**Architekturdokumentation**

MedDevMM

erstellt von

Lehner Roland, Kienböck Daniel, Stöckl Bernhard, Grill Florian, Murrent Mario

*Template Revision: 6.0 DE (Release Candidate)  
19. März 2012*

|  |  |
| --- | --- |
| 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** |
| --- | --- | --- | --- |
| 0.1 | 27.10.2014 | Mario Murrent | Basisversion |
| 0.2 | 02.11.2014 | Roland Lehner | Bearbeitung Punkt 8.2. Broker und Pipes und Filters Muster hinzugefügt |
| 0.3 | 12.11.2014 | Daniel Kienböck | Patternentscheidungen und Beschreibung |

**Basisdokumente**

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

Inhaltsverzeichnis

1. Einführung und Ziele 5

1.1 Aufgabenstellung 5

1.2 Qualitätsziele 5

1.3 Stakeholder 5

2. Randbedingungen 7

2.1 Technische Randbedingungen 7

2.2 Organisatorische Randbedingungen 8

2.3 Konventionen 9

3. Kontextabgrenzung 9

3.1 Fachlicher Kontext 9

3.2 Technischer- oder Verteilungskontext 9

4. Lösungsstrategie 9

5. Bausteinsicht 10

5.1 Ebene 1 10

5.2 Ebene 2 10

5.3 Ebene 3 14

6. Laufzeitsicht 14

6.1 Laufzeitszenario 1 14

6.2 Laufzeitszenario 2 14

6.3 ... 14

6.4 Laufzeitszenario n 14

7. Verteilungssicht 14

7.1 Infrastruktur Ebene 1 14

7.2 Infrastruktur Ebene 2 15

8. Konzepte 15

8.1 Fachliche Strukturen und Modelle 15

8.2 Typische Muster und Strukturen 15

8.3 Persistenz 15

8.4 Benutzungsoberfläche 16

8.5 Ergonomie 16

8.6 Ablaufsteuerung 16

8.7 Transaktionsbehandlung 16

8.8 Sessionbehandlung 16

8.9 Sicherheit 16

8.10 Kommunikation und Integration mit anderen IT-Systemen 16

8.11 Verteilung 16

8.12 Plausibilisierung und Validierung 16

8.13 Ausnahme-/Fehlerbehandlung 16

8.14 Management des Systems & Administrierbarkeit 16

8.15 Logging, Protokollierung, Tracing 16

8.16 Geschäftsregeln 16

8.17 Konfigurierbarkeit 17

8.18 Parallelisierung und Threading 17

8.19 Internationalisierung 17

8.20 Migration 17

8.21 Testbarkeit 17

8.22 Skalierung, Clustering 17

8.23 Hochverfügbarkeit 17

9. Entwurfsentscheidungen 17

9.1 Entwurfsentscheidung 1 17

9.2 Entwurfsentscheidung n 17

10. Qualitätsszenarien 17

10.1 Qualitätsbaum 18

10.2 Bewertungsszenarien 18

11. Risiken 18

12. Glossar 18

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

# Einführung und Ziele

… Krankenhaus …

viele Datenquellen (Integration)

alles mit Info machen können (auch mobile)

für Ärzte und Patienten

security! offen (erweiterung/OS), flexible; kein großer IT-Aufwand in Klinik

Aktuelle und künftige Standards nutzen

## Aufgabenstellung

Patienten – Record aus verschiedenen Quellen je nach Anfrage vollständig aus verschiedenen Quellen zusammenbauen können

## Qualitätsziele

Hochverfügbar

Unabhängig von Plattform

## Stakeholder

Patienten

Mitarbeiter des KH

# Randbedingungen

## Technische Randbedingungen

Plattformunabhängigkeit -> Web

## Organisatorische Randbedingungen

 Wenig Ressourcen für Maintenance des Service

## Konventionen

# Kontextabgrenzung

## Fachlicher Kontext

## Technischer- oder Verteilungskontext

## Externe Schnittstellen

Zu Datenquellen, variabel integrierbar, wenn zupassende bridge gebaut und im Broker registriert

Aktuell:

- Datenformat 1: medizinische Befunde (Textfiles, Datenquelle Radiologie)

- Datenformat 2: Bilder (DICOM, DQ: PACS-App)

- streaming!

# Lösungsstrategie

Webpage

# Bausteinsicht

**Middleware:**

* **Schnittstellenmodul (gelb)**
* **Anfragenmodul (grün)**

Anfrage

Schnittstelle

Kern

* **Kern (blau)**

Middleware

Client 1

Client 2

Client 3

Client 4

Client 5

Client 6

…

Client X

Datenquelle1

Datenquelle2

Datenquelle3

Datenquelle4

Datenquelle5

Datenquelle6

…

DatenquelleX

Cmd-Pattern

Master/Slave  
(bridged)

Broker

Pipes / Filters

F/R

Anfrage immer gleich:

* Pipes and Filters ändert die Anfrage sodass die Anfrage für das Endgerät optimiert werden kann (Mobile / Tablet / Browsser) + eigener Filter für Authentifizierung
* Broker gibt über „Forward Receiver“ (IPC) dem Datenquellen-Master bekannt „wo was wie“ … Master instanziiert einen gebridgeten Slave und speichert sich über Command-Pattern

# Laufzeitsicht

# Verteilungssicht

# Konzepte

## Fachliche Strukturen und Modelle

## Typische Muster und Strukturen

### Nicht funktionale Muster

### Broker

Der Broker ist dafür verantwortlich die Anfrage an das bestehende Service weiterzuleiten (mapping von Request auf Data-Source). Der Broker nimmt die bereits angepasste Anfrage (siehe pipes and filters) entgegen und fragt die Datenquellen an. Sollten neue Datenquellen hinzugefügt werden, muss der Broker um die Datenquellen upgedatet werden.

### Master-Slave

Master

+ result

+launch()  
*+makeSlave() : Slave*

*Slave*

+ result

*+ run()*

ConcreteSlave

+ run()

ConcreteMaster

+ makeSlave() : Slave

Für die Anfrage der Daten aus verschiedenen Schnittstellen wir prinzipiell im Schnittstellen – Modul ein Objekt als Schnittstellen-Master (Singleton) instanziiert und in Folge vom Master pro Anfrage ein (paralleler) Slave erstellt (siehe Bridge-Pattern zu Implementierungsdetails des Slave).

### Command-Prozessor

### Funktionale Muster

### Forward Receiver

### Pipes and Filters

Das Pipes and Filters Architekturmuster, kommt bei der Anfrage der Clients zum Einsatz. Dieser Prozess wird in der Middleware durchgeführt. Es werden somit dem Endgerät die optimierten Daten zur Verfügung gestellt, die es benötigt. Zum Beispiel soll ein mobiles Endgerät keine großen Bilderdaten erhalten, um den Datenkonsum klein zu halten. Darüber hinaus kann hier auch ein Authentifizierungsfilter implementiert werden.

Bsp.: = Pipe

Endgerättypenbestimmung

Authentifizierung

Bildfilter

### Bridge

Abstraction

- impl : Implementor

+ function()

*Implementor*

+ implementation()

ConcreteImplementor

+ implementation()

RefinedAbstraction

+ function()

Da viele verschiedene Schnittstellen (und dessen libraries) unter einen Hut gebracht werden müssen, wurde das Bridge-Pattern gewählt, um hier ein gemeinsames (triviales) Interface für jede Schnittstelle verwenden zu können. Jede neue Schnittstelle zu einem Fremdsystem kann somit über die „Abstraction“ (siehe Bild; interface zur Verwendung) aufgerufen werden. Die RedifenedAbstraction (siehe Bild; Code, der den Zugriff der Abstraktion auf die konkrete Library mapped) muss beim Integrationsprozess ausdefiniert werden.

**Anwendungsfall Schnittstellen**

* Medizinische Befunde mit Bildern werden beim Holen der Daten konsolidiert und werden in einem standardisierten JSON-Format ausgegeben bzw. Bilder im Portable Network Graphics (png)-Format mit einem eindeutig vergebenen Namen hinzugefügt (UUID) mit Referenz im JSON-File.
* Mit demselben JSON/PNG Format werden DICOM Files gelesen und genauso konsolidiert gespeichert.

Über das Pattern ist es möglich dem implementierenden Team nicht ein Interface sondern nur ein Requirement (also abstrakte Anforderung bspw. von einem reinen, nicht technischen Projekt-Manager) vorzugeben (bzw. Software zuzukaufen, wobei man dem Hersteller meist nur schwer eine Schnittstelle vorgeben kann). Das Requirement wird dann implementiert, getestet und über die RedifinedAbstraction in das Projekt von einem - unter Umständen anderen - Team integriert.

## Persistenz

## Benutzungsoberfläche

Browser, weil Web

## Ergonomie

## Ablaufsteuerung

## Transaktionsbehandlung

## Sessionbehandlung

## Sicherheit

Im pipe and filter

## Kommunikation und Integration mit anderen IT-Systemen

Bei bridge pattern wichtig

## Verteilung

## Plausibilisierung und Validierung

## Ausnahme-/Fehlerbehandlung

## Management des Systems & Administrierbarkeit

## Logging, Protokollierung, Tracing

## Geschäftsregeln

## Konfigurierbarkeit

## Parallelisierung und Threading

## Internationalisierung

## Migration

## Testbarkeit

## Skalierung, Clustering

## Hochverfügbarkeit

## Codegenerierung

## Buildmanagement

# Entwurfsentscheidungen

# Qualitätsszenarien

## Qualitätsbaum

## Bewertungsszenarien

# Risiken

# Glossar