PS Software Engineering

PS-SWE-WS2000 21. Dezember 2000

# Archäologisches Managementsystem

**Report**

Academic Supervisor Mag. Dipl.-Ing. Schwaiger Roland

Department of Computer Science
University of Salzburg

Correspondence to:
Universität Salzburg
Institut für Computerwissenschaften und Systemanalyse
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

# Inhaltsverzeichnis

**1 Architektur** **1**
1.1 Schichten Architektur . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Subsysteme . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
1.2.1 Funktionalität und Interface . . . . . . . . . . . . . . . . . . . . . . . 4

**2 Project Plan** **7**
2.1 Kategorisierung . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
2.2 Entwicklungsrisiko . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
2.3 Entwicklungszeit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**Zusammenfassung**

Erstellung eines Software- Packetes, das zur Erfassung und Verwaltung von archäologischen Daten verwendet wird. Die Erfassung, sowie die Bearbeitung und Darstellung der Daten soll webbasierend sein. Weiters wird über das Internet die Suche nach archäologgischen Daten (Funden) durch eine Suchmaschine ermöglicht. Eine bestimmte Auswahl an Informationen kann, durch die Erfasser aufbereitet, über das Internet in einem *virtuellen Museum* (VM) betrachtet werden.
Die Erstellung einer CD- ROM aus einem VM und die Zusammenstellung einer Auswahl der archäologischen Daten zu einem Fund- Katalog soll ebenfalls möglich sein.

# Kapitel 1

# Architektur

In einer generellen Betrachtung haben wir für die Darstellung des Systemes als Architektur den *Layer pattern* gewählt. In dieser Methode wird das System aus verschiedenen Layern aufgebaut. Die einzelnen Bestandteile haben nach innen eine starke Bindung und nach außen kommunizieren sie nur über Interfaces.

## 1.1 Schichten Architektur

Das System ist daher aus folgenden Schichten (Layer) aufgebaut:

**Layer 7:** GUI-Engine

**Layer 6:** Session: Kontrolle und Steuerung der gesamten Session mit Prüfung der Berechtigungen zu den Aktivitäten.

**Layer 5:** Datenrepräsentation: Verwaltung aller, für diese Session benötigten Daten.

**Layer 4:** Filterung und Steuerung des Datenbankzugriffes, eingeteilt in die Teilbereiche **Suchen, Einfügen und Ändern, Löschen**.

**Layer 3:** Datenbankzugriff mit Abfragen in aggregierter (komplexer) Form.

**Layer 2:** Aufspaltung von komplexen Abfragen in atomare Aufrufe und Kapselung des gesamten Datenbanksystems, um die Datenhaltung flexibel zu halten (Loadbalancing, mehrere Datenbanken, ...).

**Layer 1:** Datenbankverwaltung

Eine Darstellung dieser Architektur ist zu sehen in Abbildung 1.1.

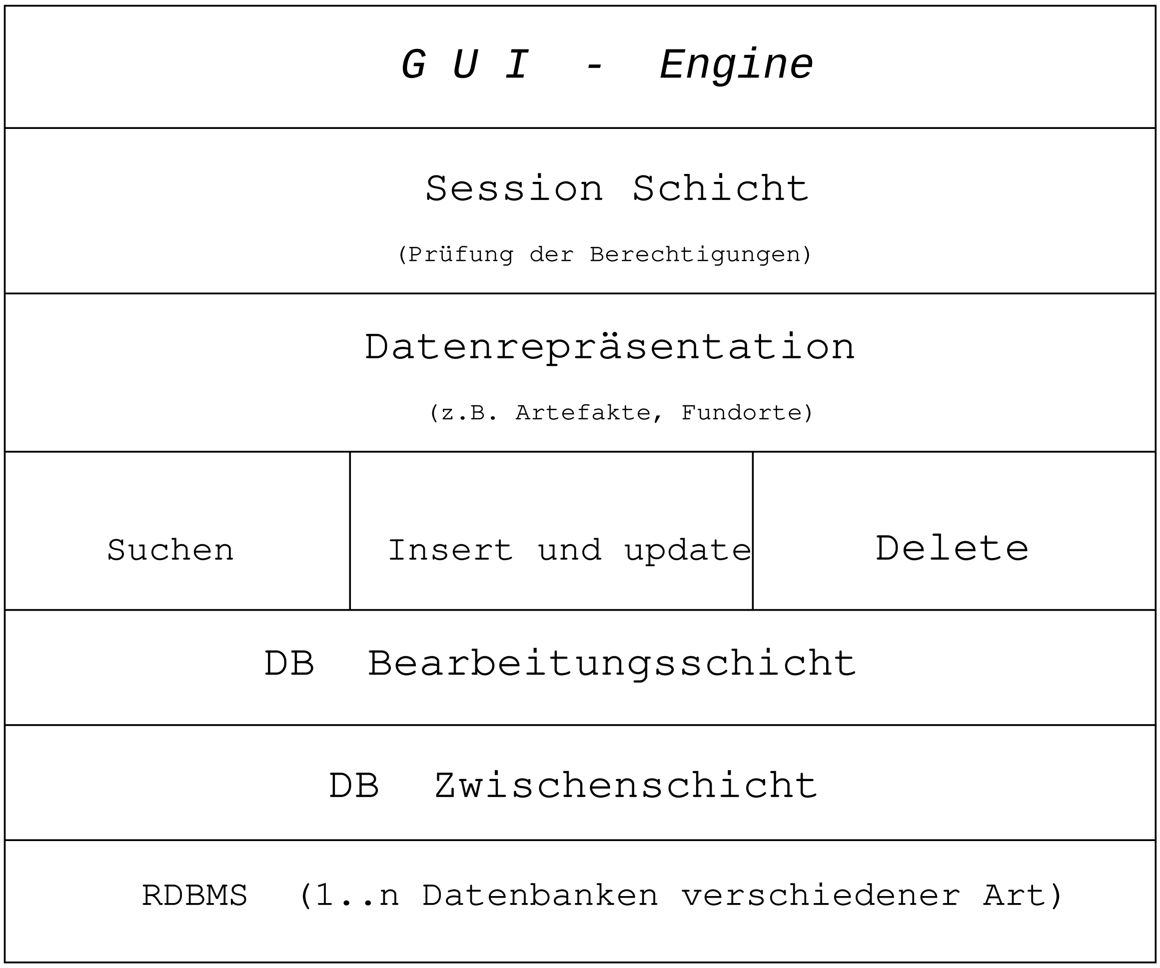


Abbildung 1.1: Architektur-Diagramm

## 1.2 Subsysteme

In Abbildung 1.2 sind die einzelnen Teile der Architektur aufgeteilt in *Subsysteme* dargestellt.

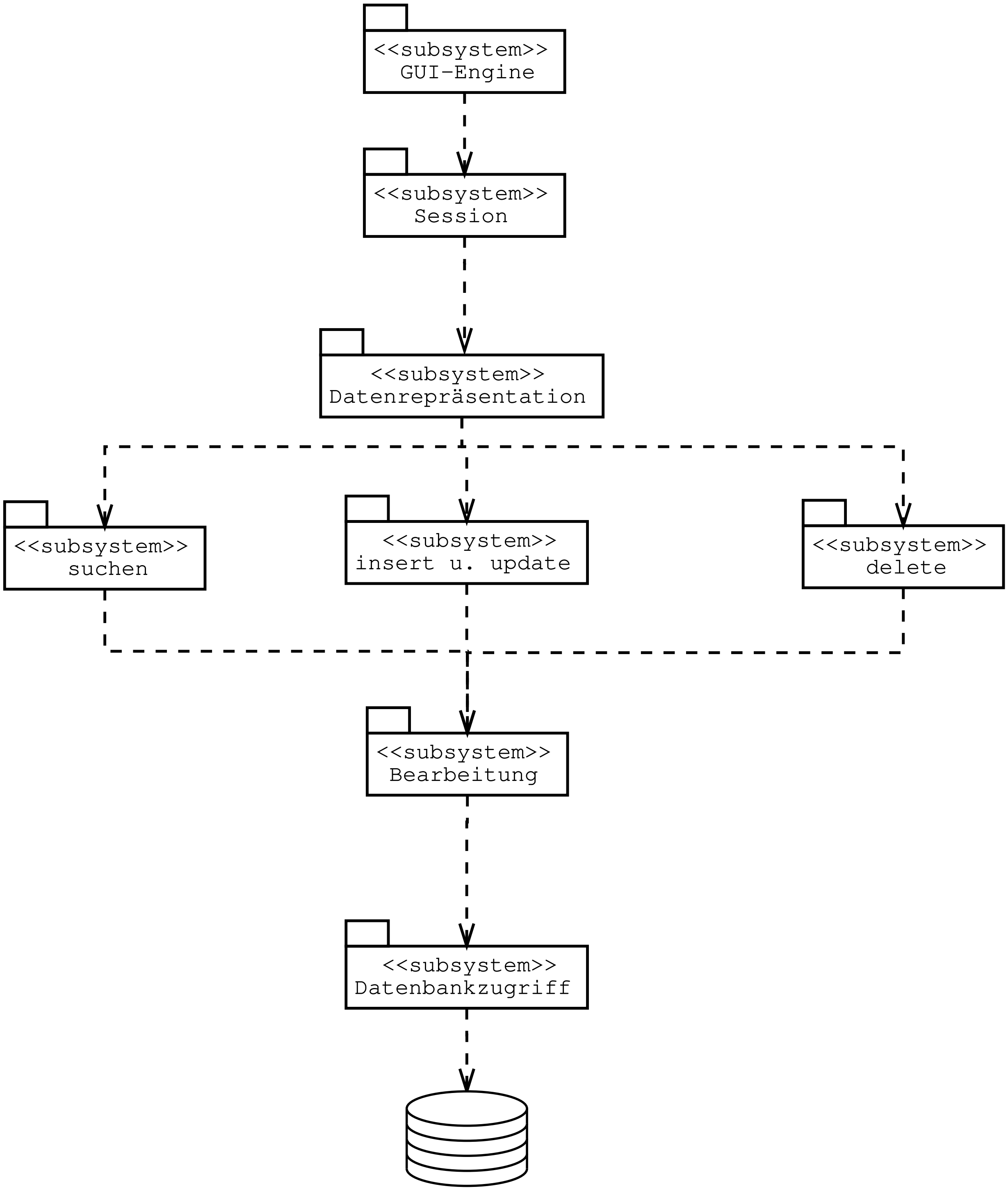


Abbildung 1.2: Diagramm von Subsystemen

Für die bisher beschriebenen Use Cases stellt diese Architektur-Darstellung einen Querschnitt der einzelnen Teilbereiche dar, die von jedem Use Case durchlaufen werden muß. Am Beispiel des Suchens durch den Erfasser:

- Die GUI-Engine erzeugt (nach korrektem Login) eine Eingabeform für den Erfasser.
- Die Session verwaltet die Berechtigungen bezüglich des Erfassers beim Suchen.
- Die Datenrepräsentation enthält alle relevanten Suchbegriffe und Ergebnisdaten der Suche.
- Bei der Suche wird im *Layer 4* die Suche aufgerufen.
- Die Frage wird weitergereicht an die darunterliegenden Layer, die nach entsprechendem Aufwand eine atomare Datenbankabfrage durchführen.
- Nach dem Durchsuchen der Datenbank werden von dort in umgekehrter Reihenfolge die einzelnen Stationen durchlaufen bis die Ergebnisse die GUI-Engine erreicht haben, die dann eine Ergebnisdarstellung aufbaut.

Die Subsysteme enthalten verschiedene Bereiche der Funktionalitäten, welche den anderen Subsystemen mittels Interfaces zur Verfügung gestellt werden.

### 1.2.1 Funktionalität und Interface

1. **GUI-Engine:**

   **Funktionalität:** Generierung einer GUI aus der Funktionalität (Use Cases), der vorhandenen Daten und der zur Verfügung stehenden Rechte.

   **Interface:** GUI-Oberfläche (Html-Fenster am Browser).

2. **Session:**

   **Funktionalität:** Auswertung, Filterung und Weiterleitung der Informationen zwischen GUI und der darunterliegenden Schichten (Daten, Rechte, ..).
   Hier befinden sich die gesamten Use-Case Repräsentationen.
   Nach den Use Cases werden die darunterliegenden Schichten entspechende Aktionen ausführen.
   Bsp: Artefakte suchen:
   Daten von Artefakte werden gespeichert und Suchaufrufe suchen nach Artefakten (nicht Personendaten). Die Überprüfung der Berechtigungen erfolgt ebenfalls hier. D.h. Funktionen der unteren Schichten werden nur dann ausgeführt, wenn auch eine Berechtigung dafür gegeben ist (z.B. Artefakte löschen).

   **Interface:**
   - getData: Ist eine überladene Methode, die je nach Use Case die entsprechenden Daten zur Verfügung stellt.
   - search_for_Data: Mit Suchkriterien als Parameter wird eine Suche veranlaßt.
   - insert_updateData
   - delete_Data

3. **Datenrepräsentation:**

   **Funktionalität:** Verwaltung und Speicherung der gesammelten Daten dieser Sitzung.

   **Interface:**
   - getData: Ist eine überladene Methode, die je nach Use Case die entsprechenden Daten zur Verfügung stellt.
   - search_for_Data: Mit Suchkriterien als Parameter wird eine Suche veranlaßt.
   - insert_updateData
   - delete_Data

4. **Suchen:**

   **Funktionalität:** Aufbereitung der Anfragen für die Suche in der Datenbank in Abhängigkeit zur Art der Suche (Artefakte suchen als Erfasser, Internet-Besucher, ...).
   Die Aufbereitung ist abhängig vom übergebenen Objekt der Suchkriterien.
   Rückgabe der Ergebnisse aus der Suche an die darüberliegende Schicht.

   **Interface:** search_for_Data: Der Parameter ist ein Objekt zur Identifikation der Art der Suche. Rückgegeben wird das Suchergebnis oder eine Fehlermeldung.

5. **Einfügen und Ändern:**

   **Funktionalität:** Ein übergebenes Objekt wird zum Insert oder Update für Aktualisierung an die untere Schicht weitergeleitet. Bestätigung des Erfolgsstatus wird an die obere Schicht zurückgegeben.

   **Interface:** insert_updateData

6. **Löschen:**

   **Funktionalität:** Informationen aus der oberen Schicht werden für die untere Schicht aufbereitet, um bestimmte Informationen aus der Datenbank zu entfernen (z.B. ganzes Artefakt entfernen, Publikation entfernen).

   **Interface:** delete_Data

7. **Bearbeitung:**

   **Funktionalität:** Stellt der oberen Schicht die Funktionen zur Verfügung, die Abfragen komplexerer Datenbankzugriffe ermoglichen und gibt diese an die untere Schicht weiter.
   Die Ergebnisse aus dem Datenbankzugriff werden in entsprechend aggregierter Form an die darüberliegende Schicht weitergegeben (Bsp Suchertgebnisse bei Suche nach Artefakten).

   **Interface:**
   - select: Rückgabe ist das Suchergebnis in aggregierter Form.
   - insert: Rückgabe ist eine Erfolgsmeldung oder Fehlermeldung.
   - update: wie bei insert.
   - delete: wie bei insert.

8. **Datenzugriff:**

**Funktionalität:** Die aggregierten Anweisungen auf die Datenbank werden von der oberen Schicht übernommen und in atomare Datenbankanweisungen aufgeteilt. Hier finden die atomaren Datenbankzugriffe statt. Die Ergebnisse des Datenbankzugriffes (in Rohform) an die obere Schicht zurückgegeben (bei Suche), oder nur die entsprechende Erfolgsrückmeldung (Bestätigung für erfolgreiches Insert, Update oder Delete).

**Interface:**

- select: Rückgabe ist das Suchergebnis in aggregierter Form.
- insert: Rückgabe ist eine Erfolgsmeldung oder Fehlermeldung.
- update: wie bei insert.
- delete: wie bei insert.

# Kapitel 2

# Project Plan

Aufgrund der Einteilung des Systems mittels Layer-Pattern ist ein Projektplan rein auf Basis der Use Cases nicht sinnvoll. Vielmehr ist die Einteilung der Implementierung in Subsysteme in Verbindung mit Use Cases zu betrachten.

## 2.1 Kategorisierung

Eine Kategorisierung der Use Cases bezüglich der Problemstellung ist zwar noch mit dem Kunden genauer zu definieren, jedoch wollen wir hier eine Diskussionsgrundlage anführen. In der nachfolgenden Reihung ist die Priorität höher, je kleiner die Nummer der Kategorie ist (1 ist die höchste).

**Kategorie 1:** Erfassung, Bearbeitung und Entfernung von archäologischen Daten.

**Kategorie 2:** Erfassung, Bearbeitung und Entfernung von personellen Daten.

**Kategorie 3:** Suche nach und Anzeige von gefundenen Daten.

**Kategorie 4:** Erfassung, Bearbeitung und Entfernung von wissenschaftlichen Daten.

**Kategorie 5:** Erzeugung, Bearbeitung und Besuch von VM und Sonderschauen.

**Kategorie 6:** Erzeugung eines Kataloges oder einer CD.

Bei der Kategorisierung muß jedoch auch die Implementierung bezüglich der Subsysteme berücksichtigt werden. So sind für die Implementierung einer Kategorie die Subsysteme der Layer eins bis vier Voraussetzung. Die Prioritäten der Subsysteme gehen aus der Layer-Darstellung der Architektur aus Abbildung 1.1 hervor. Die Aufteilung der Priorität bei den Layern:

1. **Layer 1-4:** Datenbank mit Datenbankmodell, DB Zwischenschicht, DB Bearbeitungsschicht, Suchen, Einfügen und Ändern, Löschen

2. **Layer 5-7:** Datenrepräsentation, Session, GUI-Engine. Hier werden die Use Case spezifischen Teile implementiert (siehe Kategorien).

## 2.2 Entwicklungsrisiko

Das Entwicklungsrisiko ist bei der Aufteilung der Prioritäten für die Subsysteme berücksichtigt. Größtes Risiko liegt in den Layern eins bis vier der Layer-Architektur.

## 2.3 Entwicklungszeit

In der ersten Durchführbarkeitsstudie war die gesamte Mann-Wochen Anzahl bei 135 Stunden. Dieses Ergebnis soll hier nocheinmal betrachtet werden.
Ziel der Einteilung der Implementierungsphase in einzelne gleichgroße Iterationsschritte ist, daß nach jedem Iterationsschritt ein Teil der Implementierung (ein Use Case zum Beispiel) dem Kunden zum Testen übergeben werden kann.
So ist ein regelmäßiges Feedback für beide Seiten möglich. Hierbei sei jedoch zu beachten, daß nach Abschluß der Implementierung der Layer eins bis vier diese bei der Implementierung der Use Cases weiter angepasst werden müssen.

Durch die Einteilung in *Kategorien* und *Subsystemen* ergibt sich eine Einteilung in sieben Iterationsschritte.
Die ersten vier Layer der Layer-Architektur sind ein wesentliches Grundgerüst für die einzelnen Kategorien, deshalb werden hier zwei Iteratiosschritte einberechten.

Mit der Orientierung auf die erste Durchführbarkeitsstudie mit einer Projektgesamtlänge von 135 Mann-Wochen sind hier für die Implementierung eine Dauer von 120 Mann-Wochen eigerechnet.
Ein Iterationsschritt hat demzufolge (acht Iterationen) eine Dauer von 15 Mann-Wochen.
Die Implementierungsreihenfolge nach Priorität ist nachfolgend angeführt:

1. **Subsysteme:** Layer eins bis vier der Layer-Architektur. - **zwei Iterat.-S.: 30 M-W)**
2. **Kategorie: 1** - ein Iterationsschritt: **15 M-W**
3. **Kategorie: 2** - ein Iterationsschritt: **15 M-W**
4. **Kategorie: 3** - ein Iterationsschritt: **15 M-W**
5. **Kategorie: 4** - ein Iterationsschritt: **15 M-W**
6. **Kategorie: 5** - ein Iterationsschritt: **15 M-W**
7. **Kategorie: 6** - ein Iterationsschritt: **15 M-W**

**Implementierungsgesamtdauer: 120 Mann-Wochen**

**Anzahl der Iterationsschritte: 8** (2 Schritte für Layer eins bis vier).