**Architekturdokumentation**

<Ihr System>

erstellt von

Praktikumsgruppe swt13w<gruppennummer>

<Ihr Name>

*Template Revision: 6.0 DE (Release Candidate)*

*für das Softwarepraktikum an der TU Dresden*

*angepasstes Template: 21.12.2013 (Birgit Demuth)*

|  |  |
| --- | --- |
| We acknowledge that this document uses material from the arc 42 architecture  template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke. For additional contributors see arc42.de/about/contributors.html |  |

**Änderungsübersicht**

| **Version** | **Datum** | **Bearbeiter** | **Beschreibung** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Basisdokumente**

|  |  |
| --- | --- |
| **Dokument** | **Beschreibung** |
|  |  |
|  |  |

Inhaltsverzeichnis

1. Einführung und Ziele 4

1.1 Aufgabenstellung 4

1.2 Qualitätsziele 4

1.3 Stakeholder 4

2. Randbedingungen 4

2.1 Technische Randbedingungen 4

2.2 Konventionen 5

3. Kontextabgrenzung 5

3.1 Fachlicher Kontext 5

3.2 Technischer- oder Verteilungskontext 5

3.3 Externe Schnittstellen………………………………………………………………………..……………………….5

4. Lösungsstrategie und Entwurfsentscheidungen ………………………………………….5

5. Bausteinsicht 5

6. Laufzeitsicht 6

7. Konzepte 6

7.1 Fachliche Strukturen und Modelle 6

7.2 Persistenz 6

7.4 Benutzungsoberfläche 6

7.5 Ergonomie 6

7.6 Transaktionsbehandlung 6

7.7 Sessionbehandlung 6

7.8 Sicherheit 7

7.9 Kommunikation und Integration mit anderen IT-Systemen 7

7.10 Verteilung 7

7.11 Plausibilisierung und Validierung 7

7.12 Ausnahme-/Fehlerbehandlung 7

7.13 Logging, Protokollierung, Tracing 7

7.14 Konfigurierbarkeit 7

7.15 Internationalisierung 7

7.16 Testbarkeit 7

7.17 Buildmanagement…………………………………………………………………………………………………….8

8. Glossar 8

# Einführung und Ziele

## Aufgabenstellung

## Qualitätsziele

## Stakeholder

*Die folgende Tabelle zeigt Ihre konkreten Stakeholder für das System sowie deren Interessen oder Beteiligung.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rolle | Beschreibung | Ziel / Intention | Kontakt | Bemerkungen |
|  |  |  |  |  |

# Randbedingungen

## Technische Randbedingungen

*Zu den Randbedingungen gehören eingesetzte Technologien (Betriebssysteme, Middleware, Datenbanken, Programmiersprachen, Frameworks u.ä.)*

|  |  |
| --- | --- |
| Hardware-Vorgaben | |
|  | <hier Randbedingungen einfügen> |
|  | <hier Randbedingungen einfügen> |
| Software-Vorgaben | |
|  | <hier Radbedingungen einfügen> |
| Vorgaben des Systembetriebs | |
|  | <hier Randbedingungen einfügen> |
| Programmiervorgaben | |
|  | <hier Randbedingungen einfügen> |

## Konventionen

*Fassen Sie hier alle Konventionen zusammen, die für Ihre Softwarearchitektur relevant sind.*

*Dazu gehören*

* *Programmierrichtlinien*
* *Dokumentationsrichtlinien*
* *Richtlinien für das Versions- und Konfigurationsmanagement*
* *Namenskonventionen*

# Kontextabgrenzung

*Die folgenden Unterkapitel zeigen die Einbettung des Systems in seine Umgebung.*

## Fachlicher Kontext

*Dazu gehört das Kontextdiagramm und die Top-Level-Architektur mit einer kurzen verbalen Beschreibung.*

## Technischer oder Verteilungskontext

*Relevant insbesondere für externe Praktikumsprojekte*

## Externe Schnittstellen

*Hierzu gehört mindestens das(die) Anwendungsfalldiagramm(e) einschließlich deren Beschreibung in tabellarischer Form.*

*In den externen Praktikumsprojekten muss überlegt werden, welche externen Schnittstellen im Detail beschrieben werden müssen.*

# Lösungsstrategie und Entwurfsentscheidungen

*Dieses Kapitel motiviert übergreifend die zentralen Gestaltungskriterien für Ihre Architektur.*

*Fassen Sie die Beweggründe für zentrale Entwurfsentscheidungen zusammen. Motivieren Sie ausgehend von Aufgabenstellung, Qualitätszielen und Randbedingungen, was Sie entschieden haben und warum Sie so entschieden haben. Verweisen Sie – wo nötig - auf weitere Ausführungen in Folgekapiteln.*

# Bausteinsicht

*Beschreiben Sie die Architektur des Systems über zwei Ebenen:*

1. *ein UML-Komponentendiagramm. Berücksichtigen Sie dabei Ihre implementierten Packages als auch die wiederverwendeten Module von Frameworks, Klassenbibliotheken u.ä. (grundlegende Struktur Ihres Systems).*
2. *das (die) Entwurfsklassendiagramm(e), welche(s) die Struktur Ihrer(s) eigens implementierten Module/Package(s) dokumentiert.*

# Laufzeitsicht

# *Wie kommunizieren die Instanzen Ihrer Bausteine zur Laufzeit miteinander (Laufzeitszenarien)? Das ist z.B. zu dokumentieren mit UML-Sequenzdiagrammen. Dieses Kapitel ist insbesondere für die meisten externen Praktikumsprojekte relevant.*Konzepte

*Anmerkung: In der Microsoft-Word-Variante enthält dieses Template im Folgenden Anleitungen und Ausfüllhinweise als „ausgeblendeten Text“. Durch den Befehl „Formate ein-/ausblenden“ können Sie die Anzeige dieser Texte bestimmen.*

*Entscheiden Sie zusammen mit Ihrem Tutor, ob der jeweilige Aspekt für Ihr Praktikumsprojekt relevant ist. Minimale Informationen/Diagramme/Erläuterungen sind jeweils angegeben.*

Inhalt:

Die folgenden Kapitel sind Beispiele für übergreifende Aspekte.

Falls einige der Aspekte für Ihr Projekt nicht wichtig sind oder nicht zutreffen, so halten Sie *diese* Information ebenfalls fest, anstatt das Kapitel zu löschen.

Motivation:

Manche der Aspekte lassen sich nur schwer "zentral" als Baustein in der Architektur unterbringen (z.B. das Thema "Sicherheit". Hier ist der Platz im Template, wo Sie Konzepte zu derartigen Themen geschlossen behandeln können.

Alle Aspekte, die in der Architektur an vielen Stellen Konsequenzen zeigen, beispielsweise ein Domänen-/Fachklassen- oder Business-Modell, haben ebenfalls hier einen guten Platz.

Schließlich kommen manche Strukturen in der Architektur wiederholt vor, beispielsweise ein an mehreren Stellen eingesetztes Pattern. Auch solche Aspekte können Sie hier zentral erläutern.

Form:

Kann vielfältig sein. Teilweise Konzeptpapiere mit beliebiger Gliederung, teilweise auch übergreifende Modelle/Szenarien mit Notationen, die Sie auch in den Architektursichten nutzen.

## Fachliche Strukturen und Modelle

*CRC-Karten als Anhang (wenn erstellt), Sequenzdiagramme der Analyse, Analyseklassendiagramm*

Fachliche Modelle, Domänenmodelle, Business-Modelle – sie alle beschreiben Strukturen der reinen Fachlichkeit, also ohne Bezug zur Implementierungs- oder Lösungstechnologie.

Oftmals tauchen Teile solcher fachlichen Modelle an vielen Stellen in der Architektur, insbesondere der Bausteinsicht, wieder auf.

## Persistenz

*Wie behandeln Sie grundsätzlich die Persistenz Ihrer Objekte in Ihrem Projekt?*

Persistenz (Dauerhaftigkeit, Beständigkeit) bedeutet, Daten aus dem (flüchtigen) Hauptspeicher auf ein beständiges Medium (und wieder zurück) zu bringen.

Einige der Daten, die ein Software-System bearbeitet, müssen dauerhaft auf einem Speichermedium gespeichert oder von solchen Medien gelesen werden:

1. Flüchtige Speichermedien (Hauptspeicher oder Cache) sind technisch nicht für dauerhafte Speicherung ausgelegt. Daten gehen verloren, wenn die entsprechende Hardware ausgeschaltet oder heruntergefahren wird.
2. Die Menge der von kommerziellen Software-Systemen bearbeiteten Daten übersteigt üblicherweise die Kapazität des Hauptspeichers.
3. Auf Festplatten, optischen Speichermedien oder Bändern sind oftmals große Mengen von Unternehmensdaten vorhanden, die eine beträchtliche Investition darstellen.

Persistenz ist ein technisch bedingtes Thema und trägt nichts zur eigentlichen Fachlichkeit eines Systems bei. Dennoch müssen Sie sich als Architekt mit dem Thema auseinander setzen, denn ein erheblicher Teil aller Software-Systeme benötigt einen effizienten Zugriff auf persistent gespeicherte Daten. Hierzu gehören praktisch sämtliche kommerziellen und viele technischen Systeme. Eingebettete Systeme (embedded systems ) gehorchen jedoch oft anderen Regeln hinsichtlich ihrer Datenverwaltung.

## Benutzungsoberfläche

*Entwurf der GUI und der Dialoge*

IT-Systeme, die von (menschlichen) Benutzern interaktiv genutzt werden, benötigen eine Benutzungsoberfläche. Das können sowohl grafische als auch textuelle Oberflächen sein.

## Ergonomie

*Wie schätzen Sie die Ergonomie Ihres Systems ein?*

*Wer hat es getestet (Crosstesting, OMA Test?)*

Ergonomie von IT-Systemen bedeutet die Verbesserung (Optimierung) deren Benutzbarkeit aufgrund objektiver und subjektiver Faktoren. Im wesentlichen zählen zu ergonomischen Faktoren die Benutzungsoberfläche, die Reaktivität (gefühlte Performance) sowie die Verfügbarkeit und Robustheit eines Systems.

## Transaktionsbehandlung

*Welche Transaktionen haben Sie implementiert?*

Transaktionen sind Arbeitsschritte oder Abläufe, die entweder alle gemeinsam oder garnicht durchgeführt werden. Der Begriff stammt aus den Datenbanken - wichtiges Stichwort hier sind ACID-Transaktionen (atomar, consistent, isolated, durable).

## Sessionbehandlung

Eine Session, auch genannt Sitzung, bezeichnet eine stehende Verbindung eines Clients mit einem Server. Den Zustand dieser Sitzung gilt es zu erhalten, was insbesondere bei der Nutzung zustandsloser Protokolle (etwa HTTP) wichtige Bedeutung hat. Sessionbehandlung stellt für Intra-  und Internetsysteme eine kritische Herausforderung dar und beeinflusst häufig die Performance eines Systems.

## Sicherheit

*Haben Sie sich mit Sicherheitsaspekten Ihres Systems auseinandergesetzt und wenn ja, welche Sicherheitsvorkehrungen haben Sie realisiert? Diese Frage ist insbesondere bei den externen Praktikumsprojekten relevant.*

Die Sicherheit von IT-Systemen befasst sich mit Mechanismen zur Gewährleistung von Datensicherheit und Datenschutz sowie Verhinderung von Datenmissbrauch.

Typische Fragestellungen sind:

1. Wie können Daten auf dem Transport (beispielsweise über offene Netze wie das Internet) vor Missbrauch geschützt werden?
2. Wie können Kommunikationspartner sich gegenseitig vertrauen?
3. Wie können sich Kommunikationspartner eindeutig erkennen und vor falschen Kommunikationspartner schützen?
4. Wie können Kommunikationspartner die Herkunft von Daten für sich beanspruchen (oder die Echtheit von Daten bestätigen)?

Das Thema IT-Sicherheit hat häufig Berührung zu juristischen Aspekten, teilweise sogar zu internationalem Recht.

## Kommunikation und Integration mit anderen IT-Systemen

*Das ist ggfs. in externen Praktikumsprojekten relevant.*

Kommunikation: Übertragung von Daten zwischen System-Komponenten. Bezieht sich auf Kommunikation innerhalb eines Prozesses oder Adressraumes, zwischen unterschiedlichen Prozessen oder auch zwischen unterschiedlichen Rechnersystemen.

Integration: Einbindung bestehender Systeme (in einen neuen Kontext). Auch bekannt als: (Legacy) Wrapper, Gateway, Enterprise Application Integration (EAI).

## Verteilung

*Das ist ggfs. in externen Praktikumsprojekten relevant.*

Verteilung: Entwurf von Software-Systemen, deren Bestandteile auf unterschiedlichen und eventuell physikalisch getrennten Rechnersystemen ablaufen.

Zur Verteilung gehören Dinge wie der Aufruf entfernter Methoden (remote procedure call, RPC), die Übertragung von Daten oder Dokumenten an verteilte Kommunikationspartner, die Wahl passender Interaktionsstile oder Nachrichtenaustauschmuster (etwa: synchron / asynchron, publish- subsribe, peer-to- peer).

## Plausibilisierung und Validierung

*Welche Benutzereingaben validieren Sie?*

Wo und wie plausibilisieren und validieren Sie (Eingabe-)daten, etwa Benutzereingaben?

## Ausnahme-/Fehlerbehandlung

*Wie werden Programmfehler und Ausnahmen systematisch und konsistent behandelt?*

Wie werden Programmfehler und Ausnahmen systematisch und konsistent behandelt?

Wie kann das System nach einem Fehler wieder in einen konsistenten Zustand gelangen? Geschieht dies automatisch oder ist manueller Eingriff erforderlich?

Dieser Aspekt hat mit Logging, Protokollierung und Tracing zu tun.

Welche Art Ausnahmen und Fehler behandelt ihr System? Welche Art Ausnahmen werden an welche Außenschnittstelle weitergeleitet und welche Ausnahmen behandelt das System komplett intern?

Wie nutzen Sie die Exception-Handling Mechanismen ihrer Programmiersprache? Verwenden Sie checked- oder unchecked-Exceptions?

## Logging, Protokollierung, Tracing

*Haben Sie Maßnahmen getroffen, über den Status des Programms zur Laufzeit Auskunft geben zu können und wenn ja, welche?*

Es gibt zwei Ausprägungen der Protokollierung, das *Logging* und das *Tracing* . Bei beiden werden Funktions- oder Methodenaufrufe in das Programm aufgenommen, die zur Laufzeit über den Status des Programms Auskunft geben.

In der Praxis gibt es zwischen Logging und Tracing allerdings sehr wohl Unterschiede:

1. Logging kann fachliche oder technische Protokollierung sein, oder eine beliebige Kombination von beidem.
2. Fachliche Protokolle werden gewöhnlich anwenderspezifisch aufbereitet und übersetzt. Sie dienen Endbenutzern, Administratoren oder Betreibern von Softwaresystemen und liefern Informationen über die vom Programm abgewickelten Geschäftsprozesse.
3. Technische Protokolle sind Informationen für Betreiber oder Entwickler. Sie dienen der Fehlersuche sowie der Systemoptimierung.
4. Tracing soll Debugging -Information für Entwickler oder Supportmitarbeiter liefern. Es dient primär zur Fehlersuche und -analyse.

## Konfigurierbarkeit

*Kann Ihr System konfiguriert werden und wenn ja, zu welchem Zeitpunkt und für welche Features?*

Die Flexibilität von IT-Systemem wird unter anderem durch ihre Konfigurierbarkeit beeinflusst, die Möglichkeit, manche Entscheidungen hinsichtlich der Systemnutzung erst spät zu treffen. Konfigurierbarkeit kann zu folgenden Zeitpunkten erfolgen:

1. Während der Programmierung: Dabei werden beispielsweise Server-, Datei- oder Verzeichnisnamen direkt ("hart") in den Programmcode aufgenommen.
2. Während des Deployments oder der Installation: Hier werden Konfigurationsinformationen für eine bestimmte Installation angegeben, etwa der Installationspfad.
3. Beim Systemstart: Hier werden Informationen vor oder beim Programmstart dynamisch gelesen.
4. Während des Programmablaufs: Konfigurationsinformation wird zur Programmlaufzeit erfragt oder gelesen.

## Internationalisierung

*Unterstützt Ihr System Internationalsierung und wenn ja, wie?*

Unterstützung für den Einsatz von Systemen in unterschiedlichen Ländern, Anpassung der Systeme an länderspezifische Merkmale. Bei der Internationalisierung (aufgrund der 18 Buchstaben zwischen I und n des englischen Internationalisation auch i18n genannt) geht es neben der Übersetzung von Aus- oder EIngabetexten auch um verwendete Zeichensätze, Orientierung von Schriften am Bildschirm und andere (äußerliche) Aspekte.

## Testbarkeit

*Welche automatisierten Tests haben Sie implementiert?*

Unterstützung für einfache (und möglichst automatische) Tests. Diese Eigenschaft bildet die Grundlage für das wichtige Erfolgsmuster "Continous Integration". In Projekten sollte mindestens täglich der gesamte Stand der Entwicklung gebaut und (automatisch) getestet werden - daher spielt Testbarkeit eine wichtige Rolle. Wichtige Stichworte hierzu sind Unit- Tests und Mock-Objekte.

## Buildmanagement

*Wie wird das gesamte System aus Sourcecode Bausteinen gebaut?*

*Welche Repositories (Versionsverwaltungssysteme) enthalten welchen Sourcecode, wo liegen Konfigurationsdateien, Testdaten und/oder Build-Skripte (make, ant, maven …)?*

Wie wird das gesamte System aus Sourcecode Bausteinen gebaut? Welche Repositories (Versionsverwaltungssysteme) enthalten welchen Sourcecode, wo liegen Konfigurationsdateien, Testdaten und/oder Build-Skripte (make, ant, maven, gradle oder Ähnliche)?

# Glossar

Inhalt

Die wichtigsten Begriffe der Software-Architektur in alphabetischer Reihenfolge

Motivation

Die Notwendigkeit für ein Glossar sollte nicht erläutert werden müssen. Oder haben Sie es in Ihren Projekten noch nie vermisst?

Form

einfache zweispaltige Tabelle mit <Begriff> und <Definition>