**Projekt: Mosti**

Software Architektur Spezifikation

(Software Architecture Document)

[Dokumentstruktur basiert auf RUP „Software Architecture Document“]

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| *Datum* | *Version* | *Änderung* | *Autor* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Inhalt

[1 Dokumentinformationen 2](#_Toc448394277)

[1.1 Änderungsgeschichte 2](#_Toc448394278)

[1.2 Inhalt 3](#_Toc448394279)

[2 Einführung (Introduction) 4](#_Toc448394280)

[2.1 Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations) 4](#_Toc448394281)

[2.2 Referenzen (References) 4](#_Toc448394282)

[2.3 Übersicht (Overview) 4](#_Toc448394283)

[3 Architektonische Darstellung (Architectural Representation) 4](#_Toc448394284)

[4 Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints) 4](#_Toc448394285)

[5 logische Architektur (Logical View) 4](#_Toc448394286)

[5.1 Übersicht (Overview) 4](#_Toc448394287)

[5.2 Design Pakete (Architecturally Significant Design Packages) 4](#_Toc448394288)

[5.2.1 Package ABC *(z.B. Gui)* 4](#_Toc448394289)

[5.2.1.1 Beschreibung des Package 4](#_Toc448394290)

[5.2.1.2 Diagramme 4](#_Toc448394291)

[5.2.1.3 Schnittstellen 4](#_Toc448394292)

[5.2.1.4 Operationen 4](#_Toc448394293)

[5.2.1.4.1 Interne Operation 1 4](#_Toc448394294)

[5.2.1.4.2 Interne Operation 2 4](#_Toc448394295)

[5.2.2 Package DEF *(z.B. Problem Domain)* 5](#_Toc448394296)

[5.2.2.1 Beschreibung des Package 5](#_Toc448394297)

[5.2.2.2 Diagramme 5](#_Toc448394298)

[5.2.2.3 Schnittstellen 5](#_Toc448394299)

[5.2.2.4 Operationen 5](#_Toc448394300)

[5.2.2.4.1 Interne Operation 1 5](#_Toc448394301)

[5.2.2.4.2 Interne Operation 2 5](#_Toc448394302)

[5.2.3 Package GHI *(z.B. Datenhaltung)* 5](#_Toc448394303)

[5.2.3.1 Beschreibung des Package 5](#_Toc448394304)

[5.2.3.2 Diagramme 5](#_Toc448394305)

[5.2.3.3 Schnittstellen 5](#_Toc448394306)

[5.2.3.4 Operationen 5](#_Toc448394307)

[5.2.3.4.1 Interne Operation 1 5](#_Toc448394308)

[5.2.3.4.2 Interne Operation 2 5](#_Toc448394309)

[6 Physikalische Sicht (Physical View) 5](#_Toc448394310)

[7 Prozesse und Threads (Process View) 5](#_Toc448394311)

[8 Datenspeicherung (Data View) 5](#_Toc448394312)

[9 Größen und Leistung (Size and Performance) 6](#_Toc448394313)

# Einführung (Introduction)

## Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations)

Vgl. separates Glossary-Dokument im Repository, Dateiname: Glossary.docx

## Referenzen (References)

Buch: Craig Larman: UML 2 und Patterns angewendet (2005)

## Übersicht (Overview)

Im folgenden Teil dieses Dokuments gehen wir auf die architektonische Darstellung bei unserer Software ein und beschreiben, welche Ziele und Einschränkungen damit verbunden sind. Außerdem möchten wir insbesondere auf die logische Architektur (logical view) und die Datenspeicherung (data view) eingehen, da diese Views bei unserer Software mehr zum Tragen kommen als andere. Abschließend wird noch beschrieben, welche Größen und Leistungen Mosti umfassen soll.

# Architektonische Darstellung (Architectural Representation)

Ein System lässt sich durch verschiedene architektonische Sichten darstellen. So gibt es beispielsweise Sichten für Logik, Daten, Sicherheit, Implementierung oder auch für Use Cases.

In diesem Dokument möchten wir die Architektur der Mosti-Software mit folgenden Sichten beschreiben:

Logische Sicht (logical view):

Diese Sicht soll die Struktur der Software widerspiegeln. Dies betrifft hier vor allem die Implementierung. So wird beispielsweise beschrieben, wie Packages und Klassen gegliedert bzw. organisiert werden oder auch wie Klassen anhand bestimmter Funktionen interagieren können. Wir werden bei unserer Software das Drei-Schichten-Modell an, welches später genauer erläutert wird.

Daten-Sicht (data view):

Diese Sicht soll einen Überblick darüber geben, wie wir Daten erhalten und verwalten. Bei Mosti machen wir das mithilfe einer Datenbank.

*Anmerkung:*

Eine physikalische Sicht ist bei Mosti derzeit nicht relevant, weil die Software so geplant ist, dass sie von einem Anwender nur auf einem bestimmten Rechner benutzt wird und nicht mit anderen Rechnern oder Systemen kommuniziert.

*<Verbale Beschreibung wichtiger Architekturentscheidungen. Ggf. auch andere Architekturideen kurz beschreiben, die diskutiert und (warum?) verworfen wurden.>*

# Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints)

*<Begründung der Architekturentscheidungen: Welche Anforderungen / Einschränkungen haben zu dieser Architektur geführt, d.h.: warum so und nicht anders, welche Vorteile ergeben sich aus dieser Architektur?>*

# logische Architektur (Logical View)

Bei der Entwicklung von Mosti verfolgen wir das Drei-Schichten-Modell. Dabei unterscheiden wir folgende Schichten:

* GUI-Schicht
* Domain-Logik-Schicht
* Persistenz-Schicht (oder auch Datenerhaltungsschicht)

Dieses Modell erläutern wir im folgenden Punkt (5.1 Überblick) noch genauer.

Bezüglich der Package-Struktur streben wir generell an, ähnliche Klassen in einem Package zu gliedern (hohe Kohäsion) und andererseits, dass wenige Abhängigkeiten zwischen den Packages bestehen (geringe Kopplung).

*<Beschreibung der logischen Struktur des Projekts>*

## Übersicht (Overview)

Das Drei-Schichten-Modell für unsere Software lässt sich wie folgt darstellen:

**GUI**

**Domain-Logik**

**Persistenz**

Schichten-Architektur

Die Schicht bzw. das Package GUI umfasst die grafische Benutzeroberfläche, also wie die Daten dem Benutzer präsentiert werden. Dies entspricht dem View aus der MVC-Architektur.

Die Domain-Logik-Schicht ist in dieser Darstellung die Schnittstelle zwischen der GUI-Schicht und der Persistenz-Schicht. Darin werden die spezifischen Anforderungen bzw. Funktionalitäten umgesetzt. Dies entspricht dem Model aus der MVC-Architektur. Unter Verwendung der MVC-Architektur bildet ein Controller die Schnittstelle zwischen GUI- und Domain-Logik-Schicht.

In der Persistenz-Schicht wird die Datenerhaltung geregelt. (gehört des auch zum Model oder bezieht des Model davon einfach bloß??)

Aus dem Diagramm ist zu entnehmen, dass die Abhängigkeit zwischen den Schichten von oben nach unten verläuft. So kann die GUI-Schicht nur auf die Domain-Logik-Schicht zugreifen und die Domain-Logik-Schicht wiederum nur auf die Persistenz-Schicht. Somit kann die GUI-Schicht zum einen nicht auf die Persistenz-Schicht zugreifen. Zum anderen können Domain-Logik-Schicht und Persistenz-Schicht nicht auf die GUI-Schicht zugreifen und machen somit eine unabhängige Entwicklung möglich.

## Design Pakete (Architecturally Significant Design Packages)

### Package GUI

#### Beschreibung des Package

Das Package GUI umfasst die graphische Benutzeroberfläche und somit u. a. die Präsentation von Daten, die über die Domain-Logik-Schicht zur Verfügung stehen. Bei Mosti arbeitet der Benutzer selbst ausschließlich über die GUI-Schicht, um Daten zu erfassen, abzurufen oder um bestimmte Funktionen auszuführen. Somit ist es auch wichtig, dass in diesem Package auch Benutzereingaben erkannt und an den Controller weitergeleitet werden.

#### Diagramme

<Klassendiagramm>

#### Schnittstellen

Die Schnittstelle zwischen der GUI-Schicht und der Domain-Logik-Schicht bildet aufgrund der Anwendung der MVC-Architektur der Controller.

#### Operationen

##### Interne Operation 1

<Beschreibung der Operation mit Sequenzdiagramm>

##### Interne Operation 2

<Beschreibung der Operation mit Sequenzdiagramm>

### Package Domain-Logik

#### Beschreibung des Package

Das Package Domain-Logik umfasst die Logik, welche hinter den Funktionalitäten der Software liegt. Somit ist darin geregelt, wie bestimmte Daten verarbeitet werden. Bezogen auf Mosti ist beispielsweise die Funktionalität, welche die Kassenfunktion betrifft, von hoher Bedeutung.

#### Diagramme

<Klassendiagramm>

#### Schnittstellen

Die Schnittstelle zwischen der Domain-Logik-Schicht und der Persistenz-Schicht bildet aufgrund der Anwendung der MVC-Architektur der Controller.

#### Operationen

##### Interne Operation 1

<Beschreibung der Operation mit Sequenzdiagramm>

##### Interne Operation 2

<Beschreibung der Operation mit Sequenzdiagramm>

### Package Persistenz

#### Beschreibung des Package

Im Package Persistenz wird die Datenerhaltung geregelt. Da wir zur Speicherung von Daten eine Datenbank benutzen, soll dieses Package ermöglichen, dass eine Verbindung zur Datenbank aufgebaut wird, man auf die Daten zugreifen und Datenänderungen tätigen kann.

#### Diagramme

<Klassendiagramm>

#### Schnittstellen

Die Schnittstelle zwischen unserer Software und der Datenbank wird innerhalb der Persistenz-Packages gewährleistet – nämlich mittels dem Model.

#### Operationen

##### Interne Operation 1

<Beschreibung der Operation mit Sequenzdiagramm>

##### Interne Operation 2

<Beschreibung der Operation mit Sequenzdiagramm>

# Physikalische Sicht (Physical View)

**Haben wir sowas??**

*<Aufteilung in Tiers / Partitions, Zuordnung von Modulen oder Prozessen zu einzelnen Rechnern, wenn notwendig (z.B. bei Client-Server-Architektur.>*

# Prozesse und Threads (Process View)

**Haben wir sowas??**

*<Wenn mehrere Prozesse oder Threads eingesetzt werden, wird hier beschrieben, wie diese ablaufen, miteinander funktionieren, Daten austauschen, sich synchronisieren, etc...>*

# Datenspeicherung (Data View)

Bei Mosti werden die erfassten Daten in einer Datenbank gespeichert.

*<Beschreibung mit Diagramm der Datenspeicherung [Data Model].*

# Größen und Leistung (Size and Performance)

**Wieviele Kunden und so**

*<Einschränkungen der Applikation bezüglich Speicher, Leistung, etc…. (zum Beispiel: Verwaltung unterstützt maximal 20'000 Einträge)>*