Softwareprojekt Wintersemester 2019/2020

am Fachgebiet Software Engineering, Leibniz Universität Hannover

Spezifikation
PDF-Zensor

`SWP-WS1920-PDF-Zensor-Spec-v01.pdf`

Vorgelegt

am 12.11.2019 von PDF-Zensor

Ausführende:

Nachname	*Vorname*	*Rolle*
Speckmann	Marc	Projektleiter
Hagen	Tim	Qualitätsbeauftragter
Bohlin	Lennart	
Brandt	Daniel	
Falkenberg	Johs	
Gluzmann	Maksim	
Grätz	Mike	
Möller	Lennart	
Ramadan	Ahmad	

Das Dokument enthält

☒ Die Anforderungen aus Kundensicht (User Requirements)

☒ Anforderungen, wie das zu System zu gestalten ist (System Requirements)

Datum, Unterschrift des Projektleiters, auch für die anderen Projektangehörigen

Kunden-Bewertung

Der Kunde, Fabian Pflug, bestätigt mit seiner Unterschrift, diese Anforderungsspezifikation erhalten, geprüft und für inhaltlich ☐ **in Ordnung** | ☐ **weitgehend in Ordnung** | ☐ **deutlich zu verbessernd** | ☐ **nicht akzeptabel** befunden zu haben.

Datum, Unterschrift des Kunden; evtl. Vermerk.

Inhaltsverzeichnis

1 Mission des Projekts

1.1 Erläuterung des zu lösenden Problems

Das Produkt soll ein einfaches CLI bieten, um PDF-Dateien für den privaten wie auch kommerziellen Gebrauch zu zensieren. Der Nutzer möchte den Zensor dafür mit möglichst wenig Konfigurationsaufwand betreiben können, um entweder das komplette Dokument oder nur ausgewählte Stellen [nicht] zu zensieren.

1.2 Wünsche und Prioritäten des Kunden

Es folgen die Wünsche des Kunden, nach absteigender Priorität geordnet:

- ein Kommandozeilentool ohne GUI
- alle verwendeten Lizenzen sollen in der kommerziellen Nutzung kostenlos sein
- die Bedienung und Installation soll einfach sein
- die Zensur soll irreversibel stattfinden
- die ursprüngliche Datei soll nicht versehentlich überschrieben werden
- Metadaten wie zum Beispiel Autor, Kommentare und Annotationen sollen immer entfernt werden
- Links, Regex-Matches und restlicher Text sollen mit Balken unterschiedlicher Farbe zensiert werden
- Bilder, dies beinhaltet inline-Vektorbilder, sollen erkannt und durch ein Standardbild ersetzt werden, welches die Ausmaße des Ursprungsbildes beibehält
- es sollen mindestens drei verschiedene Farben bei der Zensur verwendet werden
- die Zensur soll nur auf / außer auf Text stattfinden, der zuvor in einem anderen Programm markiert (gehighlighted) wurde
- eine für einen Menschen lesbare und in einem gewöhnlichen Texteditor änderbare Konfigurationsdatei soll verwendet werden
- die Installation soll in maximal drei Schritten stattfinden
- die Installation soll durch einem Paketmanager unterstützt werden
- Git soll zur Versionsverwaltung genutzt werden
- CI-Skripte sollen zum Testen und Ausrollen im Git genutzt werden
- grundsätzlich sollen die gleichen Anpassungen, die in der Konfigurationsdatei getroffen werden können, auch über Argumente in der Kommandozeile erreichbar sein (siehe Kapitel 5 für mehr Informationen hierzu)
- eine Manpage zu dem Kommando soll vorliegen

1.3 Domänenbeschreibung

PDF-Zensor wird von einer Einzelperson innerhalb einer Konsole ausgeführt. Dabei ist keine große UI vonnöten (und auch nicht durch den Kunden erwünscht). Das Produkt soll einzig – vergleichbar mit anderen Konsolen-Kommandos – mit einer Reihe Argumente aufgerufen werden, den Auftrag ausführen und sich danach beenden.

1.4 Maßnahmen zur Anforderungsanalyse

Es wurden folgende Überlegungen angestellt (in beliebiger Reihenfolge):

- CLI-Argumente: welche und wie sie aussehen könnten
- Konfigurationsdatei: wie sie aussehen könnte und welches Dateiformat am sinnvollsten wäre
- UML-Klassendiagramm: die wichtigsten Klassen und wie sie zusammenhängen könnten
- UML-Sequenzdiagramm: wie die Suche nach Regex-Matches und die zeitgleiche farbige Zensur aussehen könnte

Es wurden folgende Prototypen erstellt (in beliebiger Reihenfolge):

- Berechnen der Bounding-Boxen der einzelnen Glyphen und das farbige Einzeichnen dieser in die PDF
- Berechnen der Bounding-Boxen von Zeilen mit den Bounding-Boxen der Glyphen als Grundlage
- Parsen des ”Token-Streams“ der PDF und Ersetzen der ”Do“-Operationen durch Zeichnungen
- Tokenizen des PDF-Inhaltes und eine entsprechende farbige Markierung
- CLI-Parser

Manche der Überlegungen und Prototypen wurden dem Kunden präsentiert, um seine Meinung zu hören. Sie alle dienten gleichermaßen aber auch dem Testen der Schnittstellen und wie sie aussehen, sodass wir dies beim Entwurf berücksichtigen können. Insgesamt kam dabei heraus, dass wir das Format der Konfigurationsdatei frei wählen dürfen. Wir brauchen allerdings eines, das die Präzedenz der definierten Regex-Muster speichert. Das Berechnen der Bounding-Boxen von Glyphen hat ergeben, dass die Zensur in mindestens zwei Durchläufen je Seite passieren muss: 1. die Seite wird analysiert und die Textboxen werden zwischengespeichert 2. der Text wird entfernt und die Textboxen an seiner statt gezeichnet.

Anhand einer PDF des Kunden haben wir herausgefunden, dass nicht in jeder PDF-Datei die Transformationsmatrix der Grafiken am Ende wieder zurückgesetzt wird. Daher ist es notwendig, dass wir eine Funktionalität dafür selbst einbauen, damit die Zensurbalken tatsächlich an Stelle des Textes und nicht verschoben platziert werden.

2 Rahmenbedingungen und Umfeld

2.1 Einschränkungen und Vorgaben

Zu entwickeln ist ein Kommandozeilentool, welches über die Eingabe von Argumenten oder über eine Konfigurationsdatei (Standardkonfigurationsdatei in `$HOME/.config/pdf-zensor`) situationsspezifisch angepasst werden kann, um nach Präferenz des Nutzers eine PDF-Datei zu zensieren.

Die ursprüngliche Datei sollte dabei nicht automatisch durch das Programm überschrieben werden. Insbesondere ist aber auch die Verwendung des optionalen Arguments `-o` beziehungsweise `--out` zur genauen Angabe einer Ausgabedatei anerkannt, was ein Überschreiben ermöglichen würde (siehe Use Case Nr. 06).

Im Bezug auf die Zensur ist zu beachten, dass die Daten irreversibel zensiert werden, die grundlegenden Strukturinformationen jedoch erhalten bleiben. Daher soll es mindestens drei verschiedene Zensurfarben geben, die jeweils andeuten, welchen Informationstypen sie zensieren. Bilder, dies beinhaltet auch inline-Vektorbilder, sind durch ein Standardbild zu ersetzen, welches mitgeliefert wird, aber gegebenenfalls in der Konfigurationsdatei änderbar sein sollte. Dabei soll die Dimension (Ausmaße) des zensierten Bildes erhalten bleiben. Zusätzlich zu dem Modus, der den gesamten Text zensiert und anhand verschiedener Regex-Matches die Zensurfarben festmacht, ist außerdem eine Variante zu entwickeln, die nur oder aber außer auf markiertem Text arbeitet (Text, der zuvor in einer anderen Software gehighlighted wurde). Metadaten wie Autor, Kommentare und auch Anmerkungen sind grundsätzlich zu entfernen.

Von besonderer Bedeutung ist eine einfache Installation und Bedienung sowie auch die Wartbarkeit der Software. So soll die Installation in maximal drei Schritten durchzuführen sein und mittels eines Paketmanagers erfolgen. Außerdem soll eine Manpage zur Verfügung stehen, welche ausführlich über die Nutzung von PDF-Zensor informiert. Für die Versionsverwaltung ist ein durch CI-Skripte unterstütztes Git zu nutzen. Des Weiteren sollen alle verwendeten Bibliotheken in der kommerziellen Nutzung kostenfrei sein.

2.2 Anwender

Priorisiert ist die Verwendung der Software als Kommandozeilentool durch den Kunden. Dieser hat jedoch auch die Möglichkeit einer Website genannt, welche die gleiche Funktionalität wie das Kommandozeilentool, jedoch für die Allgemeinheit zugänglich, bereitstellt.

2.3 Schnittstellen und angrenzende Systeme

Die Softwareentwicklung erfolgt in Java Version 11. Es werden folgende, kommerziell kostenlos nutzbare Bibliotheken verwendet:

- PDFBox – Schnittstelle zwischen den PDF-Dateien und Java
- Log4j – Protokollierung der Abläufe der Software
- picocli – Auswerten der Kommandozeilenargumente
- Jackson – Parsen der `JSON`-Konfigurationsdatei
- Apache Commons Lang – Diverse Helfermethoden zur allgemeinen Nutzung
- JetBrains Java Annotations – Java Annotationen für verbesserte Code Dokumentation

3 Funktionale Anforderungen

3.1 Use Case-Diagramm

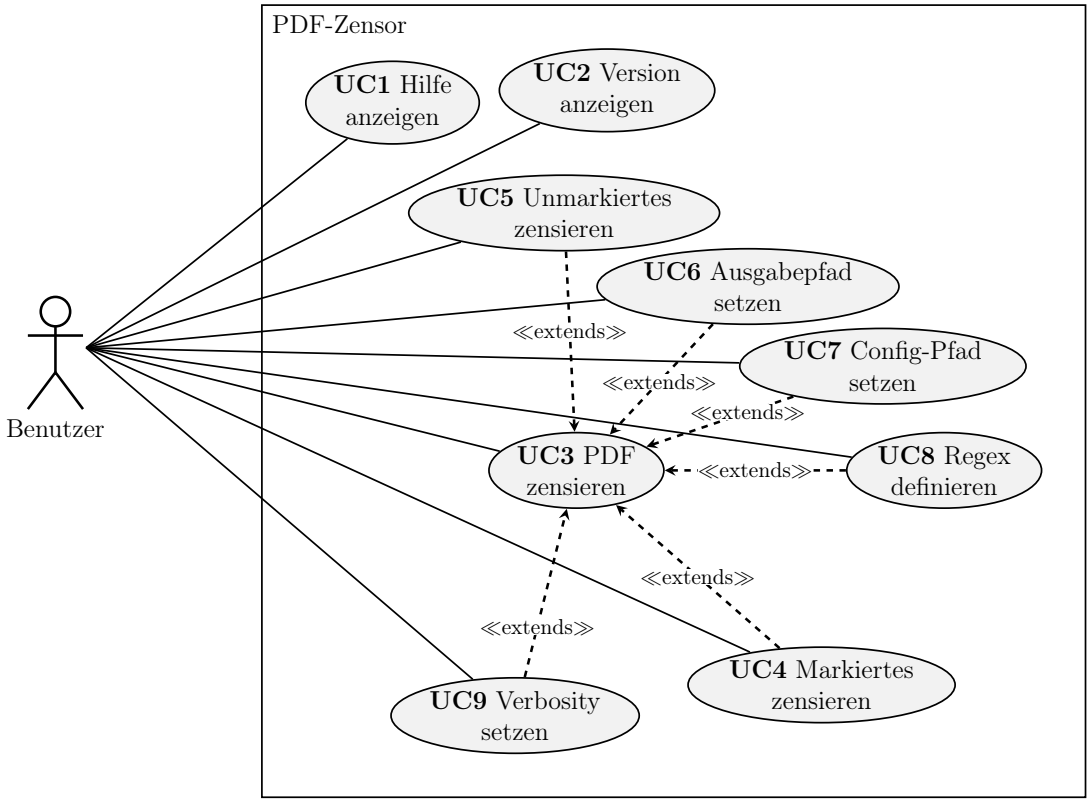

Abbildung 1: Use-Case-Diagramm

3.2 Use Case-Beschreibungen

3.2.1 UC: Hilfe anzeigen

<table>
<tr><td>Use Case Nr. 01</td><td colspan="2">Hilfe anzeigen</td></tr>
<tr><td>Erläuterungen</td><td colspan="2">Zeigt die Hilfe an, welche Informationen über die Bedienung des PDF-Zensors enthält: mögliche Argumente und deren Bedeutung.</td></tr>
<tr><td>Systemgrenzen (Scope)</td><td colspan="2">Arg.-Parser</td></tr>
<tr><td>Ebene</td><td colspan="2">Nebenfunktion</td></tr>
<tr><td>Vorbedingung</td><td colspan="2">Java Version 11 und PDF-Zensor müssen installiert sein.</td></tr>
<tr><td>Mindesgarantie</td><td colspan="2">Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.</td></tr>
<tr><td>Erfolgsgarantie</td><td colspan="2">Die Hilfe wird angezeigt. Sie enthält alle nötigen Informationen zur Bedienung des PDF-Zensors.</td></tr>
<tr><td rowspan="3">Stakeholder und Interessen</td><td>Stakeholder</td><td>Interessen</td></tr>
<tr><td>Benutzer</td><td>Möchte sich die Hilfe anzeigen lassen, um mehr über die Möglichkeiten der Benutzung des PDF-Zensors zu erfahren oder die Syntax nachzuschlagen.</td></tr>
<tr><td>Entwickler</td><td>Will die korrekte Verwendung der Software ermöglichen, ohne dabei für zusätzlichen Support bereitstehen zu müssen oder Instruktionsaufwand zu haben.</td></tr>
<tr><td>Hauptakteur</td><td colspan="2">Benutzer</td></tr>
<tr><td>Auslöser</td><td colspan="2">Eingabe von "<code>pdf-zensor -h [...]</code>“ oder "<code>pdf-zensor --help [...]</code>“ in der Kommandozeile.</td></tr>
<tr><td>Hauptszenario</td><td colspan="2">1. Der Benutzer ruft die Hilfe auf (siehe Auslöser).
2. Hilfe wird angezeigt und das Programm beendet.</td></tr>
<tr><td>Erweiterung</td><td colspan="2">Es gibt keine Erweiterungen</td></tr>
<tr><td>Priorität</td><td colspan="2">Mittel</td></tr>
<tr><td>Verwendungs-häufigkeit</td><td colspan="2">Selten</td></tr>
</table>

Erläuterungen und Details

- Wird Hilfe zu der Nutzung des Tools angefordert, so werden alle weiteren angegebenen Argumente ignoriert und die Angabe einer Eingabedatei ist nicht erforderlich.

3.2.2 UC: Versionsnummer anzeigen

Use Case Nr. 02	**Versionsnummer anzeigen**	
Erläuterungen	Zeigt die Version des installierten PDF-Zensors an.	
Systemgrenzen (Scope)	Arg.-Parser	
Ebene	Nebenfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die Version des aufgerufenen PDF-Zensors wird in den stdout ausgegeben.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die aktuelle Versionsnummer des auf dem System installierten PDF-Zensors wissen, um bei Problemen gezielter nach Hilfe fragen zu können.
	Kunde	Möchte die Software weiterentwickeln können und seine Arbeit dabei vom Ursprungsprodukt trennen.
	Entwickler	Möchte versionsspezifisch Hilfe leisten können.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor -V [...]`" oder "`pdf-zensor --version [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft die Funktion "Versionsnummer anzeigen" auf (siehe Auslöser). 2. Die Versionsnummer der laufenden Version des PDF-Zensors wird angezeigt und das Programm wird beendet.	
Erweiterung	Es gibt keine Erweiterungen	
Priorität	Niedrig	
Verwendungshäufigkeit	Sehr selten	

Erläuterungen und Details

- Wird die Ausgabe der Versionsnummer des Tools angefordert, so werden alle weiteren angegebenen Argumente ignoriert und die Angabe einer Eingabedatei ist nicht erforderlich.

3.2.3 UC: Komplette PDF-Datei zensieren

Use Case Nr. 03	**Komplette PDF-Datei zensieren**	
Erläuterungen	Zensiert die komplette PDF-Datei, die beim Aufruf des Tools angegeben wurde.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die beim Aufruf angegebene PDF-Datei wird komplett und irreversibel zensiert, sodass nur noch ihre Struktur erkennbar ist, ohne, dass in irgendeiner Form Rückschlüsse auf den Autor oder den Inhalt der ursprünglichen Datei möglich sind.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die von ihm angegebene PDF-Datei zensieren.
	Autor der PDF	Möchte, dass seine Anonymität in Bezug auf die zensierte PDF-Datei zu 100 % gewährleistet ist (auch dann, wenn der Autor nichts von der Anwendung des PDF-Zensors auf eine von ihm erstellte PDF-Datei weiß).
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die zensierte Version der Eingabedatei wird in demselben Verzeichnis wie die Eingabedatei, mit dem Namen "`in_cens.pdf`", abgelegt. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3.	
Priorität	Sehr hoch	
Verwendungshäufigkeit	Sehr häufig	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.

3.2.4 UC: Markierte Stellen in einer PDF-Datei zensieren

Use Case Nr. 04	**Markierte Stellen in einer PDF-Datei zensieren**	
Erläuterungen	Zensiert die angegebene PDF-Datei, sodass ausschließlich die zuvor in einem anderen Programm markierten Stellen (Textpassagen, Bilder, ...) bei der Zensur berücksichtigt werden.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die beim Aufruf angegebene PDF-Datei wird irreversibel zensiert, sodass die vorher markierten Stellen nicht mehr zu erkennen sind.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die angegebene PDF-Datei zensieren, sodass die von ihm markierten Stellen nicht mehr erkennbar sind.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" -m [...]`" oder "`pdf-zensor "in.pdf" --censor-marked [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die Eingabedatei wird entsprechend der angegebenen Argumente zensiert, insbesondere werden dabei nur markierte Stellen zensiert. Danach wird die PDF-Datei in demselben Verzeichnis wie die Eingabedatei, mit dem Namen "`in_cens.pdf`", abgelegt. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3.	
Priorität	Sehr hoch	
Verwendungshäufigkeit	Mittel bis häufig	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.

3.2.5 UC: Nicht markierte Stellen in einer PDF-Datei zensieren

Use Case Nr. 05	**Nicht markierte Stellen in einer PDF-Datei zensieren**	
Erläuterungen	Zensiert die beim Aufruf angegebene PDF-Datei, sodass keine der vom Nutzer zuvor in einem anderen Programm markierten Stellen (Textpassagen, Bilder, ...) zensiert werden.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die beim Aufruf angegebene PDF-Datei wird irreversibel zensiert, sodass die vorher markierten Stellen noch zu erkennen sind, alles andere hingegen nicht mehr.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die von ihm angegebene PDF-Datei zensieren, sodass die vom Nutzer markierten Stellen nicht zensiert werden.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" -u [...]`" oder "`pdf-zensor "in.pdf" --censor-unmarked [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die Eingabedatei wird entsprechend der angegebenen Argumente zensiert, insbesondere werden bei der Zensur alle markierten Stellen ignoriert und nur deren Markierung entfernt. Danach wird die PDF-Datei in demselben Verzeichnis wie die Eingabedatei, mit dem Namen "`in_cens.pdf`", abgelegt. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3.	
Priorität	Sehr hoch	
Verwendungshäufigkeit	Mittel bis häufig	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.

3.2.6 UC: Ausgabedatei setzen

Use Case Nr. 06	**Ausgabedatei setzen**	
Erläuterungen	Zensiert die beim Aufruf angegebene PDF-Datei, wobei der Name der Ausgabedatei explizit beim Aufruf angegeben wird.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die zensierte Version der Eingabedatei wird unter dem angegebenen Namen erstellt.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die von ihm angegebene PDF-Datei zensieren und sie hinterher unter dem angegebenen Namen (und eventuell im angegebenen Verzeichnis) vorfinden.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" -o "out" [...]`" oder "`pdf-zensor "in.pdf" --output "out" [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die zensierte Version der Eingabedatei wird ausgehend vom Arbeitsverzeichnis als "`out`" abgelegt. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn in demselben Verzeichnis mit demselben Namen bereits eine Datei existiert, Dann wird diese überschrieben. Weiter bei Schritt 3. 2b Wenn `"out"` ein existierendes Verzeichnis und keine Datei ist, Dann wird die zensierte Datei unter dem Namen "`in_cens.pdf`" im Verzeichnis `"out"` abgelegt. Weiter bei Schritt 3.	
Priorität	Hoch	
Verwendungshäufigkeit	Häufig	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.
- Wird mit "`-o "out"`" ein Name der Ausgabedatei angegeben, so wird diese ausgehend von dem aktuellen Arbeitsverzeichnis gespeichert. Ist "`-o "out"`" nicht gesetzt, so wird die Ausgabedatei im Verzeichnis der Eingabedatei als "`in_cens.pdf`" gespeichert.

3.2.7 UC: Temporäre Konfigurationsdatei setzen

Use Case Nr. 07	**Temporäre Konfigurationsdatei setzen**	
Erläuterungen	Zensiert die beim Aufruf angegebene PDF-Datei, wobei die angegebene Konfigurationsdatei als Grundlage für die Zensur sowie weitere Argumente verwendet wird.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein. Eine valide Konfigurationsdatei muss vorliegen.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die beim Aufruf angegebene PDF-Datei wird irreversibel zensiert, dabei wird die angegebene Konfigurationsdatei als Grundlage für die Zensur sowie jegliche weitere Argumente verwendet.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die von ihm angegebene PDF-Datei schnell und einfach zensieren, indem die Verwendung einer Konfigurationsdatei ihm die Eingabe von Argumenten erspart.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" -c "config.json" [...]`" oder "`pdf-zensor "in.pdf" --config "config.json" [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die Eingabedatei wird unter Berücksichtigung der angegebenen Konfigurationsdatei als Grundlage, erweitert mit eventuell angegebenen Argumenten, zensiert und in demselben Verzeichnis wie die Eingabedatei, mit dem Namen "`in_cens.pdf`", abgelegt.. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3.	
Priorität	Hoch	
Verwendungshäufigkeit	Selten	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.

3.2.8 UC: Zusätzliche Regex hinzufügen

Use Case Nr. 08	**Zusätzliche Regex hinzufügen**	
Erläuterungen	Zensiert die beim Aufruf angegebene PDF-Datei, wobei bei der Zensur zusätzlich angegebene Regex mit einer zu ihnen optional angebbaren Farbe zensiert werden.	
Systemgrenzen (Scope)	Arg.-Parser, Eingabe- und Ausgabe-PDF-Datei	
Ebene	Hauptfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die Eingabedatei wird unter Verwendung der angegebenen Regex mit den entsprechenden Farben zensiert und die zensierte Version wird gespeichert.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte die von ihm angegebene PDF-Datei zensieren und dabei spezifischer auswählen können, was überhaupt und mit welcher Farbe zensiert wird.
Hauptakteur	Benutzer	
Auslöser	Eingabe von "`pdf-zensor "in.pdf" -e "regex" ["hex_color"] [...]`" oder "`pdf-zensor "in.pdf" --expression "regex" ["hex_color"] [...]`" in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die Eingabedatei wird unter Berücksichtigung der angegebenen Regex und ihren zugehörigen Farben zensiert und in demselben Verzeichnis wie die Eingabedatei, mit dem Namen "`in_cens.pdf`", abgelegt. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3. 2b Wenn vom Nutzer keine Farbe zu dem von ihm angegebenen Regex gesetzt wurde, Dann wird automatisch eine Standardfarbe ausgewählt. 2c Wenn vom Nutzer ein ungültiger Farbcode eingegeben wird, Dann wird ein Fehler zu einem nicht identifizierbaren Argument ausgegeben.	
Priorität	Mittel	
Verwendungs-häufigkeit	Mittel bis selten	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.
- Zulässige Hexadezimal-Farbcodes fangen mit "`#`", "`0x`" oder "`0X`" an, gefolgt von genau sechs Zeichen des Hexadezimalsystems.

3.2.9 UC: Protokollausführlichkeit setzen

Use Case Nr. 09	**Protokollausführlichkeit setzen**	
Erläuterungen	Zeigt beim Zensurvorgang alle Statusausgaben an, die innerhalb des spezifizierten Ausführlichkeitsrahmens liegen.	
Systemgrenzen (Scope)	Arg.-Parser sowie Logging	
Ebene	Nebenfunktion	
Vorbedingung	Java Version 11 und PDF-Zensor müssen installiert sein.	
Mindesgarantie	Auftretende Fehler werden ausgegeben und es werden nicht versehentlich Daten entfernt oder die Installation beschädigt.	
Erfolgsgarantie	Die PDF-Datei wird wie angegeben zensiert und abgespeichert, währenddessen erfolgen entsprechend der Ausführlichkeitseinstellung Ausgaben in den stdout.	
Stakeholder und Interessen	**Stakeholder**	**Interessen**
	Benutzer	Möchte Informationen zu den Programmabläufen erhalten, um gegebenenfalls Probleme zu erkennen und behandeln zu können.
	Entwickler / Support	Will bei der Entwicklung des Programmes und bei der Hilfestellung bei auftretenden Problemen ausführlich und strukturiert Informationen erhalten können, um über die Arbeitsweise des Programmes informiert zu sein.
Hauptakteur	Entwickler und Support	
Auslöser	Zum Beispiel die Eingabe von "`pdf-zensor "in.pdf" -v [...]`" oder "`pdf-zensor "in.pdf" --verbose [...]`" (siehe *Erläuterungen und Details*) in der Kommandozeile.	
Hauptszenario	1. Benutzer ruft den PDF-Zensor, wie unter Auslöser beschrieben, auf. 2. Die Eingabedatei wird den Angaben entsprechend zensiert und gespeichert. Die stattfindenden Abläufe werden dabei protokolliert. 3. Das Programm beendet sich.	
Erweiterung	1a Wenn eines der übergebenen Argumente (wie zum Beispiel die Eingabedatei) ungültig ist, Dann wird eine entsprechende Fehlermeldung ausgegeben. Weiter bei Schritt 3. 2a Wenn das Argument "`-o "out"`" verwendet wird, Dann wird die Ausgabedatei wie angegeben gespeichert (siehe UC Nr. 06). Weiter bei Schritt 3.	
Priorität	Niedrig	
Verwendungshäufigkeit	Selten	

Erläuterungen und Details

- Eine "ungültige Eingabe" beinhaltet auch vorhandene, jedoch nicht lesbare oder anderweitig nicht zensierbare Dateien (zum Beispiel keine PDF-Datei).
- Das Angeben einer Eingabedatei ist notwendig, weitere Argumente zur Spezifizierung der Zensur sind optional und in den Use Cases Nr. 04 bis Nr. 08 aufgeführt. Sollten weiterhin Fragen bestehen, so kann außerdem Kapitel 5 zu Hilfe gezogen werden.
- Das Ausführlichkeitslevel ist umso höher, je öfter die im Auslöser genannten Argumente wiederholt oder je mehr `v`'s eingegeben werden (zum Beispiel `-vvvv`).

4 Qualitätsanforderungen

4.1 Qualitätsziele des Projekts

4.1.1 Zuverlässigkeit

- Die garantierte Zensur des Dokumentes
- Die Software ersetzt Text und Regex-Matches durch farblich verschiedene Balken.
- Einstellbare Regex werden in der Konfigurationsdatei hinterlegt.
- Bilder werden durch ein Standardbild ersetzt.
- Abhängig von der Einstellung werden entweder die markierten Stellen im Text oder die nicht markierten Stellen oder beide zensiert.
- Unter anderem werden auch die bereitgestellten Testdokumente zum Prüfen der Zuverlässigkeit genutzt.

4.1.2 Integrität

- Die Metadaten der bearbeiteten Datei sind entfernt und lassen sich nicht rekonstruieren, sodass bei der bearbeiteten PDF nur die Struktur erkennbar ist.
- Der ersetzte Text lässt sich nicht rekonstruieren, sodass keine Rückverfolgung des Inhaltes möglich ist.
- Die entfernten Daten werden nicht zwischengespeichert und sind somit für Dritte nicht einsehbar oder rekonstruierbar.

4.1.3 Verwendbarkeit

- Das Programm lässt sich über die Konsole konfigurieren und ist mit einer Standardkonfiguration versehen, sodass es direkt ausgeführt werden kann.
- Eine übersichtliche Hilfeausgabe steht zur Verfügung, welche Erklärungen zu den möglichen Argumenten und Anpassungsmöglichkeiten bietet.

4.1.4 Wartbarkeit

- Der Code ist gut lesbar und ausführlich dokumentiert.
- Der Code ist in Java geschrieben und eine "Javadoc“ ist vorhanden. Damit das Programm weiter ausgebaut, gepflegt und verändert werden kann, wird auf Java als Sprache gesetzt, da diese weit verbreitet ist und viele Erweiterungen anbietet.
- Der Code ist modular und nicht verschachtelt aufgebaut, sodass er eine hohe Wartbarkeit aufweist.

4.1.5 Robustheit

- Fehlermeldungen statt Fehlverhalten
- Es werden Fehlermeldungen auf der Konsole ausgeben, wenn das Programm falsch bedient wurde und zusätzliche Hilfeoptionen eingeblendet, um dem Benutzer zu erklären, wie das Programm verwendet wird.

4.2 Qualitäts-Prioritäten des Kunden

Die Qualitätsziele sind wie folgt absteigend priorisiert:

- **Zuverlässigkeit:** Die Software arbeitet zuverlässig ohne Fehler.
- **Integrität:** Die bearbeiteten Daten sind irreversibel entfernt, sodass die zensierte Datei sicher weitergegeben werden kann.
- **Verwendbarkeit:** Die Software ist durch die Standardkonfigurationsdatei sofort verwendbar. Außerdem ist die Hilfe übersichtlich und leicht verständlich. Des Weiteren lassen sich Einstellungen verändern und in der Konfigurationsdatei als Standard festlegen.
- **Robustheit:** Die Software muss stabil laufen und entsprechende Fehlermeldungen ausgeben, wenn sie falsch bedient werden sollte (zum Beispiel nicht unterstütze Dateitypen).
- **Wartbarkeit:** Die Software lässt sich leicht anpassen, zum Beispiel um Fehler zu beheben oder die Funktionen zu erweitern.

4.3 Wie Qualitätsziele erreicht werden sollen

Unser Projektablauf beinhaltet vor dem finalen Fertigstellen einer Aufgabe, dass diese in einen Reviewstatus kommt, bis sie geprüft und abgesegnet wurde. Dazu muss mindestens ein weiteres Teammitglied die zu prüfende Aufgabe kontrollieren und das Programm muss die festgelegten Tests bestehen. Gibt es keinen Nachbesserungsbedarf, so wird die Aufgabe vom Reviewenden als erledigt markiert.

Des Weiteren werden BlackBox-Tests erstellt, mit denen die Qualitätsaspekte des Projekts geprüft werden. Die BlackBox-Tests erstellt beziehungsweise beauftragt der Qualitätsbeauftragte. Dieser stellt auch sicher, dass die Testfälle ausgeführt werden und leitet die anderen Teammitglieder in der Qualitätssicherung an.

Grundsätzlich ist aber jedes Teammitglied darum bemüht, die Qualitätsziele einzuhalten.

5 Hinweise zur Umsetzung

5.1 Erstentwurf

Beim Ausführen des Tools wird auf Basis der eingegebenen PDF ein neue Datei erstellt, in der – je nach Wunsch des Nutzers – der Inhalt teilweise oder auch gänzlich zensiert wurde. Die Ausgabedatei enthält weder Anmerkungen noch Metadaten, die auf den Autoren schließen ließen. Anhand verschiedenfarbiger Zensurbalken, durch die die Dateistruktur weiterhin erkennbar bleibt, sind Verlinkungen und bestimmte Textpassagen hervorgehoben. Jedwedes Bild wird bei der Zensur durch ein Standardbild ersetzt.

Abbildung 2 zeigt ein Sequenzdiagramm, das den groben Erstentwurf des Zensurprozesses widerspiegelt. Der Vorgang sieht vor, dass für jede Seite zuerst die Bilder durch das Standardbild ersetzt und ihre Bounding-Boxen für die weitere Zensur zwischengespeichert werden. Danach wird der Text sukzessive ausgelesen, wobei die Glyphen (mit Position in der PDF) nacheinander an den Tokenizer übergeben und, sofern sie zensiert werden sollen, aus der PDF gelöscht werden. Wenn ein Token gefunden wurde ruft der Tokenizer einen Callback auf dem PDFProcessor auf. Dabei werden die nötigen Informationen übergeben, um den korrektfarbigen Zensurbalken an der richtigen Position in die Ausgabedatei zu schreiben.

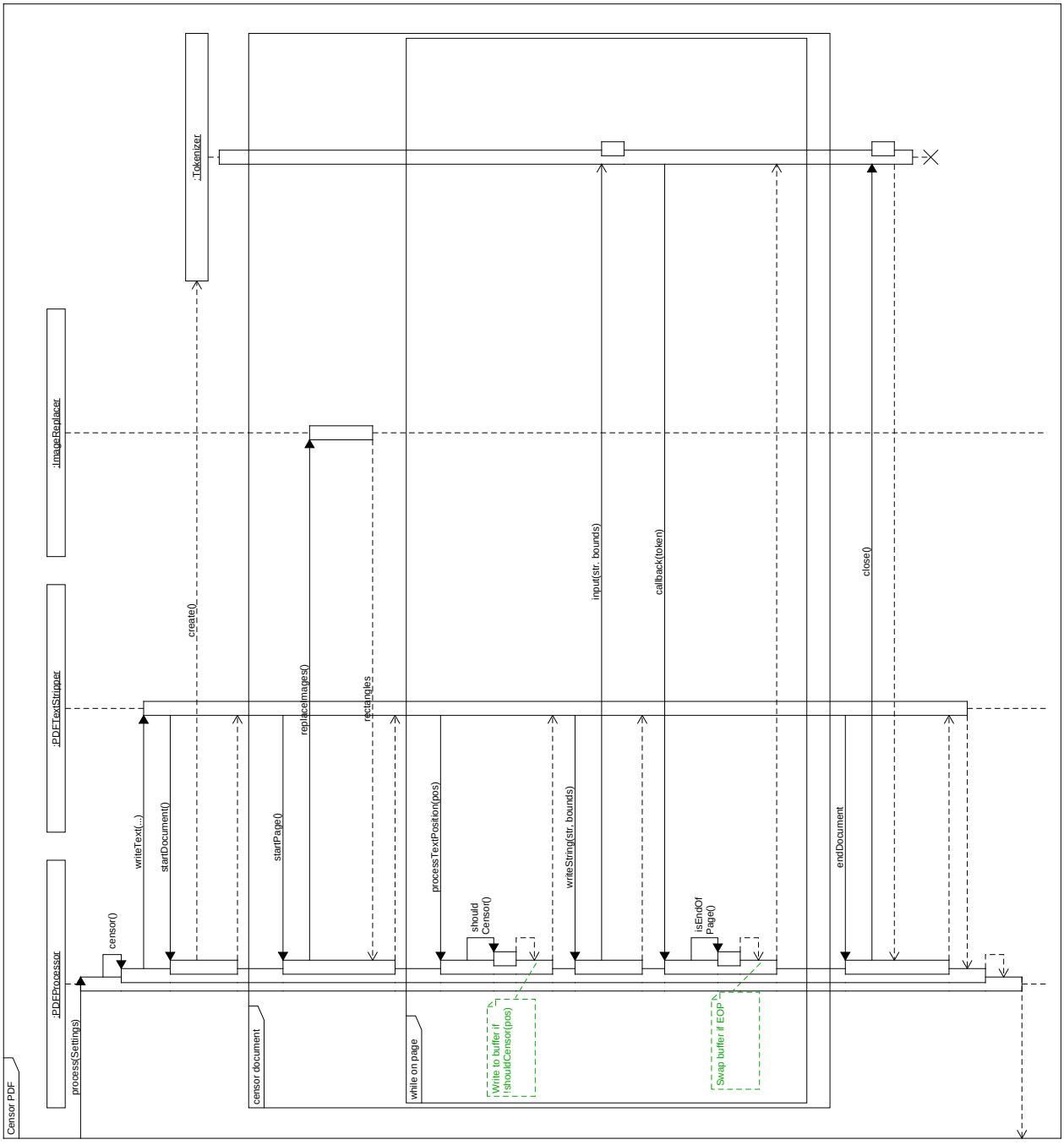

Abbildung 2: Sequenzdiagramm

5.2 Kommandozeilenargumente

Der Nutzer kann den PDF-Zensor mittels verschiedenster Argumente über die Kommandozeile aufrufen. Der Aufruf sollte dabei dem folgenden Muster folgen:

```
pdf-zensor [-m | -u] [-hVv] [-c "config.json"] [-o "out"]
[-e "regex" ["hex_color"]]... "in.pdf"
```

`"in.pdf"`	Setzt die Eingabedatei, die zensiert werden soll. Diese muss angegeben werden und eine gültige PDF-Datei sein.
`-c, --config "config.json"`	Setzt eine temporäre Konfiguration, die als Grundlage für andere eingegebene Argumente dient. Dies muss eine gültige `JSON`-Konfigurationsdatei sein.
`-e, --expression "regex" ["hex_color"]`	Fügt einen neuen Regex hinzu, der bei der Zensur verwendet werden soll. Wiederholung derselben Regex überschreibt die zuvor Eingegebenen. Einem auf diese Art und Weise angegebenen Regex kann ein Farbcode folgen (siehe Use Case Nr. 08).
`-h, --help`	Gibt den Anleitungstext aus.
`-m, --censor-marked`	Stellt ein, dass nur zuvor in einem anderen Programm markierte Stellen zensiert werden.
`-o, --out "out"`	Setzt die Ausgabedatei. Dies kann entweder eine PDF-Datei oder bloß ein Verzeichnis (Verzeichnistrennzeichen: '/') sein (siehe Use Case Nr. 06).
`-u, --censor-unmarked`	Stellt ein, dass zuvor in einem anderen Programm markierte Stellen von der Zensur ausgeschlossen werden.
`-v, --verbose`	Setzt die Ausführlichkeit der Protokollausgaben. Zum Beispiel würde die Eingabe '`-v -v -v`' oder '`-vvv`' die Ausführlichkeit auf Stufe 3 (`Level.WARN`) stellen.
`-V, --version`	Gibt die Versionsnummer des installierten PDF-Zensors aus.

5.3 Konfigurationsdatei

Zusätzlich zu den Kommandozeilenargumenten lässt sich der PDF-Zensor auch durch eine `JSON`-Konfigurationsdatei anpassen. Die Standardkonfigurationsdatei lässt sich, wie in Use Case Nr. 07 beschrieben, temporär durch die angegebene Konfigurationsdatei ersetzen. Eine Konfigurationsdatei dient dabei nur als Grundlage: Argumente, die sich mit welchen der Kommandozeile schneiden, werden durch diese überschrieben. Eine Ausnahme bilden reguläre Ausdrücke, denn diese werden denen der Konfigurationsdatei mit einer höheren Priorität hinzugefügt.

Grundsätzlich erlaubt die Konfigurationsdatei die gleichen Anpassungen wie die Kommandozeilenargumente. Das Anzeigen der Versionsnummer oder des Hilfetextes sowie die Angabe einer Eingabedatei oder einer temporären Konfigurationsdatei ist jedoch exklusiv durch die jeweiligen Kommandozeilenargumente möglich.

Zusätzliche Anpassungen, die dahingegen nur über die Konfigurationsdatei erfolgen können, sind die Angabe einer Reihe von Standardfarben, welche als Farben für Regex dienen, die ohne zugehörige Farbe hinzugefügt wurden, die Angabe der Linkfarbe sowie die Angabe einer Reservefarbe, die nur dann verwendet wird, wenn für einen Zensurbalken sonst keine Farbe vorliegen würde.

Die Syntax der Konfigurationsdatei ist ähnlich den Kommandozeilenargumenten und ist im Folgenden kurz umrissen (optionale Argumente der einzelnen Schlüssel-Wert-Paare sind dabei in <> eingeschlossen):

Schlüssel	**Wert**	
`"out"`	`"out"`	Setzt die Ausgabedatei. Analog zum entsprechenden Kommandozeilenargument.
`"verbose"`	`"Level"`	Setzt das Protokolllevel durch die schriftliche Bezeichnung (zum Beispiel "`all`" oder "`DEBUG`"; die Großschreibung spielt keine Rolle).
`"verbose"`	`Level`	Setzt das Protokolllevel durch die numerische Bezeichnung (von Null als `Level.OFF` in Einerschritten aufwärts).
`"censor"`	`"marked | unmarked"`	Setzt den Modus, nach dem PDF-Zensor arbeiten soll. Wird dieser nicht explizit oder falsch angegeben, so wird der Standardmodus (`ALL`) verwendet.
`"defaultColors"`	`["hex_color", ...]`	Setzt eine Liste von hexadezimal-Farbcodes, die regulären Ausdrücken zugewiesen werden, die ohne zugehörige Farbe definiert wurden.
`"expressions"`	`[{"regex" <,"hex_color">}, ...]`	Setzt eine Liste an regulären Ausdrücken und optionalen hexadezimal-Farbcodes; analog zum entsprechenden Kommandozeilenargument.
`"fallbackColor"`	`"hex_color"`	Setzt die Reservefarbe, sollte ein Zensurbalken sonst keine Farbe besitzen.
`"linkColor"`	`"hex_color"`	Setzt die Farbe, mit der Zensurbalken von Links gefärbt werden.

6 Probleme und Risiken

1. **Wenn** eine Vektorgrafik im Dokument existiert und zensiert werden soll
   Dann könnte PDF-Zensor diese Vektorgrafik als Teil des Dokumentlayouts interpretieren.
   Konsequenz: PDF-Zensor zensiert Teile einer Vektorgrafik nicht oder nur partiell
2. **Wenn** das Dokumentlayout aus Vektorgrafiken, wie zum Beispiel Trennlinien, besteht
   Dann könnte PDF-Zensor diese Teile des Dokumentlayouts als zu zensierende Vektorgrafik interpretieren
   Konsequenz: PDF-Zensor zensiert Teile des Dokumentlayouts oder fasst diese sogar mit Teilen einer Vektorgrafik zusammen
3. **Wenn** mehrere Vektorgrafiken an einer Stelle der PDF ein Schema ergeben
   Dann wird jede Vektorgrafik nur einzeln erkannt
   Konsequenz: Die Vektorgrafiken werden einzeln und nicht als Grafikkomplex zensiert
4. **Wenn** am Anfang einer Textzeile der Text eingerückt ist
   Dann könnte dieser Leerraum basierend auf den darüber- oder darunterliegenden Zeilen als Teil der Zeile erkannt werden
   Konsequenz: Der Leerraum wird ungewollt zensiert
5. **Wenn** ein Leerzeichen zwei Wörter in einem Text trennt oder die Buchstaben einen großen layoutbasierten Abstand zueinander haben
   Dann könnten Buchstaben beziehungsweise Wörter als einzelne Texte erkannt werden
   Konsequenz: Die Leerräume werden nicht mitzensiert
6. **Wenn** die gewählten Libraries in manchen Aufgabenbereichen eine ungenügende Leistung erbringen
   Dann müssen neue Libraries gesichtet und getestet werden
   Konsequenz: Das Projekt verzögert sich
7. **Wenn** die gewählten PDF-Beispiele nicht alle zu beachtenden Fälle abdecken
   Dann entstehen neue Sonderfälle, die beachtet und bearbeitet werden müssen
   Konsequenz: Das Projekt verzögert sich
8. **Wenn** durch Serverfehlfunktionen ein Datenverlust eintritt
   Dann müssen wir das Projekt aus Backups wiederherstellen
   Konsequenz: Das Projekt verzögert sich
9. **Wenn** ein oder mehrere Teammitglieder erkranken oder für die weitere Bearbeitung des Projektes anderweitig nicht zur Verfügung stehen
   Dann muss die Arbeit von anderen Teammitgliedern aufgefangen werden
   Konsequenz: Das Projekt verzögert sich
10. **Wenn** die Implementierung eines oder mehrerer Teilprogramme länger dauert, als vorerst angenommen
    Dann fehlt dem Team Zeit zur Implementierung anderer Funktionen
    Konsequenz: Das Projekt verzögert sich
11. **Wenn** das Team für ein wichtiges Feature keine zufriedenstellende Lösung findet
    Dann muss das Team eine Alternativlösung suchen
    Konsequenz: Das Projekt verzögert sich
12. **Wenn** der finale Test durch die Abnahmetests Mängel aufzeigt
    Dann hat das Team 7 Tage bis zur finalen Abgabe Zeit, um diese Mängel zu beheben
    Konsequenz: Für die Abgabe eines mangelfreien Projektes muss das Team die Mängel innerhalb kürzester Zeit ausbessern, damit die Abnahmetests bestanden werden

Abhilfe

1. Mehrere Vektorgrafiken an einer Dokumentstelle sind ein Indiz dafür, dass die Vektorgrafiken ein Schema ergeben, welches zensiert werden sollte. Wenn sich die Vektorgrafik nicht im Bereich des Headers oder Footers befindet, ist dies ein starkes Indiz dafür, dass diese Vektorgrafik zensiert werden sollte. Aus diesen Indizien lässt sich ableiten, ob die Vektorgrafik als zu zensierendes Element einzustufen ist.
2. Teile des Dokumentlayouts sind fast ausschließlich im Header oder im Footer zu finden. Daher sollten Vektorgrafiken basierend auf ihrer Position als Teile des Dokumentlayouts interpretiert und nicht zensiert werden.
3. Wenn sich Vektorgrafiken überlagern oder in einer hohen Dichte ohne trennenden Text oder Bilder vorkommen, kann davon ausgegangen werden, dass die Vektorgrafiken ein Schema oder Schaubild ergeben, welches wie ein Bild zu zensieren ist.
4. Die Einrückung muss durch den PDF-Zensor als solche erkannt werden (es sei denn, es liegen mehrere Spalten vor).
5. Auf einer Höhe liegende Textelemente müssen als Zeile erkannt und durch ein langes Rechteck zensiert werden.
6. Die ausgewählten Libraries werden bezüglich ihrer Einsatztauglichkeit geprüft.
7. Der Kunde stellt Beispiel-PDF-Dateien zur Verfügung.
8. Der Server muss ausreichend gewartet werden und es müssen regelmäßig Backups des Projektfortschrittes erstellt werden.
9. Die einzelnen Teammitglieder müssen ausreichende Pufferzeiten in ihr Arbeitspensum einbauen, um im Notfall Arbeit anderer auffangen zu können.
10. Das Team muss ausreichend Pufferzeiten einbauen, um mögliche Verzögerungen auffangen zu können.
11. Das Team muss sich bei möglichen Problemfeatures in der Umsetzung dieser genügend absichern.
12. Das Team sollte schon nach der Fertigstellung von großen Teilen des Projektes mit den Abgabetests die Funktionsweise der Software testen.

7 Optionen zur Aufwandsreduktion

7.1 Mögliche Abstriche

Falls die Zeit am Ende des Projektes nicht ausreicht, um alle Funktionen und Anforderungen zu implementieren, würden wir mögliche Abstriche an folgenden Anforderungen und Funktionen machen.

Eine optionale Anforderung ist die Erstellung einer Website, auf der der PDF-Zensor als Webtool für die breite Öffentlichkeit zur Verfügung gestellt werden soll. Sollte die Zeit am Ende nicht ausreichen, werden zuerst an dieser Anforderung Abstriche vorgenommen, um die volle Funktionsfähigkeit des Tools zu erhalten.

Sollten weiterhin Abstriche notwendig sein, so werden diese an anderen optionalen Funktionen des Tool vorgenommen. Dies kann zum Beispiel das Streichen von der Möglichkeit, mehrere reguläre Ausrücke anzugeben, die Farbeinstellungen zu ändern oder ein "`-r`" Operator anzugeben, um komplette Ordner zu zensieren, bedeuten.

7.2 Inkrementelle Arbeit

In diesem Projekt wird inkrementell gearbeitet, das gesamte Projekt wird also in einzelne, kleine Bausteine zerlegt. Diese Bausteine werden nach ihrer Wichtigkeit sortiert und in der entsprechenden Reihenfolge implementiert.

Alle Bausteine zusammen bilden den Backlog. Dieser wird wöchentlich zusammen mit dem Kunden sortiert, damit die wichtigsten Elemente möglichst früh implementiert werden. Eine mögliche Sortierung der Use Cases könnte folgendermaßen aussehen:

1. Komplette PDF-Datei zensieren (Use Case Nr. 03)
2. Ausgangsdatei setzen (Use Case Nr. 06)
3. Markierte Stellen in einer PDF-Datei zensieren (Use Case Nr. 04)
4. Nicht-markierte Stellen in einer PDF-Datei zensieren (Use Case Nr. 05)
5. Zusätzliche Regex hinzufügen (Use Case Nr. 08)
6. Temporäre Konfigurationsdatei setzen (Use Case Nr. 07)
7. Protokollausführlichkeit setzen (Use Case Nr. 09)
8. Hilfe anzeigen (Use Case Nr. 01)
9. Versionsnummer anzeigen (Use Case Nr. 02)
10. (Website erstellen)

8 Glossar

BlackBox-Test Sofwaretest, der nur von der Spezifikation abgeleitet ist, ohne die innere Funktionsweise des Programmes zu kennen

CI Kontinuierliche Integration der Software durch automatisierte Verwaltung

CLI Command Line Interface – Kommandozeilenschnittstelle zur textuellen Steuerung von Programmen

Git Freie Software zur verteilten Versionsverwaltung von Dateien

GUI Grafische Schnittstelle zur grafischen Steuerung von Programmen

PDF Portable Document Format – ein plattformunabhängiges Dateiformat für digitale Dokumente. In diesem Dokument auch synonym verwendet für Dateien, die in diesem Format gespeichert sind ("PDF-Dateien").

Regex Reguläre Ausdrücke zur Filterung von Text nach definierten Vorschriften

Vektorgrafik Eine Grafik, die im Gegensatz zu Rastergrafiken, welche aus Bildpunkten ("Pixeln") besteht, aus Zeichenvorschriften von primitiven Formen zusammengesetzt ist.

stdout Die Standardausgabe. Sofern sie nicht umgeleitet wurde ist das die Konsole.

JSON JavaScript Object Notation – hier das Dateiformat der Konfigurationsdatei

9 Abnahme-Testfälle

Installation

Setup	Die Installationsdatei liegt im Arbeitsverzeichnis.
Input	Aufruf der Installationsdatei.[1]
Expected	Der PDF-Zensor ist installiert und über den Befehl "`pdf-zensor`" aufrufbar.

Fehlerfall – invalide Argumente (keine Argumente)

Setup	*Kein Setup*
Input	Aufruf mit "`pdf-zensor`".
Expected	Es wird ein Fehler ausgegeben, der besagt, dass die Angabe einer Eingabedatei erforderlich ist.

Fehlerfall – invalide Argumente (zu viele Argumente)

Setup	*Kein Setup*
Input	Aufruf mit "`pdf-zensor "Zensiere1.pdf" "Zensiere2.pdf"`".
Expected	Es wird ein Fehler ausgegeben, der besagt, dass die Angabe einer Eingabedatei nur einmal erfolgen darf.

Fehlerfall – invalide Argumente (nicht existente Eingabedatei)

Setup	Ausgehend vom Arbeitsverzeichnis ist keine Datei "`NichtExistenteDatei.pdf`" zu finden.
Input	Aufruf mit "`pdf-zensor "NichtExistenteDatei.pdf"`".
Expected	Es wird ein Fehler ausgegeben, der besagt, dass die angegebene Eingabedatei nicht gefunden werden konnte.

Hilfe anzeigen ohne weitere Argumente

Setup	*Kein Setup*
Input	Aufruf mit "`pdf-zensor --help`" oder "`pdf-zensor -h`".
Expected	Der Hilfetext wird in den stdout ausgegeben und das Programm beendet sich.

Hilfe anzeigen – existierende Eingabedatei angegeben

Setup	Ausgehend vom Arbeitsverzeichnis ist die Datei "`ExistenteDatei.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor -h "ExistenteDatei.pdf"`".
Expected	Der Hilfetext wird in den stdout gegeben und das Programm beendet sich. Es wird weder die Eingabedatei geändert, noch eine zensierte Ausgabedatei erstellt.

[1] Falls der Schritt "Debian Package erstellen" in der polishing-Phase erreicht wurde, ist dies ein Debian Package

Versionsnummer anzeigen ohne weitere Argumente

Setup	*Kein Setup*
Input	Aufruf mit "`pdf-zensor --version`" oder "`pdf-zensor -V`".
Expected	Die Versionsnummer wird in der Kommandozeile ausgegeben und das Programm beendet sich.

Versionsnummer anzeigen – existierende Eingabedatei angegeben

Setup	Ausgehend vom Arbeitsverzeichnis ist die Datei "`ExistenteDatei.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor -V "ExistenteDatei.pdf"`".
Expected	Die Versionsnummer wird in den stdout gegeben und das Programm beendet sich. Es wird weder die Eingabedatei geändert, noch eine zensierte Ausgabedatei erstellt.

Komplette PDF zensieren – Eingabedatei existiert

Setup	Ausgehend vom Arbeitsverzeichnis ist die Datei "`ZuZensieren.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "ZuZensieren.pdf"`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`ZuZensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Komplette PDF zensieren – Eingabedatei existiert in Ordner

Setup	Ausgehend vom Arbeitsverzeichnis ist im Ordner "`MeinOrdner`" die Datei "`ZuZensieren.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "MeinOrdner/ZuZensieren.pdf"`".
Expected	Im Verzeichnis der Eingabedatei (also in "`MeinOrdner/`") wird eine Datei namens "`ZuZensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Komplette PDF zensieren – Ausgabedatei benennen

Setup	Ausgehend vom Arbeitsverzeichnis ist die Datei "`ZuZensieren.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "ZuZensieren.pdf" -o "MeinAusgabename.pdf"`" oder "`pdf-zensor "ZuZensieren.pdf" --out "MeinAusgabename.pdf"`".
Expected	Im Arbeitsverzeichnis des Programmes wird eine Datei namens "`MeinAusgabename.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Komplette PDF zensieren – Ausgabedatei benennen und überschreiben

Setup Ausgehend vom Arbeitsverzeichnis ist die Datei "`ZuZensieren.pdf`" zu finden, welche für den PDF-Zensor lesbar ist. Im Arbeitsverzeichnis des Programmes existiert bereits eine Datei namens "`MeinAusgabename.pdf`".

Input Aufruf mit "`pdf-zensor "ZuZensieren.pdf" -o "MeinAusgabename.pdf"`" oder "`pdf-zensor "ZuZensieren.pdf" --out "MeinAusgabename.pdf"`".

Expected Im Arbeitsverzeichnis des Programmes wird die Datei "`MeinAusgabename.pdf`" überschrieben, sodass diese nun der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Komplette PDF zensieren – Eingabe überschreiben

Setup Ausgehend vom Arbeitsverzeichnis ist die Datei "`ZuZensieren.pdf`" zu finden, welche für den PDF-Zensor lesbar ist.

Input Aufruf mit "`pdf-zensor "ZuZensieren.pdf" -o "ZuZensieren.pdf"`" oder "`pdf-zensor "ZuZensieren.pdf" --out "ZuZensieren.pdf"`".

Expected Die Eingabedatei "`ZuZensieren.pdf`" wird überschrieben, sodass in dieser nun keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Komplette PDF zensieren – Ausgabeverzeichnis angeben

Setup Ausgehend vom Arbeitsverzeichnis existiert das Verzeichnis "`AusgabeOrdner`" und die Datei "`ZuZensieren.pdf`" ist zu finden und für den PDF-Zensor lesbar.

Input Aufruf mit "`pdf-zensor "ZuZensieren.pdf" -o "AusgabeOrdner"`" oder "`pdf-zensor "ZuZensieren.pdf" --out "AusgabeOrdner"`".

Expected Ausgehend vom Arbeitsverzeichnis des Programmes wird in dem Verzeichnis "`AusgabeOrdner`" eine Datei namens "`ZuZensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Nur markierte Stellen der PDF zensieren – keine Markierungen

Setup Ausgehend vom Arbeitsverzeichnis existiert die Datei "`PartiellZens.pdf`", die für PDF-Zensor lesbar ist und in der keine Stellen in einem anderen Programm markiert wurden.

Input Aufruf mit "`pdf-zensor "PartiellZens.pdf" -m`" oder "`pdf-zensor "PartiellZens.pdf" --censor-marked`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`PartiellZens_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen.

Nur markierte Stellen der PDF zensieren – mit Markierungen

Setup Ausgehend vom Arbeitsverzeichnis existiert die Datei "`PartiellZens.pdf`", die für PDF-Zensor lesbar ist und in der zuvor Stellen in einem anderen Programm markiert wurden.

Input Aufruf mit "`pdf-zensor "PartiellZens.pdf" -m`" oder "`pdf-zensor "PartiellZens.pdf" --censor-marked`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`PartiellZens_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen und alle Inhalte der markierten Stellen zensiert wurden. Das heißt, dass Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Nur unmarkierte Stellen der PDF zensieren – keine Markierungen

Setup Ausgehend vom Arbeitsverzeichnis existiert die Datei "`PartiellZens.pdf`", die für PDF-Zensor lesbar ist und in der keine Stellen in einem anderen Programm markiert wurden.

Input Aufruf mit "`pdf-zensor "PartiellZens.pdf" -u`" oder "`pdf-zensor "PartiellZens.pdf" --censor-unmarked`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`PartiellZens_cens.pdf`" erstellt, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Nur unmarkierte Stellen der PDF zensieren – mit Markierungen

Setup Ausgehend vom Arbeitsverzeichnis existiert die Datei "`PartiellZens.pdf`", die für PDF-Zensor lesbar ist und in der zuvor Stellen in einem anderen Programm markiert wurden.

Input Aufruf mit "`pdf-zensor "PartiellZens.pdf" -u`" oder "`pdf-zensor "PartiellZens.pdf" --censor-unmarked`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`PartiellZens_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen und alle Inhalte von Stellen, die nicht markiert waren, zensiert wurden. Das heißt, dass Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden.

Temporäre Konfigurationsdatei setzen – diese existiert

Setup	Eine Datei "`zensieren.pdf`" und eine Konfigurationsdatei "`config.json`" existieren, die beide gültige Eingaben für den PDF-Zensor sind.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -c "config.json"`" oder "`pdf-zensor "zensieren.pdf" --config "config.json"`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, welche gemäß der angegebenen Konfigurationsdatei zensiert wurde.

Temporäre Konfigurationsdatei setzen – diese existiert nicht

Setup	Eine Datei "`zensieren.pdf`" existiert und ist für den PDF-Zensor lesbar und eine Konfigurationsdatei "`config.json`" existiert nicht.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -c "config.json"`" oder "`pdf-zensor "zensieren.pdf" --config "config.json"`".
Expected	Es wird ein Fehler ausgegeben, der besagt, dass die angegebene Konfigurationsdatei nicht gefunden werden konnte.

Temporäre Konfigurationsdatei setzen – Argumente werden überschrieben

Setup	Eine Datei "`zensieren.pdf`" und eine Konfigurationsdatei "`config.json`" existieren, die beide gültige Eingaben für den PDF-Zensor sind. In "`config.json`" kommt "`"verbose":"off"`" vor.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -c "config.json" -vvvvvvv`" oder "`pdf-zensor "zensieren.pdf" --config "config.json" -vvvvvvv`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, welche gemäß der angegebenen Konfigurationsdatei zensiert wurde. Anders als in der Konfigurationsdatei angegeben, werden während des Zensurvorgangs alle Protokollausgaben angezeigt.

Temporäre Konfigurationsdatei setzen – Argumente werden ergänzt und priorisiert

Setup	Eine Datei "`zensieren.pdf`" und eine Konfigurationsdatei "`config.json`" existieren, die beide gültige Eingaben für den PDF-Zensor sind. In "`config.json`" kommt "`"expressions":[{"regex":"hallo", "color":"#FF0000"}]`" vor.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -c "config.json" -e "hallo" "#00FF00"`" oder "`pdf-zensor "zensieren.pdf" --config "config.json" --expression "hallo" "#00FF00"`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, welche weitgehend gemäß der angegebenen Konfigurationsdatei zensiert wurde. Insbesondere sind jedoch alle Vorkommnisse von *"hallo"* mit einem grünen (und nicht roten) Zensurbalken zensiert worden.

Weitere Regex setzen – ungültiger Farbcode

Setup	Ausgehend vom Arbeitsverzeichnis liegt die Datei "`zensieren.pdf`" vor, die für PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -e "." "rot"`" oder "`pdf-zensor "zensieren.pdf" --expression "." "rot"`".
Expected	Es wird ein Fehler ausgegeben, da der Farbcode kein gültiger Hexadezimal-Farbcode ist. Zulässig sind nur Farbcodes, die den Richtlinien, die in Kapitel 5 sowie Use Case Nr. 08 definiert wurden, folgen.

Weitere Regex setzen – mit Farbe

Setup	Ausgehend vom Arbeitsverzeichnis liegt die Datei "`zensieren.pdf`" vor, die für PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -e "hallo welt" "#00FF00"`" oder "`pdf-zensor "zensieren.pdf" --expression "hallo welt" "#00FF00"`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe und des Weiteren alle Vorkommnisse von *"hallo welt"* mit einem grünen Zensurbalken zensiert wurden.

Weitere Regex setzen – ohne Farbe

Setup	Ausgehend vom Arbeitsverzeichnis liegt die Datei "`zensieren.pdf`" vor, die für PDF-Zensor lesbar ist.
Input	Aufruf mit "`pdf-zensor "zensieren.pdf" -e "hallo welt"`" oder "`pdf-zensor "zensieren.pdf" --expression "hallo welt"`".
Expected	Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe und des Weiteren alle Vorkommnisse von *"hallo welt"* mit einem Zensurbalken zensiert wurden, dessen Farbe automatisch von dem Programm aus einer Liste von Standardfarben ausgewählt wurde.

Weitere Regex setzen – mehrere ohne Farbe

Setup Ausgehend vom Arbeitsverzeichnis liegt die Datei "`zensieren.pdf`" vor, die für PDF-Zensor lesbar ist.

Input Aufruf mit "`pdf-zensor "zensieren.pdf" -e "hallo" -e "welt"`" oder "`pdf-zensor "zensieren.pdf" --expression "hallo" --expression "welt"`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden. Des Weiteren wurden alle Vorkommnisse von *"hallo"* mit einem Zensurbalken, dessen Farbe automatisch von dem Programm aus einer Liste von Standardfarben ausgewählt wurde, und alle Vorkommnisse von *"welt"* mit einem Zensurbalken in einer anderen Farbe, welche auch aus der Liste von vordefinierten Standardfarben stammt, zensiert.

Weitere Regex setzen – mit Präzedenz

Setup Ausgehend vom Arbeitsverzeichnis liegt die Datei "`zensieren.pdf`" vor, die für PDF-Zensor lesbar ist.

Input Aufruf mit "`pdf-zensor "zensieren.pdf" -e "ein kleiner test" "#FF0000" -e "kleiner" "#00FF00"`" oder "`pdf-zensor "zensieren.pdf" --expression "ein kleiner test" "#FF0000" --expression "kleiner" "#00FF00"`".

Expected Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch Zensurbalken einer und der restliche Text durch Zensurbalken einer (anderen) Farbe zensiert wurden. Des Weiteren wurden alle Vorkommnisse von *"ein kleiner test"* mit einem roten Zensurbalken zensiert, danach wurden alle noch nicht zensierten Vorkommnisse von *"kleiner"* grün zensiert. *"kleiner"* ist also nur grün zensiert, wenn es nicht in *"ein kleiner test"* vorkam.

Ausführlichkeit der Protokollausgaben setzen

Setup Ausgehend vom Arbeitsverzeichnis existiert die Datei "`zensieren.pdf`", die für PDF-Zensor lesbar ist.

Input Aufruf mit "`pdf-zensor "zensieren.pdf" -v -v -v -v -v`", "`pdf-zensor "zensieren.pdf" -vvvvv`" oder "`pdf-zensor "zensieren.pdf" --verbose`", wobei `--verbose` fünf Mal vorkommt.

Expected Das Level der Protokollausgaben ist auf "`DEBUG`" gesetzt, das heißt, dass während des Zensurvorgangs alle Protokollausgaben, die ungenauer als dieses Level sind, in den stdout ausgegeben werden (von diesem Level kommen definitiv Protokollausgaben vor). Im Verzeichnis der Eingabedatei wird eine Datei namens "`zensieren_cens.pdf`" erstellt, die der Eingabedatei entspricht, in der jedoch keine Metadaten mehr vorliegen, alle Bilder dem Standardbild entsprechen, welches durch PDF-Zensor gegeben wurde, Links durch blaue und der restliche Text durch schwarze Zensurbalken zensiert wurden.