Freie Universität Berlin

Lutz Prechelt

Softwaretechnik, SoSe21

Übung 07

TutorIn: Samuel Domiks
Tutorium 02
Materialien: Latex, Skript

Jonny Lam & Thore Brehmer

30. Mai 2021

---

1 Recherche Softwarearchitektur

(a) ***Architekturebenen:***

- *Welche gibt es, wodurch zeichnen sie sich jeweils aus.*
  - **Organisationsebene:**
    - Auf dieser Architektur-Ebene werden Organisationen (z. B. Unternehmen oder Institutionen), deren **Bebauungspläne, Geschäftsprozesse und IT-Landschaften** sowie deren Interaktionen mit anderen **Organisationen** betrachtet.(Seite 80, Absatz 1)
    - Des Weiteren liegen im Kontext der Organisationsebene Anforderungen an eine Organisation sowie organisationsweit zu verwendende **IT-Standards und -Richtlinien**.(80,2)
    - Insbesondere bei folgenden **Problemfeldern** wird sich ein Architekt auch auf der Organisationsebene bewegen müssen: (80,3)
      - IT-gestützte Umsetzung organisationsübergreifender Prozesse (z. B. Supply Chain Management).
      - Enterprise-Architektur.
      - Enterprise Application Integration (EAI)
  - **Systemebene:**
    - Auf der Systemebene werden die **IT-Systeme von Organisationen** betrachtet. Der innere Aufbau der Systeme spielt hier nur in Bezug auf deren Subsysteme eine Rolle.(81,1)
    - Die **wesentlichen Elemente** auf der Systemebene sind: (81,1)
      - Anforderungen an die Systeme
      - Systemkontexte der Systeme
      - Subsysteme der Systeme
  - **Bausteinebene:**

    - Hier betrachtet man die Bausteine der Systeme.(73,2)
    - Auf der Bausteinebene im Bereich Makro-Architektur wird die **innere Struktur** der einzelnen Subsysteme betrachtet.(82,1)
    - **Wesentliche Aspekte**, mit denen man sich auf der Bausteinebene beschäftigt, sind die Verantwortlichkeiten der Systembausteine, deren Schnittstellen sowie deren Interaktionen untereinander.(82,1)

- *Wozu dient ihre Unterscheidung?*
  - Architektur-Probleme/-Aspekte werden passenden Ebenen zugeordnet und sind damit einfacher und einheitlicher zu handhaben [Brown et al. 1998].(74,2)
  - Unterschiedliche Architektur-Probleme/-Aspekte werden nicht vermischt, sondern getrennt mit den jeweils probaten Mitteln behandelt.(74,2)
  - Einflüsse auf eine Architektur liegen explizit vor und können deshalb besser verstanden und berücksichtigt werden.(74,2)
- *Was verstehen die Autoren unter einem „Ebenenwechsel“?*
  - **Ebenenwechsel:** Bei sehr großen Systemen kann es sich bei den Systembausteinen, die sich aus der Dekomposition eines Subsystems ergeben haben, wiederum um Subsysteme anstelle von Software-Bausteinen handeln. In diesem Fall findet ein Ebenenwechsel von der Baustein- zurück auf die Systemebene statt. Dort findet die weitere Behandlung des Systembausteins als System ausgehend vom Systemkontext usw. statt. Dadurch erhält das Architektur-Ebenen-Modell an dieser Stelle einen rekursiven Charakter.(82,2)
- *Worin besteht der Unterschied zwischen Makroarchitektur und Mikroarchitektur?*
  - **Makro-Architektur:** (Software-Architektur, Grob-Entwurf) Makro-Architektur umfasst das Spektrum der Architektur-Ebenen, denen architektonisch relevante Elemente zugeordnet sind (Organisations- und Systemebene sowie den Teilbereich der Bausteinebene, auf dem sich tragende Systembausteine befinden). Diese befasst sich mit Aspekten wie Anforderungen, Entscheidungen und Strukturen auf einem hohen Abstraktionsniveau. Die Organisations- und Systemebene sowie der Teil der Bausteinebene, dem die tragenden Bausteine eines Systems zugeordnet sind, gehören zum Bereich der Makro-Architektur.(78,4)
  - **Mikro-Architektur:** (Detail- oder Fein-Entwurf) Mikro-Architektur dagegen befasst sich mit Aspekten auf einem niedrigen Abstraktionsniveau. Dabei handelt es sich dann um Detail-Entwurf (Architektur im „Kleinen“) mit großer Nähe zum Quelltext ohne fundamentalen Einfluss auf eine Architektur. Zum Bereich Mikro-Architektur gehört der Teilbereich der Bausteinebene, auf dem sich die nichttragenden Systembausteine befinden.(78,5)
- *Durch welches entscheidende Merkmal wird der Übergang gekennzeichnet?*
  - Die Grenze zwischen Makro- und Mikro-Architektur kann nicht immer klar gezogen werden. Diese ist auch abhängig von der Sicht der jeweiligen Interessenvertreter und deshalb ein fließender Übergang.(78,3)

(b) ***Architektursichten:***

- *Welche gibt es und wodurch zeichnen sie sich jeweils aus?*
  - Es ist in Anlehnung an [Rozanski und Woods 2005], [Bredemeyer und Malan 2004] und [Larman 2002] zu empfehlen, zumindest folgende **grundlegende Architektur-Sichten** immer vorzusehen: (89,5)
    1. **Konzeptionelle Sicht (Geschäftssicht):** Diese Architektur-Sicht beschreibt die Systembausteine und ihre Beziehungen untereinander, ohne auf Details wie z. B. Schnittstellen einzugehen. Sie ist dazu geeignet, eine Architektur nicht-technischen Interessenvertretern zu vermitteln.(90,1)

2. **Logische Sicht:** Diese Architektur-Sicht beschreibt die Systembausteine und ihre Beziehungen untereinander im Detail. Dabei werden die Systembausteine und ihre Beziehungen respektive die Kommunikationsmechanismen genau spezifiziert. Dies geschieht im Hinblick auf die technische Realisierung. Damit richtet sich diese Architektur-Sicht an technische Interessenvertreter.(90,2)
3. **Ausführungssicht (Verteilungssicht):** Diese Architektur-Sicht beschreibt im Detail die physikalische Verteilung der Systembausteine zur Laufzeit. Sie richtet sich ebenfalls an technische Interessenvertreter.(90,3)

– Als **grundlegende Architektur-Sichten** werden die Anforderungssicht, die logische Sicht, die Datensicht, die Umsetzungssicht, die Prozesssicht und die Verteilungssicht unterschieden. (101,5)

1. **Anforderungssicht:** beschreibt die Architektur-Anforderungen.(91,2)
2. **Logische Sicht:** beschreibt die Dokumentation des Architektur-Entwurfs.(91,3)
3. **Datensicht:** beschreibt die Aspekten bezüglich Speicherung, Manipulation,Verwaltung und Verteilung von Daten.(92,1)
4. **Umsetzungssicht:** beschreibt die Umsetzungsstruktur und der Umsetzungsinfrastruktur.(92,2)
5. **Prozesssicht:** beschreibt die Steuerung und Koordination nebenläufiger Bausteine.(92,3)
6. **Verteilungssicht:** beschreibt die physikalische Verteilung von Software-Bausteinen..(93,2)

- *Wozu dient ihre Unterscheidung?*
  – Alle Aspekte komplexer Systeme (Menschen, Bauwerke, IT-Systeme etc.) zu jedem Zeitpunkt komplett zu erfassen, ist zumindest für die menschliche Wahrnehmung nicht möglich. Dies zu versuchen, wäre darüber hinaus auch nicht zielführend, weil nicht zu jedem Zeitpunkt gleichzeitig alle Aspekte eines Systems relevant sind. Daher ist es sinnvoll, nur bestimmte, für den Moment interessante Aspekte eines Systems betrachten zu können. Für IT-Systeme existiert zu diesem Zweck das Konzept der Architektur-Sichten.(83,1)
  – Architektur-Sichtenmodellen umfassen alle relevanten Architektur-Sichten und ermöglichen es damit, Architektur überhaupt erst greifbar bzw. sichtbar zu machen. Die Komplexität der Architektur eines Systems kann erst durch Architektur-Sichtenmodelle bewältigt werden.(89,1)
  – Mithilfe eines Architektur-Sichtenmodells lässt sich nachprüfen, ob eine Architektur wie gewünscht alle relevanten Aspekte eines Systems abdeckt. Wird eine Architektur gleich zu Beginn auf Basis eines Architektur-Sichtenmodells erstellt, verringert sich die Wahrscheinlichkeit, dass wichtige architekturrelevante Punkte vergessen und deshalb von der Architektur nicht gewürdigt werden.(89,4)
  – Mittels Architektur-Sichten lassen sich gezielt bestimmte Aspekte eines IT-Systems betrachten. Damit verhelfen Architektur-Sichten dazu, mit der Komplexität aller Aspekte eines IT-Systems einfacher umzugehen.(101,1)

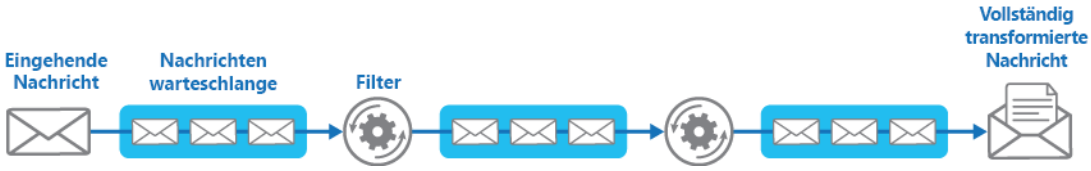
(c) ***Architekturstile**: Welche gibt es und wodurch zeichnen sie sich aus?*

- Ein Architektur-Stil gibt in erster Linie die fundamentale Struktur eines Software-Systems und dessen Eigenschaften wieder. Ein Stil kann also genutzt werden, um Architekturen zu kategorisieren. Ferner kann man Stile dazu verwenden, um die Konsequenzen einer fundamentalen Architektur und ihrer Varianten zu verstehen.(199,4) (Bsp. Pipes-and-Filters-Architektur.)
- Ein Architektur-Stil besteht bei Shaw und Garlan aus den folgenden Elementen:
  1. Eine Menge von Komponententypen, die bestimmte Funktionen zur Laufzeiterfüllen.(199,2)

2. Eine topologische Anordnung dieser Komponenten.(199,2)
3. Eine Menge von Konnektoren, die die Kommunikation und Koordination zwischen den Komponenten regeln.(199,2)
4. Eine Menge von semantischen Einschränkungen, die bestimmen, wieKomponenten und Konnektoren miteinander verbunden werden können.(199,2)

2 Architekturstile

(a) *Nennen Sie zu jedem der in der Vorlesung genannten Architekturstile ein Ihnen bekanntes Softwaresystem, das diesen Stil verwendet. Woran erkennen Sie das jeweils? (Nehmen Sie nicht die auf den Folien schon genannten Beispiele.)*

1. **Client / Server-Architektur:**
   - Beispiel **Email**:
   - Es gibt einen zentralen Rechner (Server), welcher die Daten (Emails) speichert.
   - Es gibt zahlreiche Klienten, welche die Daten abfragen und gegeben falls weitere Aktivitäten von dem Server beanspruchen (Versenden usw.)
2. **Schichten-Architektur:**
   - Beispiel **Betriebssystem**
   - Es gibt mehrere Schichten (Ringe) und höhere Schichten werden von niederen niemals benutzt.
   - Ring 0 → Kernel, Ring 1 → Executive, Ring 2 → Supervisor, Ring 3 → User
3. **Architekturstil Ereignisgesteuerung:**
   - Beispiel **Windows Ereignis Manager**
   - Es werden auf Ereignisse gelauscht und man wird benachrichtigt, falls eins Eintritt.
   - Man kann selbst Ereignisse erstellen.
4. **Architekturstil Datenflussnetze (Pipes und Filter):**
   - Beispiel **Email**
   - Vom senden einer Email, zur verschlüsselung im Server bis zum ankommen beim Adressaten muss eine Email oft bearbeitet werden. Dies kann mithilfe von Pipes und von Filter dargestellt werden.
   - 

   [6]
5. **Standartarchitektur: Web-basiertes System:**
   - Beispiel **Mobile API**
   - Creating these services have become easier using simplified web protocols, e.g. REST and JSON.[2]
   - These protocols are much easier for web developers, as they require less CPU and bandwidth. They are more recognised because of large social platforms, such as Facebook, Amazon and Twitter etc.[2]

(b) *Welche Architekturstile sind für welche der folgenden nicht-funktionalen Anforderungen besonders geeignet?*

1. ***Echtzeitverhalten*** *(d.h. zugesicherte Reaktionszeiten des Systems)*
   - Datenflussnetze (in der VL als Bsp Bioinformatik und Signalverarbeitung).
2. ***hohe Portabilität*** *(über mehrere Betriebssystemplattformen)*
   - Client-Server Architektur, Web-basierte Architektur. Also im Prinzip die Architekturen, welche nur wenig auf das Betriebssystem ankommen.

3 Architekturbeschreibung Ihres Softwareprojektes

(a) *In Aufgabe7-1sollten Sie dem4+1-Sichtenmodellvon Kruchten begegnet sein. Ma-chen Sie sich soweit mit diesem Modell vertraut, dass Sie den Zweck der einzelnenSichten verstehen.*

(b) *Entwerfen Sie die Architektur Ihres Softwaresystems. Nehmen Sie dabei die folgendenSichten ein und halten Sie jeweils mindestens drei wichtige Aspekte fest. Sollte IhrSystem aus einer der Sichten keine relevante Architektur haben, erläutern Sie dies.*

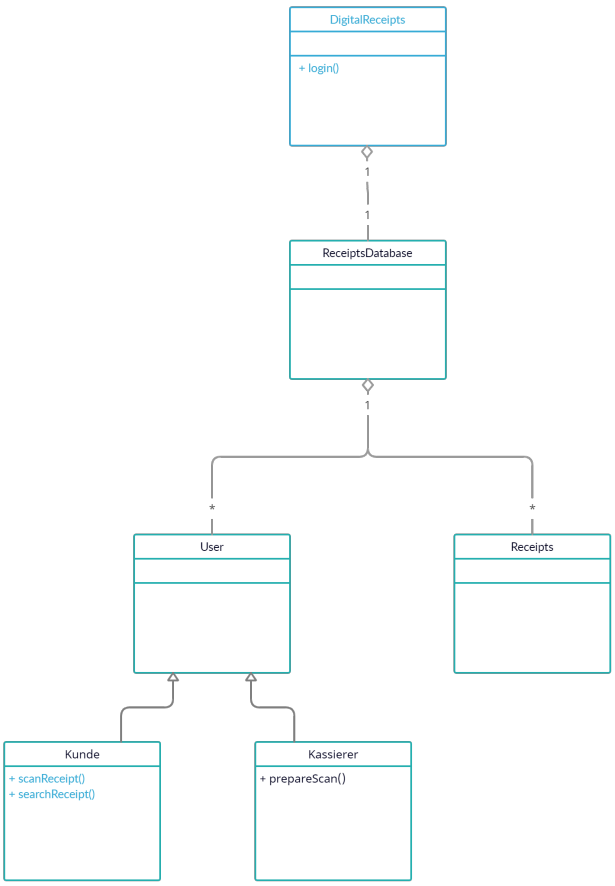
1. Anwendungsfallsicht(use-case view/scenarios)
   Die Anwendungsfallsicht umfasst die wichtigsten Anwendungsfälle, also geben wir für die Aufgabe drei wichtige Anwendungsfälle vor. Unserer Meinung nach sind die 3 wichtigsten Anwendungsfälle die folgenden:
   1. Quittung Scannen
   2. Scan vorbereiten
   3. Quittung in der App suchen (aus Übung 5)

Anwendungsfall	1.User_Scan_Receipt
Beschreibung	Benutzer scannt eine Quittung.
Akteure	Benutzer und App
Annahmen	Handy hat Kamera
Schritte	1. Benutzer öffnet App. 2. Benutzer klickt auf den Button zum Scannen einer Quittung. 3. App öffnet Kamera-Ansicht 4. Benutzer scannt QR-Code 5. App erkennt QR-Code und schließt App
Variationen	
Nicht-funktionales	Keine
Probleme	

Anwendungsfall	User_Prepare_Scan
Beschreibung	Kassierer bereitet einen Scan vor.
Akteure	Benutzer und App
Annahmen	Kasse hat einen Scan Button
Schritte	1. Benutzer öffnet App. 2. Benutzer klickt auf den Button zum Scannen einer Quittung. 3. App öffnet Kamera-Ansicht 4. Benutzer scannt QR-Code 5. App erkennt QR-Code, speichert es in die Datenbank und schließt App
Variationen	
Nicht-funktionales	Keine
Probleme	

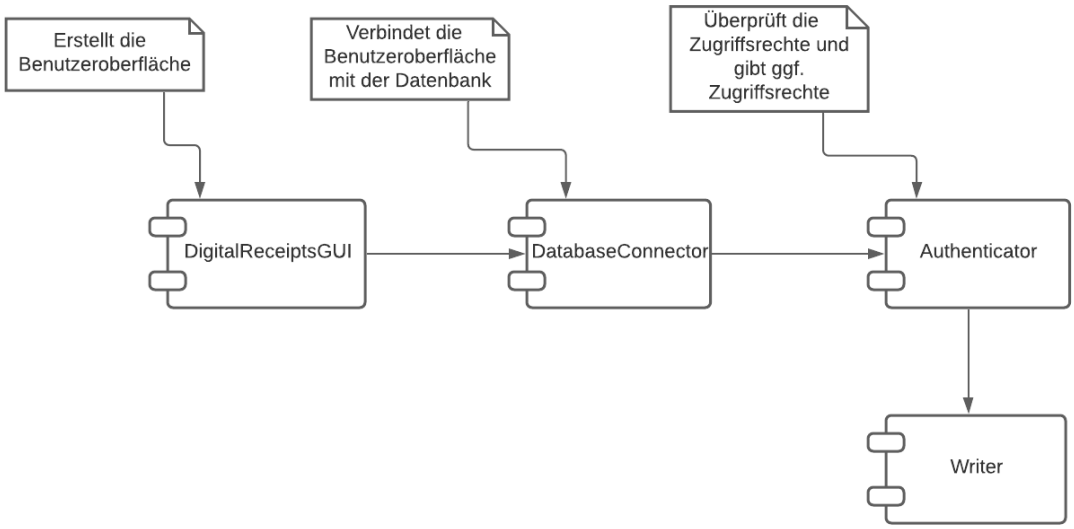
Anwendungsfall	3. User_Searchs_Receipt
Beschreibung	Benutzer sucht eine Quittung.
Akteure	Benutzer und App
Annahmen	Quittung ist in der App vorhanden.
Schritte	1. Benutzer öffnet App. 2. Benutzer klickt auf den Button zum Suchen einer Quittung. 3. App öffnet ein Dialog mit Filter für die Suche. 4. Benutzer trägt benötigte Daten ein: Datum/Zeitraum, Kategorie, Markt 5. Benutzer bestätigt seine Eingaben. 6. App schließt Dialog und zeigt alle Quittungen mit diesem Filter an.
Variationen	#4. Benutzer kann Daten weglassen #5. Benutzer bestätigt nicht und schließt den Dialog, alles bleibt so wie es ist.
Nicht-funktionales	Keine
Probleme	Falls Quittungen online gespeichert sind und kein Internetzugang besteht, werden die Quittungen nicht angezeigt.

2. Logische Sicht(logical view)
Aus der logischen Sicht, wird die Umsetzung der funktionalen Anforderungen betrachtet. Dabei werden UML-Diagramme zur Darstellung benutzt. In unserem Beispiel haben wir ein UML-Klassendiagramm benutzt um die in der Anwendungsfallsicht genannten funktionalen Anforderungen mithilfe eines Klassendiagramms darzustellen.
1. Unsere App soll sich mit der Datenbank verbinden können.
2. Unsere App soll es den Kunden ermöglichen die Quittungen zu suchen und zu scannen.
3. Unsere App soll es den Kassierern ermöglichen den Scan vorzubereiten.

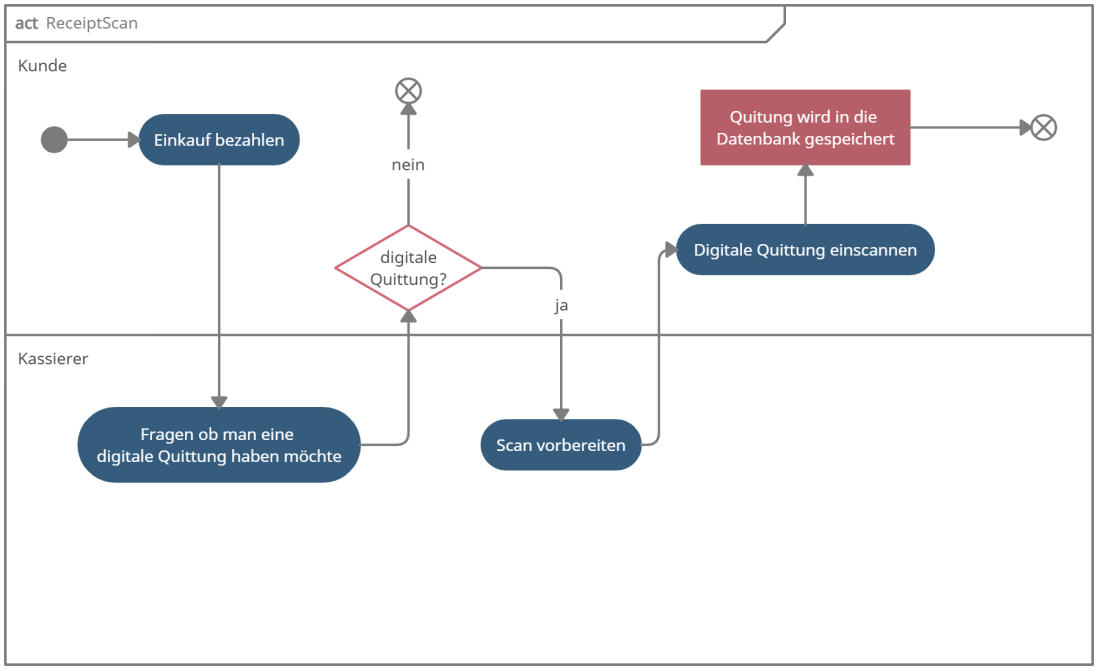


Anmerkung: DigitalReceipts soll die Benutzeroberfläche darstellen.

3. Implementierungssicht(implementation view/development view)
   Die Implementierungsschicht behandelt die Organisation und Verwaltung der Software (Softwaremanagement).
   Eine gute Verwaltung/Organisation schaffen wir mithilfe der Modularisierung, damit wir einen guten Überblick über unsere Software und ihre Einzelheiten haben.
   1. Die Benutzeroberfläche ist lokal gespeichert.
   2. Die App ist die Schnittstelle zum Server.
   3. Bevor man zugriff zum Server hat muss man sich Authentifizieren (login).
   Hier haben wir uns an die Pipe-and-Filter-Architektur aus der Vorlesung gehalten um das ganze einfach zu modellieren:



4. Prozesssicht (process view)
   Die Prozesssicht behandelt das Verhalten und Verteilung des Systems zur Laufzeit, da unser System auf mehreren Systemen läuft können wir das am besten mit einem Aktivitätsdiagramm modellieren. Hier haben wir unser wichtigste Funktion modelliert:
   1. Um eine digitale Quittung zu Scannen muss der Kassierer erst den Scan vorbereiten.
   2. Erst nach dem der Scan vorbereitet wurde kann man die digitale Quittung in form eines QR Code scannen.
   3. Erst dann wird die Quittung in die Datenbank gespeichert.



5. Verteilungssicht (deployment view/physical view)
   Die Verteilungssicht behandelt, wie die Implementierungssicht physikalisch umgesetzt werden kann.
   1. Die App stellen wir im App Store und im Play Store zur Verfügung.
   2. Die App stellt eine verschlüsselte Verbindung zum Server her über das Internet her.
   3. Die Quittungen sind alle auf dem Server gespeichert, damit man die nicht fälschen kann.

4 Quellen

[1] **Für Aufgabe 1:** “Webbasierte Systemarchitekturen”, https://link.springer.com/chapter/10.1007%2F978-3-322-89822-7_4 . [aufgerufen am 29.05.2021]

[2] “Web-oriented architecture”, https://en.wikipedia.org/wiki/Web-oriented_architecture [aufgerufen am 29.05.2021]

[3] **Für Aufgabe 3:** ”4+1 Schichtenmodell”, https://de.wikipedia.org/wiki/4%2B1_Sichtenmodell [aufgerufen am 29.05.2021]

[4] ”The 4+1 View Model of Architecture”, https://de.wikipedia.org/wiki/4%2B1_Sichtenmodell [aufgerufen am 29.05.2021]

[5] SSoftwareArchitektur”, https://link.springer.com/book/10.1007/978-3-8274-2267-5 [S.98-99]

[6] “Muster „Pipes und Filter”, https://docs.microsoft.com/de-de/azure/architecture/patterns/pipes-and-filters, [aufgerufen am 29.05.2021]