****

Entwicklungsumgebung

Stand 08.04.2022

# Versionierung

|  | **git** | **svn** | **Mercurial (hg)** |
| --- | --- | --- | --- |
| Charakter- istika | Verteiltes VKS  (jeder Nutzer hat eine lokale Kopie) | Zentralisiertes VKS (alles auf zentralem Server gespeichert) | Verteiltes VKS  (jeder Nutzer hat eine lokale Kopie) |
| Kosten | Kostenlos | Kostenlos | Kostenlos |
| Merkmale | Tool zum Erarbeiten eines version- controlled Workflows. Features: Issue Tracker, Wiki Engines, Sysadmin Tools, Filesystem, ... | Für Unternehmen geeignet, die mehr Kontrolle und Zentralverwaltung bevorzugen | Eher für unerfahrene Nutzer oder kleinere Teams und Projekte. Ähnlich, aber intuitiver als Git Effizient und skalierbar |
| Netzwerk- verbindung | Keine stetige Netzwerkverbindung notwendig | Stetige Netzwerkver- bindung notwendig | Keine stetige Netzwerkverbindung notwendig |
| Zugriffs- rechte | Alle Mitglieder haben die gleichen Rechte | Individuelle Vergabe von Lese- und Schreibrechten auf unterschiedlichen Ebenen | Alle Mitglieder haben die gleichen Rechte |
| Lernkurve | Syntax unintuitiv, daher schwer zu erlernen | Leicht zu lernen, auch Kommandozeile | Sehr leicht zu lernen und user-friendly |
| Erfahrung | Alle Teammitglieder | Keine Erfahrung | Keine Erfahrung |
| Änderungshistorie | Repository und Arbeitskopien enthalten die gesamte Historie | Im Repository komplett, Arbeitskopien enthalten nur neueste Version, leere, umbenannte und verschobene Verzeichnisse werden ohne Historie-Verlust problemlos aufgezeichnet | Repository und Arbeitskopien enthalten die gesamte Historie |
| Vorteile | * Schnell und flexibel * Effektives Branching * Interactive Rebase * Große Community * Built-in in vielen IDEs * Keine Lese- und Schreibrechte * Keine stetige Netzwerkver- bindung notwendig * Gleichzeitiges Arbeiten von mehreren Entwicklern * Lokales Back-Up vom Hauptverzeichnis | * Pfadbasierte Zugriffsrechte bei Bedarf * Zentrale Bündelung vom gesamten Projekt * Vollständige Aufzeichnung von den Strukturen leerer Verzeichnisse * Besserer Umgang mit großen Binärdateien | * Besser geeignet für unerfahrene Nutzer * Besser in Merging im Vergleich zu Git * Branches visuell besser nachvollziehbar * Historie kann man nicht umschreiben * Built-in Webserver * Gute Handhabung von Binärdateien |
| Nachteile | * Keine Aufzeichnung von Strukturen leerer Verzeichnisse * Problematisch große Binärdateien zu speichern (durch die Mehrzahl von individuellen Arbeitskopien) * Schwierig zu lernen - manchmal verbringt der Nutzer Stunden, um auf eine Lösung zu kommen * Git trackt keine Namensänderungen * Nicht user-friendly * Rebase können destruktiv sein und das ganze Team beeinflussen | * Abhängigkeit von der Netzwerkverbindung * Keine Möglichkeit gleichzeitig an derselben Datei zu arbeiten * Tree Conflict - problematisch Branches zusammenzu- führen, falls Verzeichnisse umbenannt wurden | * Die Historie kann man nicht ändern (nur den letzten commit) * Addons müssen in Python geschrieben werden * Schwieriger Features und Funktionen zu kombinieren, wenn man unterschiedliche Erweiterungen benutzt * Rollback beeinflusst nur den letzten commit |

Git ist für uns die beste Lösung. Wir wollen nicht auf die Netzwerkverbindung angewiesen sein. Außerdem sind wir bei einem Ausfall abgesichert, weil jede lokale Arbeitskopie ein Back-Up ist. Es wird keine Lese- und Schreibrechte geben und jeder kann unabhängig an denselben Dateien arbeiten. Git ist schnell und ist in IDE als built-in Tool oft vorhanden.

# 

# UML-Tool mit Quellcodegenerierung

|  | **Papyrus** | **StarUML** |
| --- | --- | --- |
| Kosten | Kostenlos | Version 1:  Kostenlos, wird aber nicht mehr Gewartet  Version 2:  99$ per User |
| Plattformen | Windows, Linux, Mac | Windows, Linux, Mac |
| Sprache | Java | Java, C++, C# |
| Vorteile | * Open Source * Eclipse Plug-In * Bietet viele verschiedene UML Diagramme zum Implementieren an * Support für SysML * customizable | * Online Support * Support für SysML * customizable |
| Nachteile | * Bedienung der Benutzeroberfläche kann sehr stressig sein (viele Menüs, kann schnell für Verwirrung sorgen) | * recht Komplex für Anfänger * Nicht mehr OpenSource (Version 2) |
| Reviews | Keine | Sourceforge:  37 Reviews (4.9/5) |
| Erfahrung | Alle Teammitglieder | Kein Teammitglied |

Wir haben uns dafür entschieden, das UML Tool Papyrus von Eclipse zu verwenden, da unser Team bereits viel damit gearbeitet hat und schon so manche Erfahrung damit sammeln konnten. Zudem nutzen wir Eclipse als IDE und können somit unsere angelegten Klassen sofort in unser Projekt als Programmcode integrieren.

# 

# Build-Tool

|  | **Gradle** | **Apache Maven** | **Apache Ant** |
| --- | --- | --- | --- |
| Kategorie | Projektmanagement-  Werkzeug sowie Build-Werkzeug | Projektmanagement-Werkzeug sowie Build-Werkzeug | Build-Werkzeug |
| Kosten | Apache 2.0 Lizenz, kostenlos | Apache 2.0 Lizenz, kostenlos | Apache 2.0 Lizenz, kostenlos |
| Betriebssystem | Plattformunabhängig | Plattformunabhängig | Plattformunabhängig |
| Sprachen | Java, Groovy, Scala,  C++, Swift, Kotlin | Java, C#, Ruby, Scala | Java |
| Konzept | * Convention over Configuration * Dependency Management * Phasen * Plug-Ins | * Deklarativ * Don’t repeat yourself * POM * Artifact * Lifecycle * Phasen * Goal | * Imperativ * Targets * Tasks |
| Build-Dateien | DSL  build.gradle | XML  pom.xml | XML  build.xml |
| Erfahrung | Alle Teammitglieder | Alle Teammitglieder | Kein Teammitglied |

Es wird beim Projekt Maven verwendet. In einem Projekt muss häufig auf neue Funktionen migriert werden, und mit Maven wird die Migration vereinfacht.

# 

# Prototyping

|  | **Figma** | **Adobe XD** | **Sketch** |
| --- | --- | --- | --- |
| Kosten (günstigster Plan) | kostenloser Plan verfügbar | 11,98€/Monat | 9€/Monat pro Editor |
| Erfahrung | alle Teammitglieder | keiner | keiner |
| Kollaboration | ja | ja, aber mit Sync aus Offline-App -> evtl. Versionsunterschiede | ja |
| Entwickler Handoff | ja | ja | ja |
| Plugins | ja | ja | ja |
| Betriebssystem | webbasiert (Internet erforderlich), Desktop-Apps verfügbar | offline auf macOS + Windows verfügbar | Web oder macOS |
| Dateiformate | PDF, PNG, JPG, SVG | PDF, PNG, JPEG, SVG | PDF, PNG, JPG, TIFF, WebP, SVG, EPS |
| HTML Export | mit Plugin | mit Plugin | mit Plugin |

Figma ist bei weitem am günstigsten und bietet alle benötigten Features. Des weiteren haben alle Erfahrung mit Figma. Auch Kollaboration wird durch Figma vereinfacht.

# 

# IDE

|  | **Eclipse** | **IntelliJ IDEA** | **NetBeans** |
| --- | --- | --- | --- |
| Kosten | Kostenlos | Kostenlos für Studenten   * Ultimate - unternehmens- orientiert und nicht kostenlos (SQL Support, Datenbanken Tools, etc.) * Community Edition - Kostenlos und open-source | Kostenlos |
| Erfahrung | ja | ja | nein |
| Java optimiert? | ja | ja | ja |
| Vorteile | * Portierbare Workspaces * Docker runtimes * Git Support * Viele Plugins * Modellierung * Für große Projekte * IntelliSense * Viele Programmier- sprachen * Shortcuts * Built-in Debugger * Große Community * Eclipse ist ein größerer Projekt mit mehr Möglichkeiten und ready-to-use Tools für Java Entwicklung | * Smart Refactorings * Android und JavaEE Support * “Approved” Plugins * Gutes UI * Debugger * Built-in Git * Studenten Angebote * Gradle Support * Fokus auf Java und JVM (Scala und Kotlin) | * Plattformüber- greifend * Community * Viele Features * Kleinerer Speicherver- brauch * Kein Re-Indexing * Gründlicher Plugin Check * SQL Support * Open-Source * User friendly |
| Nachteile | * Slow Performance | * Teuer * Nicht möglich mehrere Projekte in einem Fenster zu öffnen * Standard Hotkeys verhalten unterschiedlich * Nicht genug Plugins * Nutzt viel RAM * Öfteres Indexing | * Kein fort- geschrittenes Refactoring möglich * Wenige Plugins |
| Betriebssystem | Windows, Linux, Max | Windows, Linux, Max | Windows, Linux, Max |
| Geeignet für | Insb. Open-source Entwicklung und große, komplexe Projekte | Professionals und Studenten | Professionals und Studenten |

Begründung: Eclipse ist für uns eine bessere Lösung, weil alle Teammitglieder damit Erfahrung gemacht haben. Es ist kostenlos und ist für die Java Entwicklung sehr gut geeignet.

# 

# Test-Automatisierung

|  | **JUnit** | **TestNG** | **SikuliX** |
| --- | --- | --- | --- |
| Verwendung | Komponententests | Komponententests | GUI-Tests |
| Kosten | Common Public License | Common Public License | MIT - License |
| Betriebssystem | Plattform- übergreifend | Plattform- übergreifend | Plattform- unabhängig |
| Sprache | Java | Java | Jython |
| Funktionen | * Schreiben und Ausführen von Tests * Testmethoden * Testergebnisse * Test-Runner | * Anmerkungen * Flexible Laufzeit- konfiguration * Flexible Plug-In-API * Stellt “Testgruppen” * Unterstützt von Multithread - Tests | * automatisiert alles, was Sie auf dem Bildschirm Ihres Desktop Computers sehen (anhand von Screenshots) |
| Erfahrung | Alle Teammitglieder | Kein Teammitglied | Kein Teammitglied |

Es wurde beschlossen, JUnit zu verwenden, weil alle Teammitglieder damit Erfahrung haben und das Framework kann schnell als Plug-In mit Eclipse integriert werden.

# 

# Dokumentationstool

|  | **JavaDoc** | **Doxygen** |
| --- | --- | --- |
| Kosten | Kostenlos | Kostenlos |
| Betriebssystem | Windows, Linux, MacOS | Windows, Linux, MacOSX |
| Sprache | Java | C/C++, Java, Lex, Python, C#, PHP, etc… |
| Vorteile | * Integriertes Tool in JDK * HTML 5 support * Erzeugung von HTML, RTF, XML, PDF und anderen Formaten * Einfache Handhabung | * Erstellung von Konfigurationsdateien * Gruppierung von Entitäten in Modulen und die Erstellung einer Hierarchie von Modulen * Support für verschiedene Hilfeformate von Drittanbietern |
| Nachteile | * gibt nur wenig informationen zu den class Libraries her * geht wenig auf funktionsweise von methoden und anderen wichtigen neben funktionen wie debugging ein | * PDF erzeugung Problematisch * Keine Rekursive Einbeziehung von Ordnern |
| Erfahrung | Kein Teammitglied | Kein Teammitglied |

Wir werden das Tool JavaDoc benutzen, da es bereits in Eclipse (im JDK) mit integriert und leicht zu bedienen ist.

# 

# Obfuscator

|  | **ProGuard** | **JavaGuard** | **yGuard** |
| --- | --- | --- | --- |
| Kosten | Kostenlos | Kostenlos | Kostenlos |
| Platforms | Windows, Android, Linux | Windows, Mac, Linux | Windows, Mac, Linux |
| Languages | Java, Kotlin | Java | Java |
| Vorteile | * Opensource * Reduziert Code bis zu 90% durch entfernung von “Totem Code” * Optimiert Java Applikationen für Smartphones, Bluray Player und andere Geräte * Beschleunigt Programm bis zu 20% | * Opensource * Schutz vor Reverse Engineering * Reduziert Code | * Opensource * Reduziert Code * Einfache Handhabung * Konfigurierbarer und sicherer Code * Leistungsfähige und einfach zu bedienende XML-Syntax * Kompatibel mit allen vorhandenen Java Versionen |
| Nachteile | * bietet nur bedingt Schutz vor Reverse Engineering * Hauptfunktion: verkleinerung von Java Apps | * kaum Online Support * letztes Update laut Webseite 2016, letzte Meldung vom Developer war 2002 * Wahrscheinlich kein Support für spätere Java Versionen | * etwas schlechtere performance als ProGuard * Funktionalität variiert etwas mit jeder Java Version |
| Reviews | Sourceforge:  32 Reviews (4.9/5) | Sourceforge:  9 Reviews (4.1/5) | Keine |
| Erfahrung | Kein Teammitglied | Kein Teammitglied | Kein Teammitglied |

Da die Benutzung und Effizienz eines solchen Programms für uns noch sehr neu ist, haben wir uns entschieden ProGuard zu verwenden, da dieses von den drei oben aufgelisteten Tools am besten in der Performance und den Online Rezensionen abschneidet.

# Codeconventions

|  | **Oracle** | **Google** |
| --- | --- | --- |
| Lesbarkeit | hoch | mittel-hoch |
| Benennungen | **Pakete** - alles kleingeschrieben  **Klassen** - großer Anfangsbuchstabe für jeden Wortteil  **Interfaces** - wie Klassen  **Methoden** - großer Anfangsbuchstabe ab zweiten Wortteil  **Variablen** - wie Methoden  **Konstanten** - alles großgeschrieben | **Pakete** - alles kleingeschrieben  **Klassen** - großer Anfangsbuchstabe für jeden Wortteil  **Interfaces** - wie Klassen  **Methoden** - großer Anfangsbuchstabe ab zweiten Wortteil  **Variablen** - wie Methoden  **Konstanten** - alles großgeschrieben |
| Kommentare | Code sollte verständlich gehalten werden, so wenig Kommentare wie nötig | Code sollte alle 3-7 Zeilen ein Kommentar enthalten |
| Einrücken | 4 Whitespaces zum Einrücken,  8 Whitespaces für Tab,  Zeile nicht länger als 80 Zeichen | 2 Whitespaces zum Einrücken,  Zeile nicht länger als 100 Zeichen |

Für eine verbesserte Lesbarkeit und einfachere Wartung wird als Java Coding Convention Oracle’s Guidelines und Datenorganisierungsvorschriften verwendet. Der Großteil der Dokumentation findet außerhalb des Codes statt.

# 

# Kollaborationstools

|  | **Discord** | **Trello** | **Miro** |
| --- | --- | --- | --- |
| Kosten | Kostenfrei oder 49,99€ pro Benutzer & Jahr | Kostenfrei oder 9,99€ pro Benutzer & Monat | Kostenfrei oder 8,00€ pro Benutzer & Monat |
| Nachrichten | ja | Nur Kommentare | ja |
| Videokonverenz | ja | nein | nein |
| File Sharing | limitiert | ja | ja |
| Terminplanung | ja | ja | nein |
| Aufgaben-  verwaltung | nein | ja | ja |

Zur Kollaboration werden die beiden Anwendungen “Discord” und “Trello” ausgewählt.

**Discord** - Mit Discord soll unseren Teammitgliedern eine einfache und unkomplizierte Kommunikation ermöglicht werden, welches sowohl arbeiten zu Zweit, aber auch in größeren Gruppen gewährleistet.

**Trello** - Auf Trello organisieren wir alle Dokumente, Meilensteine und Ziele. Weiterhin werden alle anstehenden Änderungen und Fehlerbehebungen auch dort gesammelt und abgearbeitet.

# Quellen

### Versionierung

* <https://entwickler.de/programmierung/git-oder-mercurial-was-sollte-man-nutzen/>
* <https://www.perforce.com/blog/vcs/mercurial-vs-git-how-are-they-different>
* <https://www.javatpoint.com/git-vs-mercurial>
* <https://entwickler.de/programmierung/git-vs-subversion-svn-welches-versionskontrollsystem-sollten-sie-nutzen>

### Build-Tool

* <https://de.education-wiki.com/5472240-what-is-maven>
* <https://entwickler.de/java/der-nachste-grosse-schritt>

### Prototyping

* <https://www.adobe.com/de/products/xd/pricing/individual.html>
* <https://www.imaginarycloud.com/blog/figma-vs-sketch/>
* <https://www.imaginarycloud.com/blog/figma-vs-adobe-xd-main-differences/>
* <https://www.sketch.com/pricing/>

### IDE

* <https://www.educba.com/intellij-vs-netbeans/>
* <https://www.pcwelt.de/ratgeber/NetBeans-IDE-Web-Entwicklung-363490.html>
* <https://careerkarma.com/blog/intellij-vs-eclipse/>
* <https://www.educba.com/eclipse-vs-intellij/>

### Test-Automatisierung

* [JUnit-Lernprogramm (tutorialspoint.com)](https://www.tutorialspoint.com/junit/index.htm)
* [TestNG-Tutorial (tutorialspoint.com)](https://www.tutorialspoint.com/testng/index.htm)
* [RaiMan's SikuliX](http://sikulix.com/)

### Dokumentationstool

* <https://www.doxygen.nl/manual/features.html>

### Obfuscator

* <https://sourceforge.net/projects/javaguard/>
* <https://sourceforge.net/projects/proguard/>

### Codeconventions

* <https://www.oracle.com/java/technologies/javase/codeconventions-contents.html>
* <https://google.github.io/styleguide/javaguide.html>

### Kollaborationstool

* <https://www.placetel.de/ratgeber/collaboration-tools-software>

# Gruppe 02

# 

# Mitglieder:

# Lisa Hauptvogel

# Mariia Würtz

# Lloyd König

# Noricia Matschuck

# Dmitrij Kolosov (neuestes Mitglied)