OP-Scheduling – Erweiterung

Bericht der Erweiterung des OP-Scheduling

Fach: Angewandte Betrieblich Systemforschung

Professor: Prof. Dr. Michael Grütz

Autor: Stefan Brecht

Mat-Nr.: 277590

E-Mail: [Stefan.Brecht@htw-konstanz.de](mailto:Stefan.Brecht@htw-konstanz.de)

Semester: 8. Semester Wirtschaftsinformatik, Fachrichtung Software-Engineering

Datum:

Inhaltsverzeichnis

[1 Stand vor der Überarbeitung 3](#_Toc199513007)

[1.1 Features Realisiert 3](#_Toc199513008)

[1.2 Features noch nicht Realisiert 3](#_Toc199513009)

[2 Stand nach der Überarbeitung 4](#_Toc199513010)

[2.1 Features Realisiert 4](#_Toc199513011)

[2.2 Features noch nicht Realisiert 4](#_Toc199513012)

[3 Zeitaufwand für die Realisierung 5](#_Toc199513013)

[3.1 Zeitaufwand 5](#_Toc199513014)

# Stand vor der Überarbeitung

In diesem Kapitel wird der Stand vor der Überarbeitung aufgelistet.

## Features Realisiert

Vor der Überarbeitung waren folgende Features Realisiert:

* Daten können eingegeben werden
  + Über das GUI können alle nötigen Daten eingegeben werden.
* LP-Ansatz wird erzeugt
  + Durch bestätigen der Eingaben wird der LP-Ansatz erzeugt
  + Der LP-Ansatz wird an den Solver geschickt
  + Das Ergebnis des Solver wird eingelesen
  + Solver Ergebnis wird interpretiert
* Ergebnis wird ausgegeben
  + Das Interpretierte Ergebnis wird über das GUI ausgegebenen

Diese Features haben ohne größere Fehler funktioniert, wobei der LP-Ansatz von mir nicht überprüft wurde.

## Features noch nicht Realisiert

* Speichern/Öffnen Funktion nicht vorhanden
  + Es ist nicht möglich die eingegeben Daten zu speichern
* Solver Pfad kann nicht geändert werden
  + Es ist nicht möglich den Solver Pfad über die GUI zu ändern
* Es kann keine Wartung für einen Saal angegeben werden
  + Es ist nicht möglich eine Wartung für einen Saal zu planen
* Anzahl der Operationssäle ist auf 2 beschränkt
  + Es ist nicht möglich mehr oder weniger als 2 Operationssäle auszuwählen
* Grafischer Belegungsplan wird nicht ausgegeben
  + Es ist nicht möglich das Ergebnis Grafisch einzusehen
* Die Belegung von möglichst wenig Sälen kann nicht berechnet werden
  + Es kann nur die optimale Auslastung berechnet werden.

Diese Features waren nicht Implementiert.

# 

# Stand nach der Überarbeitung

In diesem Kapitel wird der Stand nach der Überarbeitung aufgelistet und die neuen Features kurz erklärt.

## Features Realisiert

* Speichern Funktion hinzugefügt
  + Das Speichern aller eingegebenen Daten ist möglich und funktioniert
  + Es wird die Dateiendung \*.opsa genutzt
  + Es wurde ein Filter implementiert damit nur Dateien des mit der Endung \*.opsa angezeigt werden
* Öffnen Funktion hinzugefügt
  + Die gespeicherten Daten können geöffnet werden
  + Hier wurde der selbe Filter wie für die Speichern-Funktion genutzt
* Der Solver Pfad kann geändert werden und wir in einer INI-Datei gespeichert
  + Der Solver Pfad kann über das GUI eingegeben werden
  + Es wird überprüft ob die Datei welche angegeben wird auch vorhanden ist
  + Der Pfad wird dann in die INI-Datei geschrieben

Die Implementierten Features wurden von mir implementiert und getestet und sind funktionsfähig so wie hier beschrieben. Es wurde falsche Nutzereingaben abzufangen.

## Features noch nicht Realisiert

* Es kann keine Wartung für einen Saal angegeben werden
  + Es ist nicht möglich eine Wartung für einen Saal zu planen
* Anzahl der Operationssäle ist auf 2 beschränkt
  + Es ist nicht möglich mehr oder weniger als 2 Operationssäle auszuwählen
* Grafischer Belegungsplan wird nicht ausgegeben
  + Es ist nicht möglich das Ergebnis Grafisch einzusehen
* Die Belegung von möglichst wenig Sälen kann nicht berechnet werden
  + Es kann nur die optimale Auslastung berechnet werden.

Diese Features sind noch nicht Implementiert.

# Zeitaufwand für die Realisierung

Es wurden in der Vorlesung folgende Features festgelegt:

* Speichern/Öffnen Funktion
* Solver Pfad änderbar über GUI

Es war mir möglich in der vorgegebenen Zeit alle Features zu realisieren.

## Zeitaufwand

|  |  |  |
| --- | --- | --- |
| **Stunden** | **Typ** | **Beschreibung** |
| 2,5 | Selbststudium | **Problem:**  Compilieren des Programmcodes war ohne JBuilder Bibliotheken nicht möglich.  **Vorgehen:**   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 1,5 | Such nach JBuilder Version (google):   * JBuilder 7 gefunden (konnte nicht installiert werden) (0,5) * JBuilder 9 gefunden (konnte auch nicht installiert werden) (0,25) * JBuilderX gefunden (Code konnte nicht Compiliert werden) (0,75) | | 1 | Recherche nach Borland Bibliotheken (google)   * Passende Bibliothek nicht gefunden (0,5) * Aber rausgefunden wie die Bibliothek aus der bestehenden Jar Datei exthrahiert (0,25) * Extrahierte Jar in Eclipse einbinden (0,25) | |
| **Lösung:**  Borland Bibliothek (jetzt im Source-Ordner als Borland.jar) in Eclipse einbinden. |
| 1,5 | Selbststudium | Einarbeiten in Programm-Code. |
| 0,5 | Selbststudium | Schreiben der INI Datei mit Solver Pfad   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,5 | Klasse zum lesen war schon vorhanden. Klasse erweitert um eine schreiben Funktion und Funktion zum ansteuern dieser schreiben Funktion | |
| 1,0 | Selbststudium | **Problem:**  Ohne JBuilder konnte das GUI nicht im Entwurfsmodus angesehen werden.  **Vorgehen:**   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,25 | Menu Punkte von Hand Programmiert | | 0,75 | Fenster für die Pfadeingabe der Solver Datei |   **Lösung:**  Von Hand den nötigen Programmcode zusammensuchen und überarbeiten. |
| 0,5 | Selbststudium | Funktionen zum speichern und öffnen:   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,25 | Funktion zum schreiben per Objektserialisierung eingebaut. (Funktion und Wissen aus der Werbebudgetplanung genutzt) | | 0,25 | Funktion zum öffnen per Objektserialisierung eingebaut. (Funktion und Wissen aus der Werbebudgetplanung genutzt) | |
| 1,0 | Selbststudium | Klasse welche alle Daten des GUI aus dem GUI lesen und in das GUI schreiben kann. Diese Klasse wird dann per Objektserialisierung gespeichert.   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,5 | Zusammensuchen der Klassen welche die Daten des GUIs enthalten | | 0,25 | Klasse gebaut welche auf die GUI-Datenklassen zugreift und alle Daten liest.  Diese Klasse wird Serialisiert. | | 0,25 | Klasse erweitert um die Daten nach dem öffnen wieder in das GUI zu schreiben. | |
| 0,5 | Selbststudium | Implementieren eines Speichern-Fensters:  (Auf wissen aus der Werbebudgetplanung zurückgegriffen)   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,25 | FileChooser für das Speichern eingebaut und MenuPunkt Speichern hinzugefügt. | | 0,25 | Implementieren des Filter mit der Dateiendung \*.wbo | |
| 1,0 | Selbststudium | Testen der implementierten Funktionen und beheben von Bugs:   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 0,25 | Testen der implementierten Funktionen | | 0,5 | Ini-Datei wurde falsch gechrieben, dadurch Syntax Error. Bug behoben durch ändern der Schreiben-Funktion in der INI-Klasse | | 0,25 | Der Datei-Pfad wurde aus dem Java-FileChooser falsch ausgelesen. Durch ändern des zugriffs auf den FileChooser Bug behoben | |
| 0,5 | Selbststudium | Erstellen von Beispiel-Daten (3 Beispiele) |
| 1,5 | Selbststudium | Schreiben der Dokumentationen für die Änderungen:  (Struktur aus der Webebugetplanung-Dokumentation übernommen)   |  |  | | --- | --- | | **Stunden** | **Beschreibung** | | 1 | Schreiben der Dokumentation | | 0,5 | Überarbeiten der Dokumentation nach Besprechung in der Vorlesung | |
| 10,5 |  | ***Summe Gesamt*** |