programmcode der **Mimesis**-Anwendung befindet sich, wie in der Abbildung 2 dargestellt in mehreren Projekten, welche unter anderem die verschiedenen Anwendungsdomänen des Plugins widerspiegeln. Hierbei sind für die Erweiterung um den Teil des Browsers vor allem die Pakete **plugin**, **visual** und **common** interessant. Das Paket **common** enthält dabei nur geteilte Datenobjekte, weshalb es aus der Charakteristik der domänenspezifischen Aufteilung herausfällt. Das **plugin**-Paket enthält die Logik, welche sich mit der Aufzeichnung der Events beschäftigt, das **visual**-Paket hingegen enthält den Teil des Programmcodes, mit welchem die Visualisierung der aufgezeichneten Events umgesetzt wurde. Um die Erweiterung umzusetzen, waren Änderungen in diesen beiden Paketen (und entsprechend auch dem **common**-Paket) vorzunehmen, um ein neues Event einzupflegen, welches die Daten enthält, die bei der Nutzung des **eclipse** internen Browsers anfallen.

Das **Mimesis**-Plugin bietet eine vorausschauende Architektur, die eine Integration neuer Komponenten auf einfache Art und Weise ermöglicht, da durch den allgemeinen `Recorder` eine Schnittstelle gegeben ist, in welche weitere innerhalb von **eclipse** auftretende Events eingespeist werden können.

Der Quellcode für den Mitschnitt der Browser-Aktivität in **eclipse** beschränkt sich (abgesehen von Datenobjekten) auf zwei Klassen. Zunächst gibt es die Klasse **WebBrowserLocationListener**, welche sich an den **eclipse**-Browser ankoppelt und somit die erzeugten Events verarbeiten kann. Diese Klasse bildet das Herzstück der Erweiterung, da mit ihr ein `EventListener` an den **eclipse**-Browser angehängt wird. Der **eclipse**-Browser ruft mit jeder Veränderung der Adresszeile den `EventListener` auf und übergibt ihm die Daten des Events. Das Event wird einmal in eine interne Darstellung für das **Mimesis**-Plugin umgewandelt und dann an den Programmteil übergeben, der für die Aufzeichnung der Events zuständig ist. Das weitergereichte Event ist das `WebBrowserEvent`, welches die geladene Adresse mitliefert.

Diese Art der Datenerfassung fügt sich optimal in die bestehende Struktur des **Mimesis**-Plugins ein, da hier bereits auf einer Event-Struktur [30] gearbeitet wurde. Das Event wird lediglich aufbereitet und an die Klasse `Recorder` übergeben, welche das Event aufzeichnet und persistiert. Um den `EventListener` zu erzeugen, wurde in der `WebBrowserLocationListener`-Klasse auch eine entsprechende `Factory`-Klasse erzeugt, da das **Mimesis**-Plugin alle gegebenen `EventListener` beim Start aus einer Liste instanziiert. In dieser `Factory`-Klasse war es notwendig, mit **Reflections** [6] zu arbeiten, da der **eclipse**-Browser nur eine sehr begrenzte Schnittstelle bietet. Des Weiteren war die Implementierung der Methoden `attach` und `detach` notwendig, welche für die Verbindung und Trennung von `EventListener` und Browser zuständig sind, denn so wird der `EventListener` angehängt und vom Browser mit Events versorgt. Die Klasse wurde außerdem so angepasst, dass die URLs von den drei größten Suchmaschinen (Google, Bing, DuckDuckGo) [35] lesbarer dargestellt und die von den Testpersonen eingegebenen Suchbegriffe leichter erkannt werden können. In der Abbildung 4 ist diese Erkennung nachvollziehbar. Die Auswertung der Suchbegriffe wird damit erleichtert, außerdem wird hiermit eine Limitierung des **eclipse**-Browsers umgangen, welcher keine Abfrage des Titels einer Seite ermöglicht, sondern lediglich die derzeitige URL zurückliefert.

Die Logik der Implementierung wurde mit Hilfe von **Unit-Tests** [5] [20] überprüft.

Die `WebBrowserEvents` werden nun aufgezeichnet und persistiert, dementsprechend sind sie nun auch in der Visualisierung einsehbar, dargestellt in Abbildung 3. Die aufgezeichnete `BrowserLocation` setzt sich zusammen aus dem aufgerufenen Host und der angefügten Query. So ist es möglich, nachzuvollziehen, mit welchen Parametern beispielsweise eine Websuche durchgeführt wurde. Der interne **eclipse**-Browser bietet keine Möglichkeit beispielsweise den Seitentitel abzurufen oder den Quelltext der Seite zu analysieren, daher werden die Informationen mit Hilfe von Hostname und Query abgebildet.

Durch die aus den Einträgen gewonnenen Informationen kann abgeleitet werden, an welcher Codestelle eine Testperson auf externe Quellen zugegriffen hat und welche Suchmaschinen mit welchen Parametern verwendet wurden.

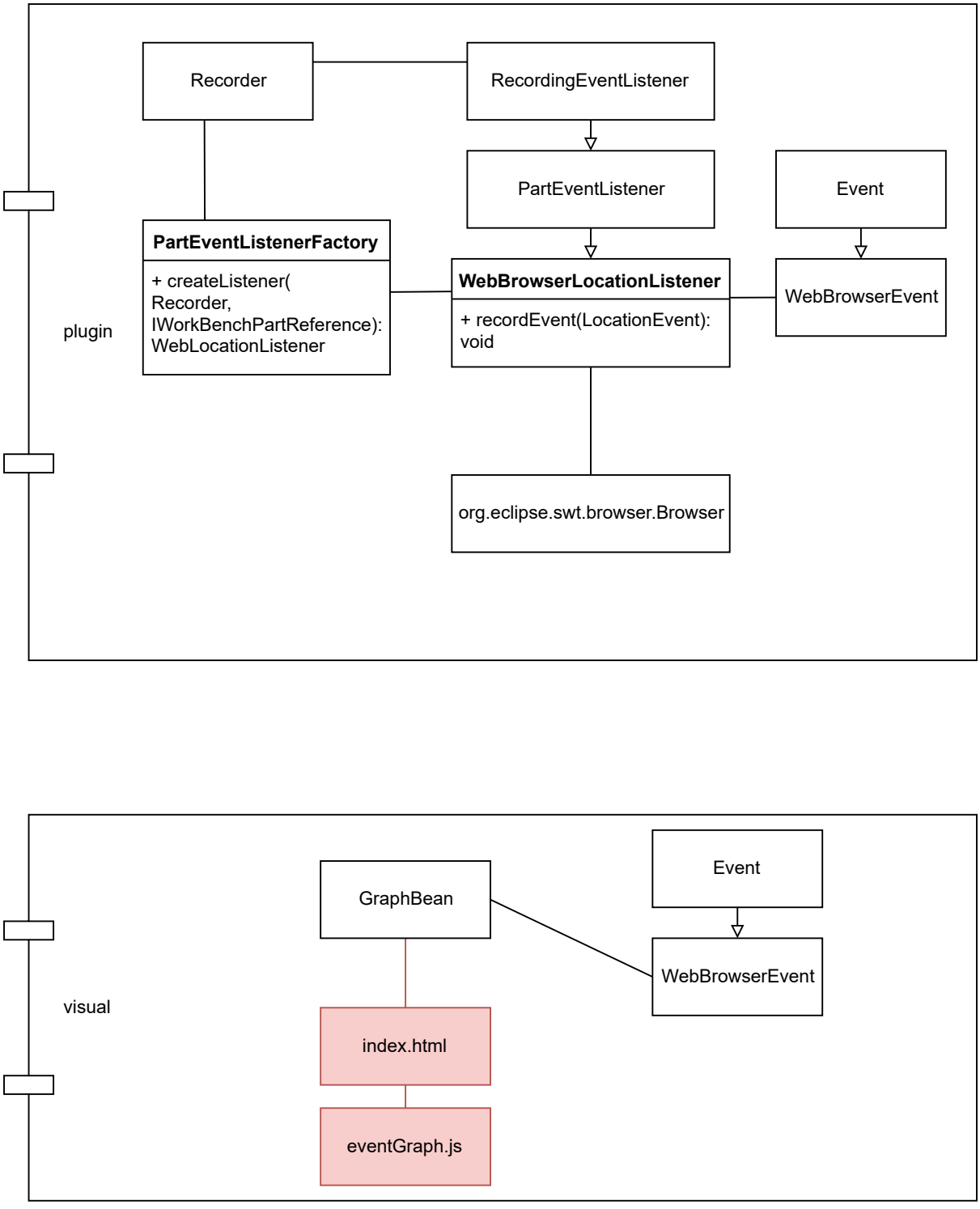



Abbildung 2: Zusammengefasstes Klassendiagramm

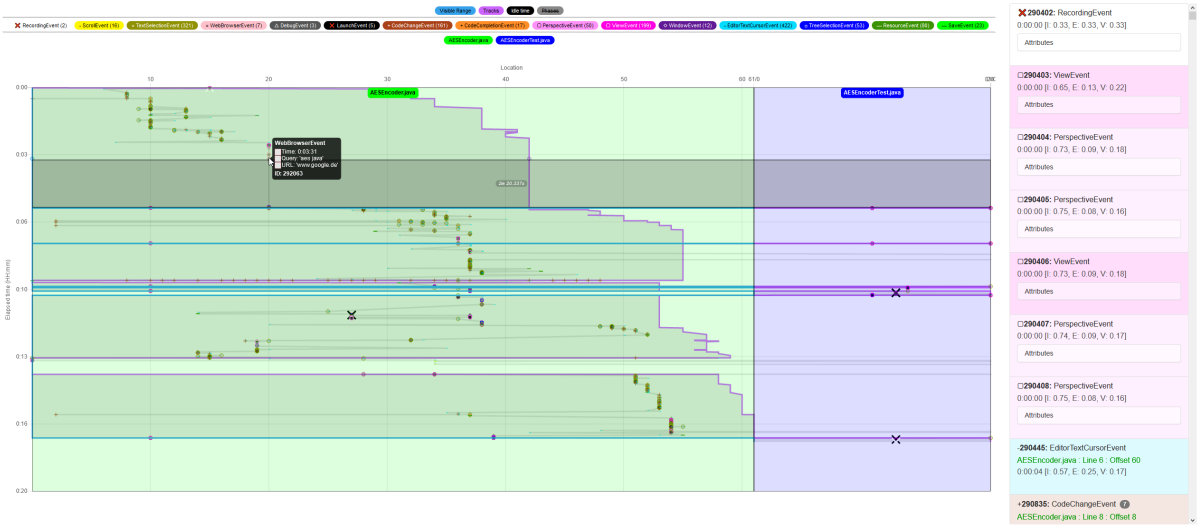


Abbildung 3: Visualisierung von **WebBrowserEvents** in **Mimesis**

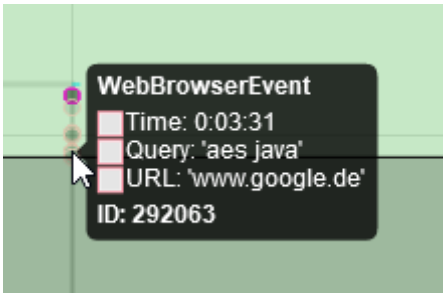



Abbildung 4: Detailansicht eines **WebBrowserEvents** in **Mimesis**

# 7 Studie

## 7.1 Hintergrund

Für die Beantwortung der am Anfang der Arbeit dargestellten Fragen wird auf die von den Testpersonen wahrgenommenen Herausforderungen und Problematiken eingegangen, sowie herausgestellt, wo die Schwächen oder Stärken in der Dokumentation im spezifischen Bereich der Aufgabenstellung lagen [9]. Die einzelnen Aufzeichnungen enthalten jeweils eine hohe Datenmenge, was detaillierte Rückschlüsse aus den gesammelten Informationen erleichtert. Im Zuge der Analyse werden Ähnlichkeiten untersucht und herausgestellt, welche die durch die Fragestellungen erzeugten Annahmen bestätigen oder widerlegen.

## 7.2 Akzeptanztest

Um die eingangs formulierten Fragestellungen mithilfe des **Mimesis** Plugins und des neuen **WebBrowserEvents** in einer praxisnahen Situation zu untersuchen, wurde eine entsprechende Studie durchgeführt. Für diese Studie wurde eine Aufgabenstellung formuliert, welche im Verlauf von Testpersonen bearbeitet werden soll.

Während der Evaluation wird neben einer qualitativen Studie auch gleichzeitig ein Akzeptanztest durchgeführt, mit welchem nachvollzogen wird, ob die Erweiterung des Plugins dafür geeignet ist, die gesuchten Informationen zu ermitteln. Diese Überprüfung wird als erfolgreich betrachtet, wenn eine Auswertung der Studie auf die gewünschte Art und Weise möglich ist.

Die implementierte Erweiterung des **Mimesis**-Plugins wird außerdem auf ihre Einsetzbarkeit in weiteren Studien hin geprüft. Hierzu soll mit einer Auswahl von Testpersonen eine Anforderung umgesetzt werden und während der Bearbeitung wird das Verhalten im Bezug auf die Nutzung von Quellen untersucht. Hierbei wird nicht nur die Implementierung der **WebBrowser**-Komponente getestet, während der Umsetzung konnten auch Fehler im **Mimesis**-Plugin aufgedeckt und verbessert werden. Nach der Bearbeitung der Aufgabenstellung durch die jeweilige Testperson wird auch die Anwendbarkeit der Erweiterung geprüft. Hierfür sollen die Testpersonen bewerten, wie gut sie mit der Erweiterung umgehen konnten und ob die Arbeit mit der Erweiterung ihren Arbeitsfluss eingeschränkt hat oder ob sie wie gewohnt arbeiten konnten.

## 7.3 Methode

Die Studie besteht aus den qualitativen Daten der Befragungen und Aufzeichnungen, sowie deren Auswertung.

Der qualitative Ansatz wurde gewählt, da es im Bereich der Nutzung von Dokumentation und dem Blickwinkel auf Zeitpunkt und Einfluss auf Qualität und Sicherheit kaum Arbeiten existieren und im Vorfeld keine Thesen vorhanden sind, welche gefestigt oder widerlegt werden könnten [24].

Um einen möglichst gleichartigen Ablauf zu gewährleisten [25] und wenig Verunreinigungen durch einen qualitativen Ansatz einzubringen, wurde ein Protokoll der Kommunikation mit den einzelnen Testpersonen angelegt. Das Protokoll sieht folgenden Ablauf vor:

### 7.3.1 Protokoll

| Persönliches Gespräch mit der Testperson während der Vorbereitung |
| --- |
| 1. Aufklärung über Datenschutzvereinbarung |
| 2. Unterschrift der Datenschutzvereinbarung |
| 3. Einrichtung der **eclipse**- und **Mimesis**-Umgebungen auf dem Zielsystem |
| 4. Beantwortung der Fragen bezüglich der Verwendung von Mimesis |

**Persönliches Gespräch zur Aufgabenstellung**

5. Der Inhalt der Aufgabenstellung wird kurz mündlich erläutert

6. Die Testpersonen werden darauf hingewiesen, dass der Entwurf des Algorithmus selbst nicht die Aufgabenstellung ist, sondern die Umsetzung mittels bereits vorhandener Werkzeuge

**Durchführung der Aufzeichnung**

7. Beobachtung des Verhaltens der Testperson während der Aufzeichnung

8. Die Möglichkeit der Kommunikation bleibt dauerhaft bestehen, nur Fragen zu **eclipse** oder **Mimesis** werden beantwortet, auf inhaltliche Fragen kann nicht eingegangen werden

### 7.3.2 Versuchsaufbau

Auf der Webseite des **Mimesis**-Teams können sogenannte Experimente angelegt werden. Diese bestehen aus Fragebögen und Aufzeichnungen. Hier kann auch ein komplettes Bundle bestehend aus **eclipse** und dem Plugin heruntergeladen werden, welches auch die Aufzeichnung direkt in die Datenbank des Servers zulässt und entsprechend vorkonfiguriert ist. Das von **Mimesis** erzeugte Bundle enthält ein **eclipse**-IDE, den Workspace mit der Aufgabenstellung und das Plugin, welches direkt in **eclipse** integriert ist. Den Teilnehmern stand es hierbei frei, an welchem Ort gearbeitet wird, da die Datenaufzeichnung remote auf dem jeweiligen System der Testperson durchgeführt wurde.

### 7.3.3 Fragebogen

Auf der **Mimesis**-Webseite kann jeweils ein Link für die Teilnahme an einem Fragebogen erzeugt werden und nach erfolgter Bearbeitung wird der Testperson ein spezifischer Download angeboten mit welchem das entsprechend vorkonfigurierte Bundle direkt heruntergeladen werden kann. Der Fragebogen ist im Anhang D hinterlegt und enthält Fragen über demographische Daten, zur Vorbildung und Kenntnisstand im Umgang mit Java und Verschlüsselungsalgorithmen. Der Bildschirminhalt der Teilnehmer wurde während der Bearbeitung überprüft und die Arbeitsweise zusätzlich zu der Aufzeichnung analysiert.

### 7.3.4 Datenschutzerklärung

Den Teilnehmern wurde eine Datenschutzerklärung vorgelegt, die im Anhang B einsehbar ist. Die Datenschutzerklärung wurde zusätzlich zum Bundle ausgeliefert. Innerhalb der Datenschutzvereinbarung werden die Teilnehmer darauf hingewiesen in welcher Form ihre Daten verwendet werden und inwiefern ein Rücktritt von der Verwendung möglich ist. Die Erklärung wurde mit der Möglichkeit einer digitalen Unterschrift ausgestattet.

## 7.4 Aufgabenstellung

Für die Studie und den damit einhergehenden Akzeptanztest wurde eine Aufgabenstellung gewählt, welche spezifisch darauf ausgelegt ist, eine moderate Komplexität und dennoch die Notwendigkeit der Verwendung externer Quellen zu beinhalten. Dieses Vorgehen ermöglicht es, die aufgezeichneten Daten in einem überschaubaren Rahmen zu halten und gleichzeitig den Kern der Fragestellung dieser Arbeit zu untersuchen.

Im Sinne der Anlehnung an einen sicherheitsrelevanten Kontext soll die Verschlüsselung eines beliebigen Textes mit AES in Java implementiert werden. Eine AES Verschlüsselung entspricht dem Stand der Technik [21] und bietet somit einen angemessenen Ansatz für die Beantwortung der Frage, wie Softwareentwickler im Sicherheitsbereich mit externen Quellen arbeiten. Durch die Verbreitung der Verschlüsselungsart sind jedoch in den meisten modernen Programmiersprachen bereits Komponenten implementiert, die dem Entwickler die Arbeit erleichtern und die dafür sorgen, dass eine Fülle an Quellen verfügbar ist, was der Beantwortung der Forschungsfrage entgegen kommt und gleichzeitig den Aufwand für die Testpersonen vermindert. Die Aufgabe ist dennoch so gewählt, dass ihre Lösung vermutlich nicht

zu dem Wissen gehört, dass ein durchschnittlicher Softwareentwickler im Alltag nutzt und daher angenommen werden kann, dass die Wahrscheinlichkeit für die Verwendung externer Quellen recht hoch ist. Eine AES Verschlüsselung kann des Weiteren auf einige verschiedene Arten entworfen werden, die alle dasselbe Ziel erreichen, jedoch zum Beispiel durch die Verwendung von entweder dynamischen oder statischen Schlüsseln eine unterschiedliche Sicherheit bieten. Die verfügbaren Quellen zeigen Implementierungen in beiden Varianten.

Während der Umsetzung der Aufgabenstellung kann nachvollzogen werden, wann der entsprechende Entwickler bestimmte Quellen aufgerufen hat und wie sich diese auf seine Implementierung ausgewirkt haben. Die Aufgabe, welche die Testpersonen bearbeitet haben, wurde im Anhang A hinterlegt. Eine mögliche Lösung kann im Anhang C eingesehen werden.

Ein Lösungsansatz wird im Rahmen dieser Arbeit dann als sicher eingestuft, wenn die Vorgabe von 128 Bits für die Schlüsselgröße eingehalten, ein dynamischer Schlüssel verwendet und die korrekte AES Implementierung gewählt wurde.

## 7.5 Testpersonen

**Hinweis zu den Aufzeichnungen**

Die Aufzeichnungen der Studie befinden im Verzeichnis **Studienaufzeichnungen**.

Es wurden insgesamt 13 Testpersonen untersucht und ihre Interaktion mit der IDE und den Quellen aufgezeichnet. Die Testpersonen teilen sich auf in 9 Personen, die hauptberuflich als Softwareentwickler tätig sind und 4 Personen, die einen informationstechnischen Studiengang besuchen. Alle Personen befinden sich in einem Alter zwischen 20 und 30 Jahren, wobei der größte Teil älter als 25 ist.

# 8 Ergebnisse

Von den 13 Testpersonen konnten 8 eine Lösung implementieren und 4 dieser 8 Teilnehmer erzeugten eine Lösung, die nicht mit einem statischen Schlüssel arbeitet und somit nach den selbst gewählten Kriterien als sicher eingestuft werden kann. Es konnten 5 Personen die Aufgabe nicht erfolgreich bearbeiten, wobei 2 der Personen angaben, dass sie nach einer Stunde Bearbeitungszeit nicht weiterarbeiten möchten, die anderen 3 haben aufgrund technischer Schwierigkeiten die Bearbeitung abgebrochen. Hierbei stießen sie auf den **Tag Mismatch**-Error, welchen einige der Testpersonen beheben konnten, doch diese 3 Kandidaten fanden keine Lösung. .

Alle Testpersonen, die an dieser Studie teilgenommen haben, starteten die Bearbeitung nach Durchsicht der Aufgabenstellung sofort mit einer Websuche über die Suchmaschine **Google**. Diesen Weg wählten die Testpersonen von selbst.

Auffällig war auch die Ähnlichkeit der Suchbegriffe, mit welchen eine Websuche gestartet wurde. Die drei wichtigsten Schlüsselworte waren hier „**java aes encryption**“, welche teilweise von Zusätzen wie „**example**“ oder auch „**gcm**“ begleitet wurden.

## 8.1 TagMismatchError

Ein Hindernis, das die Testpersonen überwinden mussten, war im Zuge dieser spezifischen Aufgabenstellung der sogenannte **TagMismatchError**, welcher verursacht wird, wenn der Dekodierungsalgorithmus nicht nur auf den Cipher-Text, sondern die gesamte verschlüsselte Nachricht angewendet wird. Der verschlüsselte Text wurde hierbei nicht um den voranstehenden Tag bereinigt, welcher Informationen über den nachfolgenden Text beinhaltet. Zu diesem spezifischen Fehler finden sich nur wenige Lösungsansätze. Eine Person, die sich mit dem entsprechenden Fehler konfrontiert sieht, findet zwar leicht Hinweise auf die Ursache des Problems, jedoch gestaltet sich die Lösung als inhaltliche Veränderung des Codes.

Die Fehlermeldung resultiert also aus einem logischen Fehler im Code. Scheinbar wurde den Testpersonen, welche mit dem Fehler konfrontiert wurden, während des Versuchs eine Lösung zu finden klar, dass das Problem nur durch eine neue Strukturierung ihrer Lösung behoben werden kann, was die genutzte Quelle als nicht hilfreich dargestellt hat. Die Testpersonen haben vermutlich nicht damit gerechnet, nach Anwenden der gefundenen Lösung den Code selbst erweitern und verändern zu müssen, um die gewünschte Funktionalität zu ermöglichen.

## 8.2 Gültigkeitsbereich

Die Testpersonen bestanden aus angehenden Softwareentwicklern und solchen, die bereits beruflich in diesem Feld tätig sind. Die Anzahl direkter Experten in dem Themengebiet war mit einer einzelnen Person gering. Die Anzahl der weniger erfahrenen Personen war verhältnismäßig hoch.

Es wurde eine Aufgabenstellung gewählt, welche spezifisch darauf ausgelegt war, dass die meisten Entwickler keine vollständige Implementierung von selbst erzeugen können, sondern auf externe Quellen zurückgreifen müssen. Dies wurde im Zuge dieser Studie auch eingehalten. Es ist jedoch nicht klar, ob die Testpersonen weniger auf externe Quellen zurückgegriffen hätten, wenn die Aufgabe mit einer anderen Wertung im Wortlaut gestellt worden wäre. In dieser Studie wurde den Testpersonen explizit mitgeteilt, dass es um ihr Verhalten im Bezug auf die Nutzung von Quellen geht und dass die Verwendung dieser ausdrücklich erlaubt ist – es bestünde also Grund zur Annahme, dass unter anderen Prämissen die Testpersonen nicht mit dem Beginn der Arbeit zu so großer Anzahl direkt zur Websuche gegriffen hätten.

Die Testpersonen wurden des Weiteren während der Bearbeitung der Aufgabe beobachtet, diese Tatsache kann dafür gesorgt haben, dass das Verhalten der Testpersonen sich anders gestaltet als in einer natürlichen, unbeobachteten Umgebung.

# 9 Diskussion

**F1: Welchen Einfluss haben Dokumentation und externe Quellen auf die Softwarequalität und wie werden sie während der Entwicklung einer Software oder der Umsetzung einer Softwareanforderung verwendet?**

Während einige Teilnehmer gezielt nach offizieller Dokumentation seitens **Oracle** gesucht haben, wählten die meisten direkt die ersten Treffer in ihrer bevorzugten Suchmaschine und suchten nach eigener Aussage systematisch nach ihnen logisch erscheinendem Quellcode. Es wurde allgemein wenig Wert auf beschreibenden Text gelegt und dieser zumeist überflogen. Hierbei lies sich im Rahmen der Aussagen der Testpersonen dieser Studie ein erstes Muster erkennen: Je besser sich ein Entwickler mit einer Sprache auskennt, desto eher wird im Quellcode nach „Erklärungen“ gesucht, statt den vor- oder nachgestellten Text in der jeweiligen Quelle zu studieren. Nach Aussagen der Testpersonen, die sich mehr auf den Quellcode statt auf die angefügten Erklärungen konzentriert haben, war der gezeigte Quellcode aussagekräftig genug, um Lösungsansätze zu finden.

**F2: Zu welchen Zeitpunkten greift ein Entwickler auf externe Quellen zurück, um Wissen zu erlangen und in welcher Form wird nach Quellen gesucht?**

Es konnte beobachtet werden, dass Testpersonen, welche sich selbst als erfahrener im Umgang mit Java und Programmierung im Allgemeinen eingeschätzt haben, viel spezifischer bei der Auswahl ihrer Quellen waren. Diejenigen Testpersonen, welche nach eigener Einschätzung weniger erfahren sind, haben sich vor allem auf die von der Suchmaschine angebotenen Blogs mit vollständigen Beispielen gestützt und haben versucht, diese so vollständig wie möglich zu übernehmen, während die erfahreneren Entwickler vor allem auf StackOverflow oder offizielle Dokumentationen vertraut haben, welche zum Beispiel durch Oracle bereitgestellt wurde.

**F3: Welche Informationen werden zur Bewältigung sicherheitsrelevanter Aspekte in der Programmierung benötigt und aus welchen Quellen werden sie beschafft?**

Ohne Ausnahme wurde von den Testpersonen ausgesagt, dass die offizielle Dokumentation von Oracle unzureichend war und mit den dort gelieferten Informationen nicht gearbeitet werden konnte. Alle außer zwei Kandidaten haben diese nach eigener Aussage erst gar nicht aufgerufen, da sie bereits Erfahrungen gesammelt hatten und die Qualität zumeist unzureichend ist, die anderen beiden wollten sich zunächst auf die offizielle Dokumentation beziehen, aber nach eigener Aussage haben sie irgendwann aufgegeben. Insbesondere Testperson 3 war in diesem Sinne auffällig, da sie unverhältnismäßig (in Relation zu anderen Teilnehmern) viel Zeit mit der offiziellen Dokumentation verbracht hat und nur eine weitere Quelle ausprobiert hat, bevor die Bearbeitung abgebrochen wurde.

Ein großer Teil der Testpersonen stieß auf den **Tag-Mismatch**-Error. Es fiel den Testpersonen nach eigener Auffassung schwer, Lösungen für diesen Fehler zu finden, da die Fehlermeldung wenig Aussagekraft mitbringt. Dem Nutzer wird hier nur ein **TagMismatchError** angezeigt, es wird jedoch kaum begründet, womit dieser zusammenhängt.

**F4: Können Quellen in sicherheitsrelevanten Anwendungsgebieten als vertrauenswürdig eingestuft werden? Welche Risiken bestehen?**

Eine Auffälligkeit dieser Studie besteht darin, dass vier Testpersonen einen statischen Schlüssel zur Verschlüsselung verwendet haben. In den Quellen, welche mit einem statischen Schlüssel gearbeitet haben, wurde nicht darauf hingewiesen, dass dieser Umstand die Sicherheit der dargestellten Lösung negativ beeinflusst.

Drei der nach eigener Einschätzung thematisch weniger versierten Testpersonen gerieten an den Blog „axxg.de“, welcher ein veraltetes Beispiel für eine AES Verschlüsselung in Java anbietet. Dies führte dazu, dass diese Testpersonen ihren Lösungsweg nicht weiter führen können, da das Beispiel Klassen verwendet, die es so nicht mehr gibt oder zu einer Suche nach möglichen Alternativen, mit denen die fehlenden Klassen ersetzt werden können. Dies führte bei zwei Testpersonen zum Abbruch der Bearbeitung.

Die bereitgestellten Beispiele waren insofern veraltet, dass Klassen der Java-Implementierung von Oracle verwendet wurden, welche in der in **eclipse** enthaltenen Implementierung von OpenJDK zwar auch existieren, jedoch in einem anderen Paketpfad abgelegt wurden. Einigen Testpersonen fiel es schwer, diese Klassen zu ersetzen. Der Umstand wurde zwar in den Kommentaren des Blogeintrages angesprochen und es wurde dort auch die entsprechende Lösung präsentiert, jedoch wurde der eigentliche Blogeintrag nicht editiert und die veraltete Information erhalten. Hierdurch entsteht die Vermutung, dass nicht mehr aktuelle Informationen den Arbeitsfluss eines Softwareentwicklers negativ beeinflussen können.

Die Frage, wie diese Informationen aktuell gehalten werden können, um diese Problematik zu beheben lässt sich nicht pauschal beantworten. Im vorliegenden Beispiel wurde die aktualisierte Lösung von einer dritten Person und nicht vom Autor des originalen Artikels in den Kommentaren zur Verfügung gestellt, wobei es gerade hier leicht wäre, eine solche Lösung einzuarbeiten. Der Blog wird bis dato weitergeführt und neue Artikel erscheinen.

# 10 Fazit

Veraltete oder falsche Informationen stellten in dieser Studie für die Testpersonen erwartungsgemäß nach eigener Aussage das größte Problem dar. Die nach ihrer Einschätzung erfahreneren Entwickler zeichneten sich vor allem durch eine deutlich zielgerichtetere Suche aus, wobei sie auch begründeten, dass einige Quellen aus Erfahrung bessere Informationen liefern als andere. Besonders problematisch fielen fehlerhafte Beispiele bei den unerfahreneren Testpersonen auf. Diesen fiel es äußerst schwer, die entsprechenden Fehler zu beheben und sie hielten sich während der Bearbeitung verhältnismäßig lange damit auf, nach Unterschieden zwischen dem Beispiel und ihrem Code zu suchen oder den Fehler in ihrer eigenen Lösung zu vermuten.

Die nach eigener Aussage weniger erfahrenen Entwickler hatten somit augenscheinlich vor allem das Problem, dass sie mehr Vertrauen in die Richtigkeit und Vollständigkeit der gefundenen Lösungen gesteckt haben, als in ihre eigenen Fähigkeiten, während die sich selbst als erfahren einschätzenden Personen kleinere Fehler schnell erkannten und auch bewusst nach weiterführenden Quellen suchten. Hier entsteht die Annahme, dass die weniger erfahrenen Testpersonen nach einer möglichst vollständigen Lösung aus einer Quelle suchten und Probleme damit hatten, die Komplexität zu beherrschen, insbesondere dann, wenn die Lösung unvorhergesehene Fehler erzeugt hat.

Die Qualität der Quellen selbst wurde vor allem durch die Aktualität und Lesbarkeit der Quellen bestimmt, was auch der Annahme aus anderen Arbeiten entspricht [39]. Solche Quellen, die eine scheinbare Überfülle an Code und Text lieferten und damit zunächst als besonders ausführlich bewertet werden könnten, stellten sich als zu komplex heraus und solche Quellen, die wichtige Informationen ausließen, um die Komplexität zu verringern sorgten für weniger sichere Lösungen. Quellen, deren Informationen veraltet waren und die nicht aktualisiert wurden, stellten außerdem ein Problem dar, da die Nutzbarkeit mit aktueller Grundsoftware nicht ausreichend gewährleistet oder zumindest gekennzeichnet war, insbesondere weniger erfahrene Personen hatten hier Probleme.

Die Komplexität von Quellen selbst lies sich vor allem der durch den Blog **MKYong** bereitgestellten Informationen gut in Relation setzen, da dort ein sehr komplexe Beispiele gezeigt wurden, welche eine Vielzahl von Anwendungsfällen abdecken, von den Testpersonen jedoch meistens nur flüchtig überflogen und nach eigener Aussage aufgrund der schieren Menge an Quellcode nicht zur Lösung des gestellten Problems genutzt wurden.

Eine weitere Schwierigkeit während der Bearbeitung dieser spezifischen Aufgabe innerhalb dieser Studie schien für die weniger erfahrenen Testpersonen auch in der Komplexität von Aufgabe und Lösung zu liegen. Diese Annahme entsteht aus der Beobachtung heraus, dass diese Entwickler während der Bearbeitung auf einen gefundenen Lösungsansatz fokussiert waren und sich von diesem nur schwer lösen konnten. Der eingangs beschriebene **TagMismatchError** trat erst auf, wenn bereits eine Lösung geschaffen wurde, welche kompilierte und durch den mitgelieferten **UnitTest** ausgeführt werden konnte, was relativ betrachtet zu einem fortgeschrittenen Zeitpunkt der Bearbeitung der Fall war.

Zu Beginn dieser Arbeit wurde unter anderem von einer „Expertenblindheit“ gegenüber neuen Lösungsansätzen gesprochen. Diese konnte in der hier gestellten Aufgabe nur teilweise beobachtet werden. Vor allem wurde von den Testpersonen mit einer höheren Selbsteinschätzung ausgesagt, dass sie spezifischen Quellen eher vertrauen und andere Quellen gar nicht erst aufrufen, da sie in der Vergangenheit negative Erfahrungen gemacht haben – die Beispiele des Blogs „MKYong“ wurden als qualitativ weniger hochwertig eingestuft. Primär wurde auf „StackOverflow“ vertraut, nachdem die Dokumentation von **Oracle** ausgeschlossen wurde. Dies stellt an sich noch keine Blindheit dar, zeigt jedoch eine Fokussierung auf bestimmte Quellen, basierend auf Erfahrungen aus der Vergangenheit.

# 11 Perspektive und Ausblick

Die Studie dieser Arbeit ist von qualitativer Natur und ermöglicht das Festellen erster Pfade, welche in weiterführenden quantitativen Studien vertieft werden können. Hierbei wäre vor allem zu beobachten, ob sich die eingangs präsentierten Vermutungen auch in anderen Szenarien nachstellen lassen, da die Qualität der Quellen im Bereich der AES Verschlüsselungen nicht repräsentativ für das gesamte Feld der Softwaresicherheit ist.

Die Testpersonen haben des Weiteren gesammelt ausgesagt, dass die Arbeit mit dem Plugin und der präparierten Entwicklungsumgebung unproblematisch war. Als Störfaktor für die Akzeptanz der Testpersonen spezifisch gegenüber der Erweiterung zum Mitschnitt der Browseraktivität konnte herausgestellt

werden, dass die Testpersonen unzufrieden mit der Funktionalität des von **eclipse** gelieferten internen Browsers waren. Insbesondere die fehlende Möglichkeit, mehrere Tabs aufzurufen, wurde als Problem eingestuft. Eine Testperson umging diese Problematik indem sie die URLs verschiedener Suchergebnisse in einer gesonderten Datei ablegte. Dies lässt sich auch anhand der Aufzeichnung nachvollziehen.

Insgesamt wurde dieser Umstand als problematisch wahrgenommen, hat aber aufgrund der übersichtlichen Komplexität der Aufgabenstellung in dieser Studie keine Hindernisse erzeugt. Bei einer komplexeren Arbeitsaufgabe kann dies die Arbeit der Testpersonen stark einschränken.

# 12 Referenzen


[1] J. Bayer und D. Muthig. „A view-based approach for improving software documentation practices“. In: *13th Annual IEEE International Symposium and Workshop on Engineering of Computer-Based Systems (ECBS'06)*. IEEE, 2006. DOI: 10.1109/ecbs.2006.18.

[2] Saman Bazrafshan und Rainer Koschke. „Effect of Clone Information on the Performance of Developers Fixing Cloned Bugs“. In: *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2014. DOI: 10.1109/scam.2014.10.

[3] L.C. Briand. „Software documentation: how much is enough?“ In: *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.* IEEE Comput. Soc. DOI: 10.1109/csmr.2003.1192406.

[4] Justin Cappos u. a. „Vulnerabilities as Blind Spots in Developer's Heuristic-Based Decision-Making Processes“. In: *Proceedings of the 2014 workshop on New Security Paradigms Workshop - NSPW '14*. ACM Press, 2014. DOI: 10.1145/2683467.2683472.

[5] Yoonsik Cheon und Gary T. Leavens. „A Simple and Practical Approach to Unit Testing: The JML and JUnit Way“. In: *ECOOP 2002 — Object-Oriented Programming*. Springer Berlin Heidelberg, 2002, S. 231–255. DOI: 10.1007/3-540-47993-7_10.

[6] Shigeru Chiba. „Load-Time Structural Reflection in Java“. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, S. 313–336. DOI: 10.1007/3-540-45102-1_16.

[7] Steven Clarke. „Describing and Measuring API Usability with the Cognitive Dimensions“. In: *Dimensions of Notations 10th Anniversary Workshop* (2005). URL: https://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/workshop2005/Clarke_position_paper.pdf.

[8] Anne Edmundson u. a. „An Empirical Study on the Effectiveness of Security Code Review“. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, S. 197–212. DOI: 10.1007/978-3-642-36563-8_14.

[9] Uwe Flick. „Qualitative Research - State of the Art“. In: *Social Science Information* 41.1 (2002), S. 5–24. DOI: 10.1177/0539018402041001001.

[10] Golara Garousi. *A Hybrid Methodology for Analyzing Software Documentation Quality and Usage*. 2012. DOI: 10.11575/PRISM/24760.

[11] Golara Garousi u. a. „Evaluating usage and quality of technical software documentation“. In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE '13*. ACM Press, 2013. DOI: 10.1145/2460999.2461003.

[12] Valentina Grigoreanu u. a. „End-user debugging strategies“. In: *ACM Transactions on Computer-Human Interaction* 19.1 (2012), S. 1–28. DOI: 10.1145/2147783.2147788.

[13] Anja Guzzi u. a. „Collective Code Bookmarks for Program Comprehension“. In: *2011 IEEE 19th International Conference on Program Comprehension*. IEEE, 2011. DOI: 10.1109/icpc.2011.19.

[14] Cormac Herley. „So long, and no thanks for the externalities“. In: *Proceedings of the 2009 workshop on New security paradigms workshop - NSPW '09*. ACM Press, 2009. DOI: 10.1145/1719030.1719050.

[15] S. Islam und P. Falcarin. „Measuring security requirements for software security“. In: *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*. IEEE, 2011. DOI: 10.1109/cis.2011.6169137.

[16] Andrew J. Ko u. a. „An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks“. In: *IEEE Transactions on Software Engineering* 32.12 (2006), S. 971–987. DOI: 10.1109/tse.2006.116.

[17] Franz Lehner. „Quality control in software documentation based on measurement of text comprehension and text comprehensibility“. In: *Information Processing & Management* 29.5 (1993), S. 551–568. DOI: 10.1016/0306-4573(93)90079-s.

[18] T.C. Lethbridge, J. Singer und A. Forward. „How software engineers use documentation: the state of the practice“. In: *IEEE Software* 20.6 (2003), S. 35–39. DOI: 10.1109/ms.2003.1241364.

[19] Walid Maalej u. a. „On the Comprehension of Program Comprehension“. In: *ACM Transactions on Software Engineering and Methodology* 23.4 (2014), S. 1–37. DOI: 10.1145/2622669.

[20] Tim Mackinnon, Steve Freeman und Philip Craig. „Endo-testing: unit testing with mock objects“. In: *Extreme programming examined* (2000), S. 287–301. URL: https://www2.ccs.neu.edu/research/demeter/related-work/extreme-programming/MockObjectsFinal.PDF.

[21] Prerna Mahajan und Abhishek Sachdeva. „A study of encryption algorithms AES, DES and RSA for security“. In: *Global Journal of Computer Science and Technology* (2013). URL: https://computerresearch.org/index.php/computer/article/view/272.

[22] G. McGraw. „Software assurance for security“. In: *Computer* 32.4 (1999), S. 103–105. DOI: 10.1109/2.755011.

[23] Michael Meng, Stephanie Steinhardt und Andreas Schubert. „Application Programming Interface Documentation: What Do Software Developers Want?“ In: *Journal of Technical Writing and Communication* 48.3 (2017), S. 295–330. DOI: 10.1177/0047281617721853.

[24] Günter Mey und Katja Mruck, Hrsg. *Handbuch Qualitative Forschung in der Psychologie*. Springer Fachmedien Wiesbaden, 2020. DOI: 10.1007/978-3-658-26887-9.

[25] Katja Mruck und Franz Breuer. „Subjectivity and Reflexivity in Qualitative Research—The FQS Issues“. In: *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 4.2 (2003). ISSN: 1438-5627. DOI: 10.17169/fqs-4.2.696. URL: https://www.qualitative-research.net/index.php/fqs/article/view/696.

[26] M.P. O'Brien und J. Buckley. „Modelling the Information-Seeking Behaviour of Programmers - An Empirical Approach“. In: *13th International Workshop on Program Comprehension (IWPC'05)*. IEEE. DOI: 10.1109/wpc.2005.24.

[27] Adrian O'Dowd. „Major global cyber-attack hits NHS and delays treatment“. In: *BMJ* (2017), j2357. DOI: 10.1136/bmj.j2357.

[28] Daniela Oliveira u. a. „It's the psychology stupid“. In: *Proceedings of the 30th Annual Computer Security Applications Conference on - ACSAC '14*. ACM Press, 2014. DOI: 10.1145/2664243.2664254.

[29] Reinhold Plosch, Andreas Dautovic und Matthias Saft. „The Value of Software Documentation Quality“. In: *2014 14th International Conference on Quality Software*. IEEE, 2014. DOI: 10.1109/qsic.2014.22.

[30] P.J.G. Ramadge und W.M. Wonham. „The control of discrete event systems“. In: *Proceedings of the IEEE* 77.1 (1989), S. 81–98. DOI: 10.1109/5.21072.

[31] Tobias Roehm u. a. „How Do Professional Developers Comprehend Software?“ In: *Proceedings of the 34th International Conference on Software Engineering*. ICSE '12. Zurich, Switzerland: IEEE Press, 2012, 255–265. ISBN: 9781467310673.

[32] J. Sillito, G.C. Murphy und K. De Volder. „Asking and Answering Questions during a Programming Change Task“. In: *IEEE Transactions on Software Engineering* 34.4 (2008), S. 434–451. DOI: 10.1109/tse.2008.26.

[33] Sergio Cozzetti B. de Souza, Nicolas Anquetil und Káthia M. de Oliveira. „A study of the documentation essential to software maintenance“. In: *Proceedings of the 23rd annual international conference on Design of communication documenting & designing for pervasive information - SIGDOC '05*. ACM Press, 2005. DOI: 10.1145/1085313.1085331.

[34] National Institute for Standards und Technology. *CWE Over Time*. URL: https://nvd.nist.gov/vuln/visualizations/cwe-over-time.

[35] StatCounter. *Search Engine Market Share Worldwide*. 2019-2020. URL: https://gs.statcounter.com/search-engine-market-share.

[36] Rahul Telang und Sunil Wattal. „Impact of Software Vulnerability Announcements on the Market Value of Software Vendors - An Empirical Investigation“. In: *SSRN Electronic Journal* (2005). DOI: 10.2139/ssrn.677427.

[37] Gias Uddin und Martin P. Robillard. „How API Documentation Fails“. In: *IEEE Software* 32.4 (2015), S. 68–75. DOI: 10.1109/ms.2014.80.

[38] Anna Wingkvist u. a. „A Metrics-Based Approach to Technical Documentation Quality“. In: *2010 Seventh International Conference on the Quality of Information and Communications Technology*. IEEE, 2010. DOI: 10.1109/quatic.2010.88.

[39] Junji Zhi u. a. „Cost, benefits and quality of software development documentation: A systematic mapping“. In: *Journal of Systems and Software* 99 (2015), S. 175–198. DOI: 10.1016/j.jss.2014.09.042.

# Anhang

## A Aufgabenstellung

## AES Verschlüsselung

Implementieren Sie die Funktionen der vorgegebenen Klasse, um eine AES Verschlüsselung anzuwenden, welche die zugehörigen Erwartungen der Unit-Tests erfüllt. Die AES Verschlüsselung soll mit einer Schlüsselgröße von 128 Bits arbeiten, wie auch in der Klasse als statische Variable hinterlegt.

Die Nutzung von Quellen aus dem Internet ist ausdrücklich erlaubt - sofern der von Eclipse bereitgestellte Browser verwendet wird.

Code 1: AESEncoder.java

```
1  package de.uni_bremen.tihansen.aes_aufgabe;
2
3  public class AESEncoder {
4
5      public static final int AES_KEY_SIZE = 128; // in bits
6      public static final int GCM_NONCE_LENGTH = 12; // in bytes
7      public static final int GCM_TAG_LENGTH = 16; // in bytes
8
9
10     /**
11      * Encrypt a string-value using AES.
12      *
13      * @param toEncrypt The value to encrypt
14      * @return The encoded value
15      */
16     public static String encrypt(String toEncrypt) {
17         throw new UnsupportedOperationException();
18     }
19
20     /**
21      * Decrypt a string-value using AES.
22      *
23      * @param toDecrypt The value to decrypt
24      * @return The decrypted value
25      */
26     public static String decrypt(String toDecrypt) {
27         throw new UnsupportedOperationException();
28     }
29
30 }
```

Code 2: AESEncoderTest.java

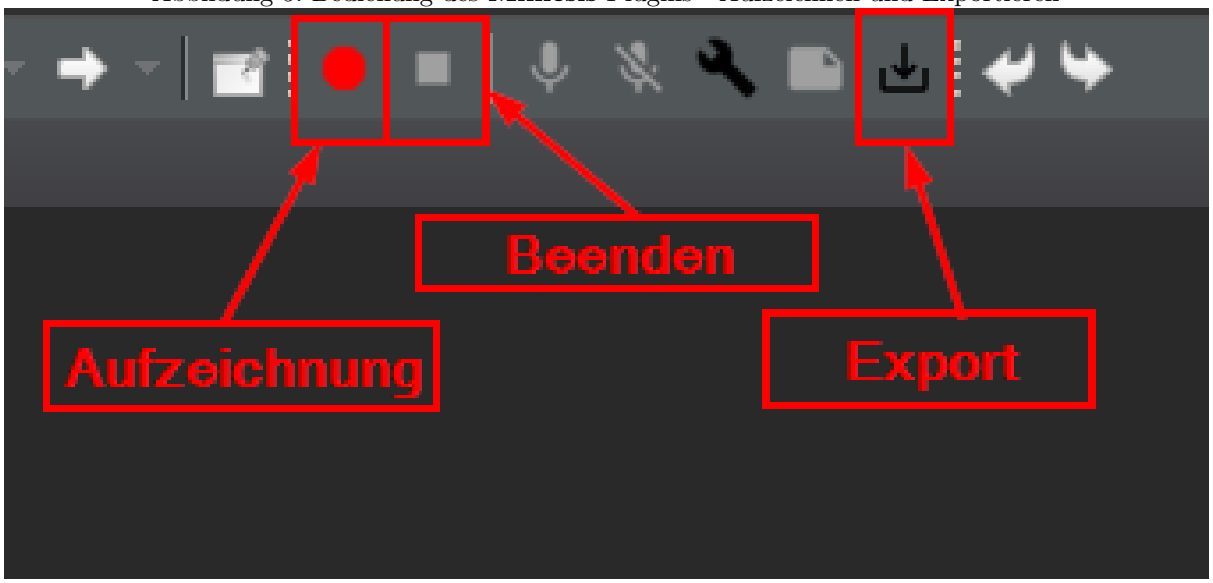
```
package de.uni_bremen.tihansen.aes_aufgabe;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class AESEncoderTest {

    private static String decodedMessage = "Some message to encode to"
            + " ensure the encryption is working properly!";

    @Test
    void test() {
        String encrypted = AESEncoder.encrypt(decodedMessage);

        String decrypted = AESEncoder.decrypt(encrypted);

        assertEquals(decrypted, decodedMessage);
    }
}
```

Die Aufzeichnung erfolgt über das **Mimesis**-Plugin in **eclipse**. Um dieses Plugin zu installieren, muss zunächst die mit dieser Aufgabenstellung ausgelieferte *.jar-Datei in dem **dropins**-Verzeichnis im **eclipse**-Hauptverzeichnis abgelegt werden. Damit das Plugin ausgeführt wird, ist eventuell ein Neustart von **eclipse** notwendig.

Die Aufzeichnung durch das **Mimesis**-Plugin kann über den roten Button in der Taskleiste in **eclipse** gestartet werden. Durch einen Klick auf das schwarze Quadrat daneben kann die Aufnahme anschließend beendet werden.

Ein Export der Daten ist über den Button auf der rechten Seite möglich.

Abbildung 5: Bedienung des **Mimesis**-Plugins - Aufzeichnen und Exportieren



Der interne Eclipse-Browser kann wie folgt aufgerufen werden:

Abbildung 6: Window - Show View - Other...

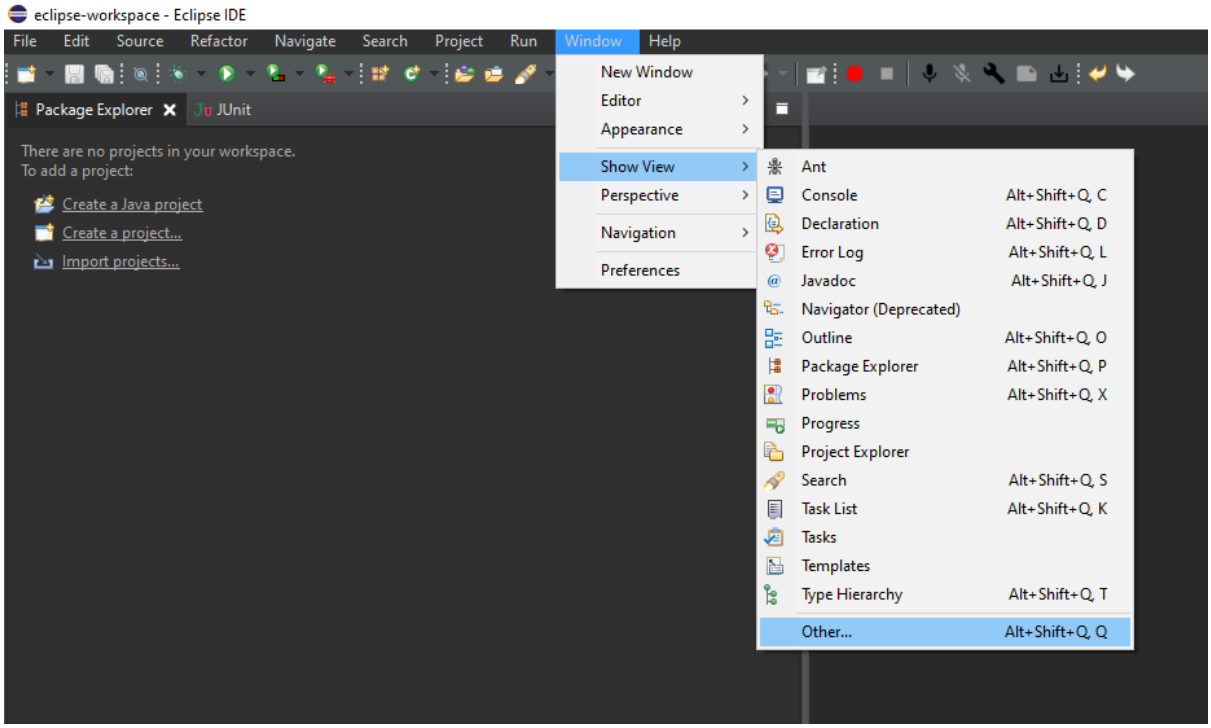



Abbildung 7: Über die Suchleiste kann nach **Internal Web Browser** gesucht werden

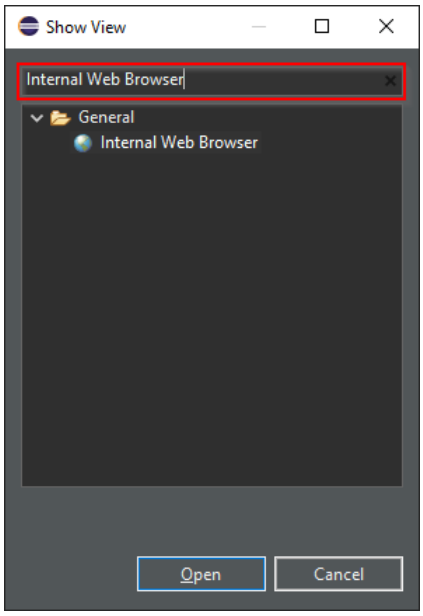
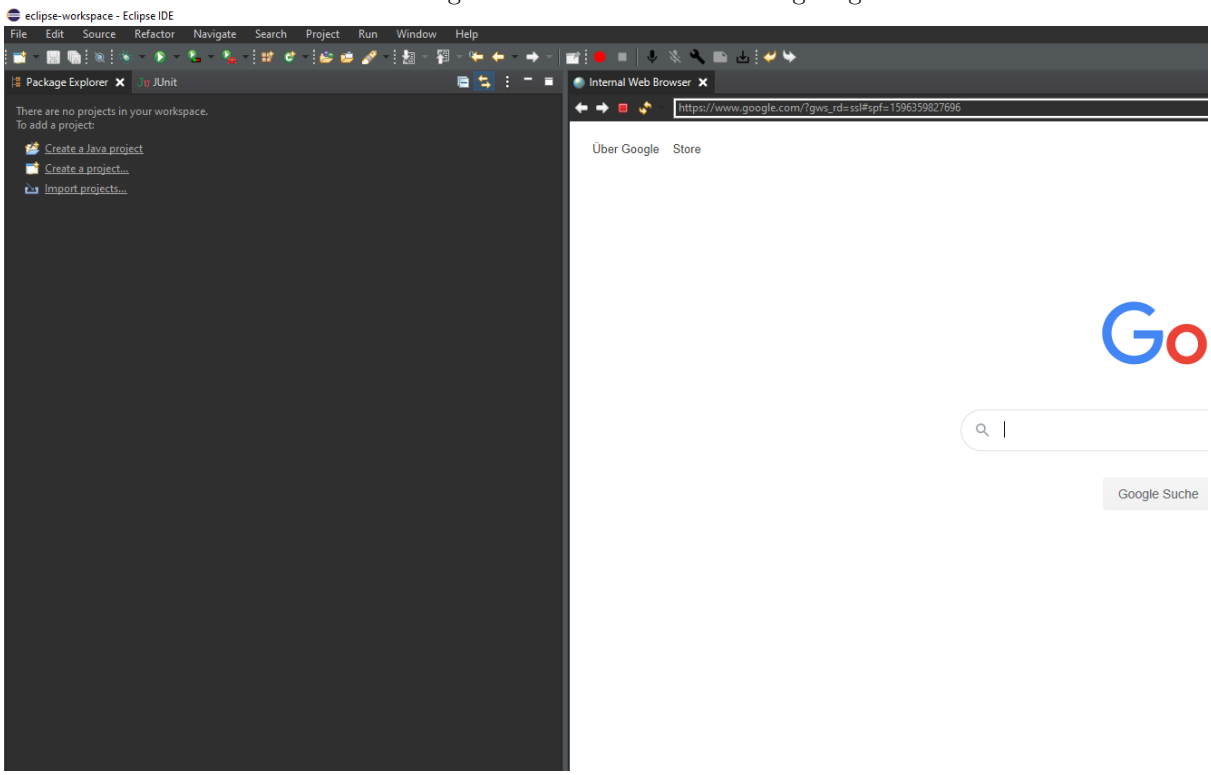


Abbildung 8: Interner Browser wird angezeigt



# B DSGVO / Datenschutzerklärung

## Datenschutzerklärung

### Teilnahme an der Studie / Informationen zum Datenschutz gemäß Art. 13 DSGVO

Hiermit informieren wir die Teilnehmer und die Teilnehmerinnen der Studie über die Verarbeitung ihrer personenbezogenen Daten im Rahmen der Teilnahme an der Studie und Ausarbeitung dieser Abschlussarbeit.

Im Zuge dieser Studie werden Daten aufgezeichnet, erfasst und gespeichert. Die Daten werden anonymisiert und ausschließlich im Rahmen dieser Studie verwendet.

#### 1. Verantwortlich für die Datenverarbeitung und Kontaktdaten

**Tim Hansen**
Ahrensburger Straße 71
22041 Hamburg

tihansen@uni-bremen.de

Bereich Wirtschaftsinformatik
Universität Bremen

#### 2. Zweck der Datenverarbeitung

Die erfassten Daten werden ausschließlich im Rahmen der Studie dieser Abschlussarbeit verwendet. Die Daten werden Dritten ausschließlich in anonymisierter Form zur Verfügung gestellt.

Während der Studie werden nur Daten innerhalb der bereitgestellten Entwicklungsumgebung und im Bezug auf die Bearbeitung der gestellten Aufgabe erhoben. Es werden keine weiteren Daten über das System oder den Nutzer erhoben.

Die während der Umfrage erhobenen Daten werden nur zur Einordnung der Ergebnisse verwendet und Dritten nur in anonymisierter Form zur Verfügung gestellt.

#### 3. Daten, die verarbeitet werden

Folgende Daten werden erfasst und verarbeitet:

- Alter und Bildungsstand
- Berufsbild
- Während der Aufzeichnung gesammelte Daten (ausschließlich aus der Entwicklungsumgebung)
- Antworten auf Fragestellungen

Weitere, personenbezogene Daten sind für die Teilnahme an der Studie nicht nötig und werden nicht erhoben.

#### 4. Widerruflichkeit der Einwilligung

Die Einwilligung kann zu jedem Zeitpunkt bis zur Publikation der Arbeit zurückgezogen werden. Die Daten und Aufzeichnungen werden vollständig gelöscht und aus der Arbeit entfernt.

**5. Empfänger der personenbezogenen Daten**

Die Daten werden teilweise durch das **Mimesis**-Plugin auf dem Zielsystem des Teilnehmers oder der Teilnehmerin erfasst und über eine gesicherte Verbindung an eine Datenbank auf den Systemen der Universität Bremen übertragen.

Innerhalb eines gesicherten Fragebogens auf den Systemen der Universität Bremen werden Daten erfasst.

Weitere Daten werden im persönlichen Gespräch erfasst.

**6. Ihre Rechte**

Als Teilnehmer existieren folgende Rechte hinsichtlich personenbezogener Daten (entsprechend Artikel 15 bis 21 DSGVO):

- Recht auf Auskunft
- Recht auf Berichtigung
- Recht auf Löschung
- Recht auf Einschränkung der Bearbeitung
- Recht auf Datenübertragbarkeit / Recht auf Erhalt einer Kopie

Zudem haben Sie das Recht der Datenverarbeitung jederzeit zu widersprechen. Sofern noch keine Publikation stattgefunden hat, werden die Daten nicht mehr verarbeitet.

# C Beispiellösung

Code 3: AESEncoderSolved.java

```
package de.uni_bremen.tihansen.aes_aufgabe;

import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;

public class AESEncoderSolved {

    public static final int AES_KEY_SIZE = 128; // in bits
    public static final int GCM_NONCE_LENGTH = 12; // in bytes
    public static final int GCM_TAG_LENGTH = 16; // in bytes

    private static SecretKey key = null;
    private static GCMParameterSpec ivspec = null;

    private static void generateKey()
            throws UnsupportedEncodingException,
               NoSuchAlgorithmException
    {
            SecureRandom random = SecureRandom.getInstanceStrong();
            KeyGenerator keyGen = KeyGenerator.getInstance("AES");
            keyGen.init(AES_KEY_SIZE, random);
            key = keyGen.generateKey();

            final byte[] nonce = new byte[GCM_NONCE_LENGTH];
            random.nextBytes(nonce);
            ivspec = new GCMParameterSpec(GCM_TAG_LENGTH * 8, nonce);
    }

    /**
     * Encrypt a string-value using a given secret with AES.
     *
     * @param toEncrypt The value to encrypt
     * @return The encoded value
     */
    public static String encrypt(String toEncrypt) {
        try
        {
            generateKey();
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "
               SunJCE");
            cipher.init(Cipher.ENCRYPT_MODE, key, ivspec);
            return Base64.getEncoder().encodeToString(
                        cipher.doFinal(toEncrypt.getBytes("UTF-8"))
                    );
        }
        catch (Exception e)
        {
            System.out.println("Error while encrypting: " + e.toString
```

```
                ());
54          }
            return null;
56      }

58      /**
         * Decrypt a string-value using a given secret with AES.
60       *
         * @param toDecrypt The value to decrypt
62       * @return The decrypted value
         */
64      public static String decrypt(String toDecrypt) {
            try
66          {
                Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "
                    SunJCE");
68              cipher.init(Cipher.DECRYPT_MODE, key, ivspec);
                return new String(cipher.doFinal(Base64.getDecoder().decode
                    (toDecrypt)));
70          }
            catch (Exception e)
72          {
                System.out.println("Error while decrypting: " + e.toString
                    ());
74          }
            return null;
76      }

78  }
```

# D Fragebogen

| Welchem Geschlecht gehören Sie an? |
|---|
| Männlich |
| Weiblich |
| Divers |
| N/A |

| Wie alt sind Sie? |
|---|
| Unter 18 |
| 18-25 |
| 26-35 |
| 36-45 |
| Über 45 |

**Welchen Bildungsgrad haben Sie?**

- Noch in der Schule
- Hauptschulabschluss
- Mittlere Hochschulreife
- Abitur
- Bachelor
- Master
- Doktor

**Wie schätzen Sie Ihre Fähigkeiten im Umgang mit Java ein?**

- Unerfahren
- Kaum erfahren
- Mäßig erfahren
- Erfahren
- Experte

**Wie schätzen Sie Ihre Kenntnisse im Bereich von Sicherheitstechniken ein?**

- Unerfahren
- Kaum erfahren
- Mäßig erfahren
- Erfahren
- Experte

**Wie schätzen Sie Ihre Kenntnisse im Bereich von Verschlüsselungstechniken ein?**

- Unerfahren
- Kaum erfahren
- Mäßig erfahren
- Erfahren
- Experte