Mensch-Computer Interaktion SS 2003

Projekt: **Informationssystem *"Reiseverbindungen auf Schiene und Straße"***

Aufgabe 2: Planung, Entwurf und Implementierung mit Anbindung an eine Datenbank
Aufgabe 3: Evaluierung der Funktionalität und der Usability

Abnahme: 26./27. KW

Name: ..... Studiengang/Matr-Nr: .....

Datum: ……..… Unterschrift des Dozenten (wenn erfolgreich): ……..……

Aufgabe 2 (Planung, Entwurf, Implementierung)

Bei der Planung, dem Entwurf und der Implementierung des in Aufgabe 1 beschriebenen Projekts *Reiseverbindungen* müssen Sie folgende Punkte zu beachten.

1. Alle Daten über Reiseverbindungen, Bahnhöfe, Orte, Haltestellen, usw. sind aus einer Datenbank zu laden
    a. **Abfahrts- und Ankunftszeiten** für einen Streckenabschnitt
    b. Namen, Lokalisation, Beschreibungen, usw. von **Bahnhöfen** und **Haltestellen**
    c. **Dauer** und **Preis** eines Reiseabschnitts
    d. Informationen über die **Verkehrsmittel** wie z.B. Klasse, Ausstattung, Sitzplätze, usw.

2. Funktionen
    a. **Suche nach Verbindungen** zwischen zwei Orten, Bahnhöfen und/oder Haltestellen. Der Benutzer soll wählen bzw. ausschließen können, ob Verbindungen mit oder ohne Umsteigen gesucht werden sollen.
    b. Suche nach Verbindungen mit Angabe einer **frühesten Abfahrtzeit** und/oder **spätesten Ankunftszeit**.
    c. Anfrage nach **kürzesten**, **schnellsten** und/oder **billigsten** Verbindungen
    d. Ermittlung und Darstellung mehrerer **alternativer Verbindungen**
    e. Die günstigsten (kürzeste, schnellste, billigste) Verbindungen sollen mit dem **Algorithmus von Dijkstra** ermittelt werden (→ Extra-Vorlesung)
    f. **Tabellarische Übersicht** mehrerer möglicher Verbindungen aufgrund einer Anfrage
    g. **Detaillierte Darstellung** einer ausgewählten Verbindung: ausführliche Route mit Ankunfts- und Abfahrtzeiten aller Stationen, besondere Hervorhebung der Umsteigestationen, Fahrtzeiten, ...

3. User-Interface
    a. Versuchen Sie, beim Entwurf von Interaktionsfolgen und Informationsdarstellungen die **Sicht des Benutzers** einzunehmen. Als Entwickler wissen Sie genau, wie Interaktionen ablaufen oder die Darstellungen von Suchanfrageergebnissen zu interpretieren sind. Beachten Sie, dass Benutzer dieses Wissen nicht haben.
    b. Stellen Sie geeignete Eingabemöglichkeiten für die **Auswahl von Orten, Bahnhöfen, Haltestellen** sowie **Abfahrts-** und **Ankunftszeiten** zur Verfügung. Zum Beispiel müssen Ortsnamen sinnvoll strukturiert werden. Eine andere Möglichkeit ist die Benutzung einer „ungefähr"-Funktion. Überlegen Sie sich für Ihr System ein effizient benutzbares **Konzept**.
    c. Entwerfen Sie klare **Layouts** für Eingabe- und Suchmasken. **Strukturieren** Sie die Eingabefelder in sinnvolle Gruppen.

d. Entwerfen und realisieren Sie ein Konzept für Fehlermeldungen, Warnungen und Hinweise z.B. über die Ausführung von Suchanfragen, Ergebnisse, usw.

4. Vier-Ebenen-Architektur.

   Für die Implementierung des Systems *Reiseverbindungen* sollen Sie eine Vier-Ebenen-Architektur verwenden. So können klar definierte Schnittstellen zwischen den Ebenen eingerichtet werden, die ein weitgehend unabhängiges Arbeiten an den Softwarekomponenten der einzelnen Ebenen ermöglichen.

   a. **Datenbank**. Alle Daten werden aus einer relationalen Datenbank geladen.
   b. **Funktionslogik**. Funktionen wie z. B. der Algorithmus von Dijkstra sind logisch von anderen Ebenen wie Datenbank und User Interface zu trennen. Algorithmische Komponente können so unabhängig entwickelt und getestet werden.
   c. Verwenden Sie **austauschbare Datenstrukturen** (XML) mit entsprechenden **extensible Stylesheets** (XSL). Dadurch sind flexible, leicht änderbare und erweiterbare Strukturen für User Interfaces möglich. Zum Beispiel kann durch die Formulierung spezieller Stylesheet-Regeln ein völlig anderes User Interface für ein Handy oder einen PDA generiert werden, ohne dass die Funktionslogik und die XML-Bäume davon betroffen sind.
   d. Realisieren Sie das **User Interface** durch HTML-Bildseiten mit entsprechenden Cascading Stylesheets (CSS). Mit Hilfe von Cascading Stylesheets kann das grafische Layout (Farben, Hintergründe, Schriftarten, Schriftgrößen, Tabelleneigenschaften, usw.) eines User Interface gestaltet werden. Das Erscheinungsbild gleicher Darstellungsstrukturen kann dadurch leicht an verschiedene Arbeitsumgebungen angepasst werden.

Aufgabe 3 (Evaluierung)

Vor der Übergabe einer Software an einen Kunden müssen Sie die **Funktionalität** und die **Usability** überprüfen und evaluieren. Für die **Abnahme des Praktikums** bedeutet dies folgendes: Ihre System *Reiseverbindungen* wird aus der **Sicht des Benutzers** beurteilt. Herr Poborski und ich werden die Rolle des Benutzers einnehmen und die Funktionalität und die Usability aus dieser Sicht bewerten. Bitte gehen Sie folgende Checklisten durch.

Funktionalität

- Sind alle Funktionen implementiert?
- Arbeiten alle Funktionen korrekt? Testen Sie mit realen und ausreichend vielen und komplexen Testdaten.
- Testen Sie Ihr System auch mit so genannten Randdaten. Was passiert z.B. bei einer Suche, wenn die Ankunftszeit vor der Abfahrtszeit liegt? Dies kann z.B. aufgrund falscher Daten in der Datenbank oder durch falsche Benutzereingaben passieren.
- Sind alle Funktionen mit ausreichenden Fehler- und Warnmeldungen versehen? Schließen Sie technische Fehlermeldung aus.
- Sind alle Funktionen, insbesondere auch die algorithmischen Funktionen genau dokumentiert? Welche Schnittstellen (Eingabedaten, Ausgabedaten) haben die verwendeten Algorithmen?

Usability

- Sind die Funktionen, Eingabefelder und Darstellungsfenster sinnvoll strukturiert? Welche Funktionen und/oder Eingabefelder sollten sich auf einer Bildseite befinden? Welche Informationen benötigt der Benutzer, um eine Aufrage an das System zu formulieren?
- Sind alle Funktionen leicht erreichbar? Legen Sie keine Verstecke in Menüs an.

- Ist die Benutzung aller Funktionen selbsterklärend? Nehmen Sie die Rolle eines Benutzers ein oder lassen Sie die Funktionen von jemandem ausführen, der das System noch nicht kennt. So werden Sie eine schlechte Usability schnell aufdecken.
- Wird dem Benutzer deutlich gemacht, wann das System gerade arbeitet und wann es Eingaben erwartet?
- Wird der Benutzer bei Interaktionsfolgen, die über mehrere Bildseiten gehen, hinreichend geführt?
- Sind widersprüchliche Eingaben ausgeschlossen? Beispiel: Die Ankunftszeit liegt vor der Abfahrtzeit.
- Prüfung der eingegebenen Daten. Die Ausführung einer Anfrage sollte erst dann gestartet werden können, wenn alle benötigten Daten vollständig eingegeben sind.
- Sind syntaktische Fehler bei der Eingabe ausgeschlossen? Beispiel: Ist die Uhrzeit oder das Datum im korrekten Format eingetragen?
- Wird der Benutzer in geeigneter Art und Weise über die Ergebnisse einer Anfrage informiert? Beispiele: Was ist der Grund für eine leere Treffermenge? Wie werden sehr große Treffermengen strukturiert, damit sie dargestellt werden können?
- Sind insbesondere tabellarische Darstellungen sinnvoll strukturiert? Stehen alle Informationen einer Zeile im realen Zusammenhang? Beispiel: Sind die Zusammenhänge zwischen Abfahrt-/Ankunftszeiten und Abfahrt-/Ankunftsorten klar zu erkennen? Eine tabellarische Sicht sollte nur die wesentlichen Informationen in knapp strukturierter Form enthalten.

Beispiel

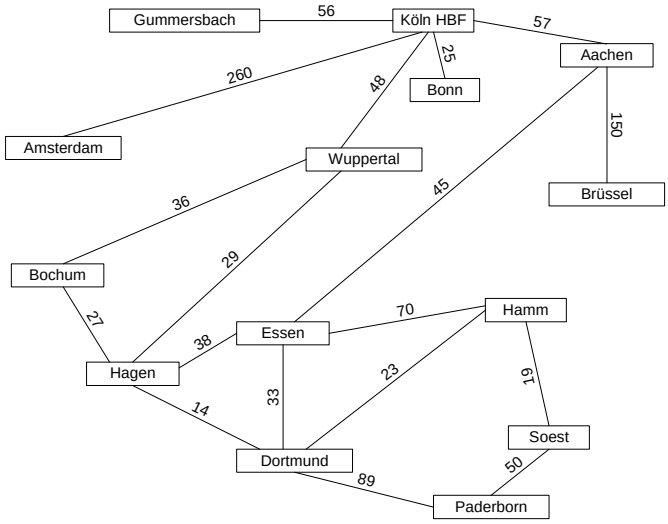
Die Verbindungen zwischen Städten/Bahnhöfen können durch gerichtete gwichtetet Graphen dargestellt werden. Ein Graph G=(N, E) besteht aus einer Menge von Knoten N (=Nodes) und einer Menge von Kanten E (=Edges). Jeder Kante wird eine Zahl als Gewicht zugewiesen, die z. B. für die Fahrtdauer, die Entfernung oder für den Fahrpreis eines Streckenabschnitts steht.

Die Definition eines solchen Graphen erfolgt am einfachsten durch eine Menge von Tupeln, wie das folgende Beispiel zeigt:

"Köln HBF" Bonn 25
"Köln HBF" Aachen 57
"Köln HBF"s Gummersbach 56
Aachen Brüssel 150
Aachen Amsterdam 260
Aachen Essen 45
Essen Dortmund 33
Dortmund Hamm 23
Hamm Soest 19
Soest Paderborn 50
Paderborn Soest 50
Dortmund Bochum 16
Bochum Wuppertal 36
Dortmund Hagen 14
Hagen Wuppertal 29
Wuppertal Köln 48
…

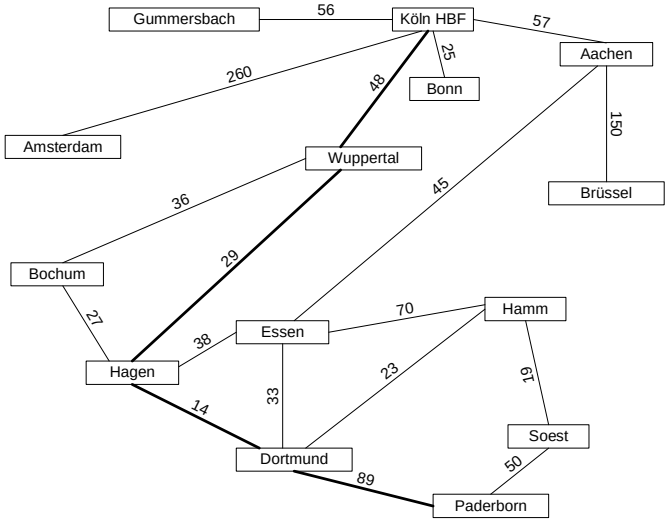
Die Tupel können einfach in einer Datenbanktabelle abgelegt werden. Bei einer Anfrage nach einer Reiseverbindung wird aus den gefundenen Datenbanktupel ein XML-Baum erzeugt. Dieser könnte zum Beispiel wie folgt aussehen:

```
<train>
    <section>
        <from>Köln HBF</from>
        <to>Bonn</to>
        <distance unit="km">25</distance>
        <dauer unit="min">15</dauer>
        <price unit="eur">3,25</price>
    </section>
    <section>
        <from>Köln</from>
        <to>Aachen</to>
        <distance unit="km">57</distance>
        <dauer unit="min">40</dauer>
        <price unit="eur">3,25</price>
    </section>
    <section>
        <from>Aachen</from>
        <to>Brüssel</to>
        <distance unit="km">150</distance>
        <dauer unit="min">125</dauer>
        <price unit="eur">27,50</price>
    </section>
    …
</train>
```



Aus dem XML-Baum muss eine Datenstruktur erzeugt werden, aus welcher der verwendete Algorithmus die Daten liest. Start und das Ziel erhält der Algorithmus (z. B. Dijkstra) als Parameter. Danach wird der kürzeste (schnellste, oder preiswerteste) Weg zwischen zwei Städten/Bahnhöfen berechnet. Als Ausgabe wird ein XML-Baum erzeugt, welcher z.B. folgende Form haben könnte:

```
<dijkstra_result>
    <connection>
        <from>Köln HBF</from>
        <to>Paderborn</to>
        <distance unit="km">180</distance>
        <price unit="eur">45,60</price>
        <duration unit="min">136</duration>
    </connection >
    <route>
        <from>Köln</from>
        <to>Wuppertal</to>
        <from>Wuppertal</from>
        <to>Hagen</to>
        <from>Hagen</from>
        <to>Dortmund</to>
        <from>Dortmund</from>
        <to>Paderborn</to>
    </route>
</dijkstra_result >
```



Eine Klasse *Graph* mit dem Algorithmus von Dijkstra finden Sie auf der Praktikumseite (code.zip und docs.zip)

Bitte überfluten Sie mich nicht mit E-Mails, denn ich schaffe es nicht, alle E-Mails (schnell) zu beantworten.

Bitte erstellen Sie eine ausführliche und exakte Dokumentation! Beachten Sie die Hinweise zum Praktikum!