# 7.       Iterationenplan (Timeboxes)

[eine Beschreibung der 3 nächsten Realsierungsphasen (Iterationen) in Form von Timeboxen. Verlangt eine vorherige Berwertung von UseCases – das Ranking.

Ranking:

UseCases werden geordnet nach ihrer Bewertung. Die drei höchstbewerteten werden in den drei ersten Timeboxes realisiert. Die Berwertung geschieht nach folgenden Kriterien:

1. Risiko
   * Komplexität des UseCses
   * Anforderungen (inkl. an die Usability): haben/bekommen wir die richtigen.Anforderungen?  wie stabil sind sie? haben wir Ansprechpartner?
   * Team KnowHow: der dazu notwendigen Technologie, und des Domänenverständnisses
   * Technologie: ist die einzusetzendeTechnologie gut definiert, stabil, erprobt, einfach?
   * politisch
2. Architekturrelevanz gibt die Realisierung des UseCases wichtige Einblicke in die SW-Architektur des Systems? Hilft er bei den wichtigsten Entscheidungen zur Architektur?
3. Benutzerrelevanz ist der Benutzer an einer frühen Realiserung dieses UseCases interessiert?
4. Lern-und Entwicklungsrelevanz ist der UseCase geeignet, um mangelndes teaminternes KnowHow bzgl. GUI-Entwurf, Klassenentwurf, Datenbankanbindung etc. aufzubauen?

Tip: Ranking zunächst nach 1-3 , dann 4 miteinbeziehen. Nur 1-3 in diesem Dokument erwähnen.

Da in unserem Zeitplan die Zeiten der drei ersten Timeboxes feststehen, muss für jede Timebox der zu realsierende UseCase (Main Success Scenario/Extensions) und die Architektur festgelegt werden in der er realsiert wird.

die Architektur betrifft Einschränkungen bezüglich:

* GUI
* Domänenmodell
* DB-Anbindung
* Reports, Ausdrucke, Networking, Security, HW-Plattform
* Fehlerbehandlung

und muss eventuell berücksichtigen, dass gewisse Systemteile, die der UseCase voraussetzt, bisher noch fehlen und in irgendeiner Weise zunächst bereitgestellt werden müssen, bis spätere Iterationen diese teile liefern werden.]

## Überblick

[Reihenfolge der Timeboxen, welche UseCses, in welchen Zeiten, mit welchen Abhängigkeiten]

## 1. Timebox

[eine Iterationsbeschreibung, in der ein Systemteil realisiert wird, in einer fest vorgegebenen Zeit]

### Benutzungsfall/fälle (UseCase(s))

Basisabläufe:

* Check-In
  + Reservierung
  + Walk-In
* Check-Out
* Reservierung buchen

### Architektur

* Datenbankmodellierung basierend auf Domänenmodell
* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING

### Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Relationales Datenbankmodell

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

### Abhängigkeiten

* Umzusetzende Use cases sind komplett definiert
* Domänenmodell ist vorhanden
* Datenbankserver ist vorhanden und konfiguriert
* Testumgebung/en vorhanden

## 7.3     2. Timebox

### 7.3.1         Benutzungsfall/fälle (UseCase(s))

Basisabläufe:

### Tagesabschluss

* Rechnung erstellen
* Zwischenrechnung erstellen
* Rechnung legen

### Architektur

* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING

### Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Keine

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

### 

### Abhängigkeiten

* Fertigstellung der ersten Timebox
* Umzusetzende Use cases sind komplett definiert

## 7.4     3. Timebox

### 7.4.1         Benutzungsfall/fälle (UseCase(s))

Basisablauf:

### Belegungsvorschau

Alternativabläufe:

* Check-In
* Check-Out
* Reservierung buchen

### Architektur

* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING

### Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Keine

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

### 

### Abhängigkeiten

* Fertigstellung der zweiten Timebox
* Umzusetzende Use cases sind komplett definiert