**Software-Engineering I**

**Programmentwurf  
TINF18B1  
3.+4. Semester (2019/2020)**

***Thema:   
Museumsverwaltung***

DHBW Karlsruhe  
Studiengang Angewandte Informatik

Dr.-Ing. R. Lutz

Institut für Automation und angewandte Informatik (IAI)

Karlsruher Institut für Technologie (KIT)

Bearbeitende:

<Name1> und <Name2>

Inhalt

[1. Einleitung 3](#_Toc38379983)

[2. Lastenheft 3](#_Toc38379984)

[2.1. Zielsetzung 3](#_Toc38379985)

[2.2. Anwendungsbereiche 3](#_Toc38379986)

[2.3. Zielgruppen, Benutzerrollen und Verantwortlichkeiten 3](#_Toc38379987)

[2.4. Zusammenspiel mit anderen Systemen 3](#_Toc38379988)

[2.5. Produktfunktionen 4](#_Toc38379989)

[2.6. Produktdaten 5](#_Toc38379990)

[2.7. Produktleistungen 5](#_Toc38379991)

[2.8. Qualitätsanforderungen 5](#_Toc38379992)

[3. Aufgaben 6](#_Toc38379993)

[3.1. Analyse 6](#_Toc38379994)

[3.2. Sequenzdiagramm und Aktivitätsdiagramm (Analyse oder Entwurf) 6](#_Toc38379995)

[3.3. Entwurf 6](#_Toc38379996)

[3.4. Implementierung 7](#_Toc38379997)

[4. Vereinfachungen für den Programmentwurf 7](#_Toc38379998)

[5. Besonderheiten 8](#_Toc38379999)

# Einleitung

Für unser inzwischen international bekanntes Museum „*Musée Déabevée*“ benötigen wir ein Verwaltungssystem, um alle Daten besser und effizienter erfassen und verwalten zu können.

Unser Museum hat inzwischen zahlreiche Angestellte, die neben der Erfassung und Pflege der Exponate auch für den intensiven Kontakt zu unseren Förderinnen und Förderern zuständig sind.

Bisher vor kurzem war es möglich, mit Hilfe von *Excel* die Verwaltung unserer Daten durchzuführen, was durch die stark steigende Anzahl an Exponaten, Fördererinnen und Förderern und auch Räumen nun nicht mehr auf Dauer realisierbar ist.

# Lastenheft

## Zielsetzung

Ziel des Entwicklungsauftrags soll eine Software für die Verwaltung von Exponaten, von fördernden Personen (im Folgenden zur Vereinfachung als „Fördernde“ bezeichnet), Räumen und Angestellten sein, wozu vor allem die Dokumentation aller zu einem Exponat zugeordneten Daten eine wesentliche Rolle spielt. Alle Daten sollen zentral gespeichert werden, da durch geplante Erweiterungen mehrere Benutzer gleichzeitig auf die Daten und Termine zugreifen werden.

Ein selektiver Import und Export von Daten über lesbare Dateien muss für Backups und zum Datenaustausch möglich sein.

Eine intuitive, leicht bedienbare Benutzeroberfläche setzen wir als selbstverständlich voraus. Es sollen keine besonderen Computerkenntnisse zur Bedienung der Software erforderlich sein.

## Anwendungsbereiche

Die Software soll ausschließlich für die Verwaltung von Exponaten, Fördernden, Räumen und Angestellten und den damit direkt verbundenen Elementen eingesetzt werden. Sie soll ausschließlich innerhalb der Museumsräume eingesetzt werden.

## Zielgruppen, Benutzerrollen und Verantwortlichkeiten

Es gibt folgende Benutzerrollen:

* Benutzer(innen) zur Pflege der Daten über die Exponate. Diese gehören grundsätzlich zum geschulten Fachpersonal.
* Angestellte zur Pflege der Fördernden und Angestellten im System
* Eine hauptverantwortliche Person (Administrator) hat Vollzugriff auf sämtliche Daten, vor allem für deren Import und Export sowie deren Backup.

## Zusammenspiel mit anderen Systemen

Die Daten über die Angestellten (Gehälter bzw. Löhne, Steuern, Kranken- und Rentenversicherung usw.) werden separat durch ein vorhandenes Personalbuch­haltungs­programm verwaltet und müssen hier nicht berücksichtigt werden. Die finanztechnischen Daten werden über unser vorhandenes Finanzsystem erfasst und müssen hier ebenfalls nicht berücksichtigt werden.

Ein sicherer Web-Zugriff auf unser Angebot (Teilinformationen über unsere Exponate) muss erst in einer späteren Erweiterung über eine Web-Seite möglich sein. Jedoch sollen bereits jetzt Schnittstellen dafür definiert werden.

Möglichst alle Daten sollen vom alten in das neue System übertragen werden.

## Produktfunktionen

|  |  |
| --- | --- |
| /LF10/ | Der jeweilige Benutzer muss die Möglichkeit haben, über eine grafische Benutzeroberfläche alle für ihn relevanten Daten einfach und übersichtlich zu verwalten. |
| /LF20/ | Verwaltet werden sollen in erster Linie unsere Exponate. Dabei soll es den berechtigten Angestellten möglich sein, die Exponate zu erfassen, zu ändern, zu löschen und mit vorgegebenen Kriterien nach ihnen zu suchen.  Die Exponate sind aufgeteilt in unterschiedliche Kategorien (Bilder, Skulpturen, Videos, Audios, usw.) und mindestens einem oder mehreren Exponattypen (Epochen usw.) zugeordnet wie z.B. Renaissance, Realismus, Surrealismus, Heimatgeschichte uvm. Diese Exponattypen müssen im zu erstellenden System leicht erweiterbar sein.  Jedem Exponat ist mindestens ein Besitzer zugeordnet. Ein Besitzer kann unser Museum selbst sein oder eine oder mehrere beliebige juristische Personen, die uns das Exponat zu Ausstellungen zur Verfügung stellen.  Jedes Exponat ist charakterisiert durch eine Inventarnummer, dem Erstellungsjahr, dem Einkaufswert, dem aktuellen Schätzwert, dem Leihwert (pro Zeiteinheit), einer Historie, Besitzer uvm.  Zusätzliche sind folgende Datumsangaben anzugeben:   * Erwerb des Exponats durch das Museum * Ausleihe (Eingang im Museum) * Ausleihen an andere Museen und/oder Ausstellungen sowie deren Rückgaben (Ausgang bzw. Eingang) * Verkauf (Details können der Historie hinzugefügt werden) * Anlage und letzte Änderung im System |
| /LF30/ | Als Fördernde gelten alle juristischen Personen, die entweder ausgewählte Exponate finanziell sponsern (Schenkung, Kauf, Ausleihen, Unterhalt (Zuschüsse zu Leihgebühren, zur Pflege usw.)) oder die auf eine andere Art das Museum finanziell unterstützen. Die finanztechnischen Abläufe werden mit unserem Finanzsystem erfasst und verarbeitet, im neuen System soll allerdings auf einfache Weise festgehalten werden, in welcher Form und mit welchen Mitteln ein Fördernder das Museum unterstützt.  Einem Fördernden sind alle ihn betreffenden Exponate zugeordnet und umgekehrt kann auch ein Exponat von mehreren Förderern finanziert werden. |
| /LF40/ | Um den Förderern auf einfache Weise Mails und Informationsmaterialien zukommen zu lassen, sind ihre Kontaktdaten so vollständig wie möglich zu erfassen. |
| /LF50/ | Alle Angestellten müssen verwaltet werden. Jedem Exponateintrag im System müssen die Angestellten zugeordnet sein, die das Exponat anlegen bzw. ändern |
| /LF60/ | Zur einfacheren Eingabe der Daten soll es Auswahllisten für deren Eigenschaften geben, wo immer es möglich ist. Die Auswahllisten sollen auf einfache Weise erweiterbar und für sämtliche Angestellte im System verfügbar sein. |
| /LF70/ | Sämtlichen Elementen sollen mehrere Bilder mit Titel zugeordnet werden können, die zentral auf einem Verzeichnis liegen sollen |

## Produktdaten

|  |  |
| --- | --- |
| /LD10/ | Die Daten sollen in einer zentralen Datenbasis (lesbare Dateien) abgespeichert werden. |

## Produktleistungen

|  |  |
| --- | --- |
| /LL10/ | Die Anzahl der zu verwaltenden Elemente wird auf ca. 10.000 geschätzt. |
| /LL20/ | Um bei HW- und SW-Anschaffungen und -neuerungen flexibel zu bleiben, ist auf Plattformunabhängigkeit besonders zu achten. |

## Qualitätsanforderungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Produktqualität | sehr gut | gut | normal | nicht relevant |
| Funktionalität | X |  |  |  |
| Zuverlässigkeit |  | X |  |  |
| Effizienz |  | X |  |  |
| Benutzbarkeit | X |  |  |  |
| Änderbarkeit |  |  | X |  |
| Übertragbarkeit |  |  | X |  |

# Aufgaben

Einzelne Lastenheftpunkte sind bewusst offengehalten. Denken Sie darüber nach, welche Informationen zusätzlich sinnvoll oder auch notwendig sind. Recherchieren Sie evtl. nach einzelnen Zusammenhängen im Internet.

## Analyse

Für die Analyse sind zu erstellen:

* Analyse des Lastenhefts (Fragen und Antworten).
* Ein Use-Case-Diagramm der gesamten Anwendung incl. Beschreibung.
* Eine Verfeinerung des Use-Case-Diagramms incl. Beschreibung. (nach Absprache)
* Ein Analyse-Klassendiagramm incl. Beschreibung (Untersuchen Sie dabei den Einsatz geeigneter Analysemuster)
* Einfache GUI-Skizzen (Mockups) der wesentlichen GUI-Komponenten (Hauptseite, Tabs, etc.). Die Skizzen können mit einem einfachen Grafikprogramm erstellt werden. Auch sorgfältige Handzeichnungen sind erlaubt. Bitte keine Login-GUI skizzieren!

## Sequenzdiagramm und Aktivitätsdiagramm (Analyse oder Entwurf)

Erstellen Sie ein Sequenzdiagramm und ein Aktivitätsdiagramm (incl. Beschreibung) für folgende Szenarios (ein AD für das eine Szenario, ein SD für das andere Szenario):

* Die Aktionen „Exponat anlegen“ und „Exponat löschen“ durchführen. Ausgehend von einem neuen Exponat werden dessen gesamte Daten erfasst und in das System eingetragen bzw. alle relevanten Daten eines Exponats gelöscht.
* Die Aktionen „Förderer anlegen“ und „Förderer löschen“ durchführen. Dabei soll das Anlegen eines Förderers die Anforderungen aus LF30 erfüllen.

Fassen Sie bei beiden Diagrammen die Eingabe aller primitiven Attribute (Name, Datums­angaben, …) in einer einzigen Aktion zusammen (z.B. „Attribute eintragen“).

Für beide Diagrammarten ist das jeweilige Szenario ausführlich zu entwickeln (idealerweise mit Pseudocode). Es sind sämtliche referenzierten Elemente zu berücksichtigen und es sollen jeweils mehrere beteiligte Personen zugeordnet werden.

In allen Fällen wird eine (noch) leere Datenbasis angenommen. Denken Sie an geeignete Diagrammverfeinerungen. Sie können beim Neuanlegen eines Exponats oder Förderers auf das jeweils andere Diagramm (AD, SD) verweisen. Beachten Sie allerdings, dass es möglich sein muss, entweder ein Exponat oder einen Förderer ohne Referenzen anlegen zu können.

## Entwurf

Abzuliefern sind hier (alle Diagramme und GUIs jeweils mit Beschreibung):

* Entwurfsklassendiagramm (Untersuchen Sie dabei den Einsatz geeigneter Entwurfsmuster)
* GUI-Modellierung:  
  Es ist das Kommunikationsschema eines Teils der während der Analyse skizzierten GUI mit **UML** zu modellieren. Die Anwendung selbst soll dabei nach dem einfachen Model-View-Control-Muster aufgebaut sein. Dazu sind mindestens ein Controller, die erforder­lichen Modellklassen sowie eine unabhängige GUI (View) erforderlich.

Die meisten GUI-Elemente werden über eine einfache kleine Java-Bibliothek zur Verfügung gestellt (*swe-utils*.*jar*), deren GUI-Komponenten in das Klassendia­gramm zu integrieren sind, wenn sie verwendet werden.

Die GUI-Modellierung kann in einem separaten Diagramm mit den relevanten Modellklassen erfolgen, falls das Entwurfsklassendiagramm sonst zu komplex werden würde.

## Implementierung

Es ist eine einfache Java-Applikation zu implementieren, die es ermöglicht, Museumsdaten anzulegen, zu ändern und zu löschen.

Zur Realisierung wird die oben bei der Entwurfsaufgabe erwähnte Java-Bibliothek zur Verfügung gestellt (*swe-utils*.*jar*), die neben mehreren GUI-Komponenten einen *CSVReader,* einen *CSVWriter* sowie mehrere Interfaces bereitstellt (in den Packages *event* und *model*).

Daneben ist eine Mini-Test-Applikation gegeben, die die Funktionsfähigkeit der GUI-Komponenten demonstriert (Start mit *java -jar swe-utils.jar*). Details sind der Java-Dokumentation der Bibliothek zu entnehmen.

Zur leichteren und zukunftssicheren Evaluation Ihres Programmentwurfs soll die Java-Applikation als eine Desktop-Applikation mit CSV-Dateien (alternativ XML oder JSON) als zentrale Datenbasis realisiert werden, die von beliebigen Rechnern aus gestartet wird. Dabei sind mehrere Dateien analog zu Datenbanktabellen zu erzeugen.

Einzelne Aufgaben

* Hauptaufgabe ist die Realisierung einer MVC-Applikation mithilfe des Observer-Patterns entsprechend des vorgegebenen GUI-Entwurfs und der gegebenen Java-Bibliothek.
* Die Erzeugung der Instanzen soll in einer Entity-Factory erfolgen und zur Verwaltung der Instanzen ist ein Entity-Manager zu realisieren (beides siehe Vorlesung).
* Neben der Haupt-GUI ist **ein** beliebiger weiterer TAB aus diesem GUI-Entwurf zu implementieren.
* Es muss eine ausführbare JAR-Datei abgegeben werden, die mit

„java -jar SWE-PE-2020\_Museum\_<name1>\_<name2>.jar OPTIONEN“

gestartet werden kann. Hierfür ist ein BASH-Skript namens *startApp* zu erstellen.

Verwendung von CSV-Dateien:

* Die Daten sollen in CSV-Dateien vorliegen und können mittels den gegebenen Bibliotheksklassen *CSVReader* und *CSVWriter* gelesen bzw. beschrieben werden. Zur Vereinfachung können die Daten jeweils komplett geschrieben werden.
* Abgegeben werden soll ein ZIP-File (oder TAR-File) mit allen Java- und CSV-Dateien (letztere gesammelt in einem eigenen Verzeichnis):

„SWE-PE-2020\_Museum\_<n1>\_<n2>.zip (tar oder tar.z)

* Als OPTIONEN in der Startanweisung soll der Pfad zu den CSV-Dateien sowie zu einer Properties-Datei angegeben werden können:

„java -jar SWE-PE-2020\_Museum\_<n1>\_<n2>.jar **–d <csvpath> –p <propfile>**“

# Vereinfachungen für den Programmentwurf

1. Es muss nicht dafür gesorgt werden, dass auf dieselben Daten bzw. CSV-Dateien nicht gleichzeitig zugegriffen werden kann, d.h. es ist kein *Locking*-Mechanismus erforderlich.
2. Eine Protokollierfunktion und ein Login-Vorgang sind für die Anwendung nicht erforderlich (in der Realität natürlich schon!).

Alle hier aufgeführten, farbig hinterlegten Zeilen bitte nach Erledigung der Aufgaben löschen!

Nun die komplette Aufgabenstellung kopieren und hier einfügen (🡺 als originale Aufgabenstellung und als Template für die Fragen und Antworten 🡺 Überschrift „Analyse“).

Hier kommen nun neben „Analyse“ Ihre eigenen Abschnitte hin, **letzter** Abschnitt ist (mit erhöhter Nummer):

# Besonderheiten

In diesem Abschnitt nennen Sie alle Besonderheiten, die ich als Bewertender beurteilen soll, damit ich Ihre besonderen Ideen und Realisierungen beim Korrigieren nicht übersehe.