Hochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Studiengang Ingenieurinformatik

Exposé zur Bachelorarbeit

Vorläufiger Arbeitstitel: Entwicklung eines workflow-basierten Software-Interfaces für den Krause Lumisetter

Calvin Paunovic (1208155)

1 Thema, Problemstellung und Zielsetzung

Die Bachelorarbeit beschäftigt sich mit der Entwicklung einer Softwarelösung zur Unterstützung des Druckvorstufenprozesses für den Krause Lumisetter. Ziel ist es, den Workflow von der Dateieingabe über die Optimierung der Plattenbelegung bis zur Maschinenansteuerung zu digitalisieren und zu vereinfachen, um Zeitaufwand und Materialverlust zu reduzieren.

Bisher werden Aufträge manuell auf den teuren Magnesiumplatten platziert, was fehleranfällig ist und zu Restplatten führt. Ein bereits entwickelter Optimierungsalgorithmus berechnet die effizienteste Anordnung der Aufträge, sodass Flächen optimal genutzt und Restplatten weiterverwendet werden können. Unter einem „Job“ versteht man dabei die verschiedenen Motive oder Layouts, die auf der Magnesiumplatte belichtet werden.

Die Arbeit konzentriert sich auf die Umsetzung eines workflow-basierten Software-Prototyps, der die Bedienerführung verbessert, die Ergebnisse des Optimierungsalgorithmus verständlich aufbereitet und Funktionen wie Lagerverwaltung integriert. Die Optimierungsalgorithmen selbst sind nicht Bestandteil der Arbeit, sondern dienen als Grundlage für die Prototyp-Entwicklung.

2 Vorgehensweise der Arbeit

Die Arbeit wird in mehreren Phasen durchgeführt, um die Ziele systematisch zu erreichen:

1. **Workflow-Analyse:** Bestehende Prozesse untersuchen, Pflichtenheft erstellen.
2. **Modellierung:** Use-Case-Diagramme, Klassendiagramme, Aktivitätsdiagramme.
3. **Softwareentwicklung:** Umsetzung der zentralen Features.
4. **Evaluation:** Tests mit Beispielaufträgen, Feedbackgespräche, Verbesserungsvorschläge ableiten.

3 Geplante Technologien

Für die Umsetzung des Prototyps sind folgende Technologien vorgesehen:

- **Programmiersprachen:** JavaScript (Frontend), Python (Backend). Der bestehende Optimierungsalgorithmus ist in Java implementiert, soll aber, falls praktikabel, in Python übertragen werden.
- **Frameworks und Frontend-Technologien:** Webbasierte Oberfläche (z. B. Vue.js oder ein vergleichbares Framework). Ziel ist eine zukunftsfähige Lösung für PC und mobile Geräte.
- **Datenhaltung:** SQLite als leichtgewichtige relationale Datenbank zur Verwaltung von Aufträgen, Restplatten und Lagerbeständen.

Schnittstellen: Eingabe über JSON-Datei mit Jobinformationen, Ausgabe über JSON-Datei mit Layoutbeschreibung (Dateiname, Koordinaten), Kommunikation zwischen Frontend und Backend über Websockets oder REST-API,

- **Plattform:** Plattformübergreifend lauffähig (Web-Applikation).
- **Mockup-Deisgn:** Erstellung der Benutzeroberfläche mit Balsamiq.

4 Features

Der Prototyp soll folgende zentrale Funktionen umfassen:

- **Visualisierung:** Bildliche Vorschau und Beschreibung der eingehenden Jobs sowie der Lösungsvorschläge des Optimierungsalgorithmus
- **Filter, Sortieren und Kategorisieren:** Mehrfaches Filtern und Sortieren der Jobliste nach verschiedenen Kriterien, automatisierte oder manuelle Kategorisierung
- **Drag & Drop:** Jobs manuell neu anordnen und dem Optimierungsalgorithmus für neue Vorschläge übergeben
- **Dynamisches Interface:** Automatische Aktualisierung der Oberfläche durch kontinuierliches Abfragen von Status- und Jobinformationen
- **Lagerverwaltung:** Verwaltung von Restplatten, Einsehen verfügbarer Restplatten und Nutzung durch den Algorithmus

5 Benutzeroberfläche (UI)

Die UI unterstützt den Workflow von Eingabe bis Ausgabe mit klarer Bedienführung. Sie umfasst die Hauptfunktionen:

- **Input/Eingangsordner:** Import von Aufträgen und Verwaltung nach Plattendicke
- **Jobliste:** Anzeige, Sortierung und Filterung der Aufträge
- **Lösungsvorschläge:** Optimierungsergebnisse darstellen
- **Ausgabe:** Rückmeldung an Benutzer und Export von Layout-Datei.

6 Zeitplan

Der Arbeitsplan gliedert sich in folgende Meilensteine:

- **Kurzfristig:** Erstellung eines Mockups der Benutzeroberfläche und Übertragung des Optimierungsalgorithmus von Java nach Python.
- **Während der Bachelorarbeit:**

  - Auswahl und Begründung des geeigneten Frontend-Frameworks.
  - Installation des Grundgerüsts (Framework, Python, Frontend, SQLite).
  - Anbindung des Frontends an das Backend (Kommunikation über Websocket/REST).
  - Anbindung des Backends an den Eingangsordner und Befüllung der Datenbank.
  - Erstellung und Dokumentation des Datenmodells.
  - Umsetzung eines Mockups in JavaScript (für die Bachelorarbeit; separates Mockup für Kundenprojekt nicht relevant).
  - Integration der Logik für den Aufruf des PlateOptimizers im Backend.
  - Präsentation der Ergebnisse in der UI.
  - Implementierung einer Minimalversion der wichtigsten Funktionen, anschließend Erweiterung/Skalierung.
  - Ausgabe einer JSON-Datei zur Layoutbeschreibung.
  - Gestaltung der Benutzeroberfläche (Farbschema, Layout) in einem finalen Schritt.
  - Durchführung von Testszenarien zur Evaluation.
- **Schreibphase:** Parallel laufende Dokumentation.

7 Vorläufige Literatur

1. Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3. Auflage, Addison-Wesley, 2004.
2. Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6. Auflage, Pearson, 2017.

8 Vorläufige Gliederung der Arbeit

1. Einleitung: Motivation, Problemstellung, Zielsetzung
2. Theoretische Grundlagen: Workflow-Management, Interface-Design
3. Analyse des Workflows: Beschreibung der Prozessschritte, Softwareanforderungen
4. Modellierung: UML-Diagramme
5. Prototypische Umsetzung: Architektur, Benutzeroberfläche, zentrale Funktionen
6. Evaluation: Testszenarien, Anwenderfeedback, Ergebnisdiskussion
7. Fazit und Ausblick