# Verwendung von Schnittstellen

Die wichtigsten Aspekte der Architektur sind Schnittstellen und Beziehungen zwischen den Komponenten. Schnittstellen sind Teil der Basis des Gesamtsystems, durch sie werden die Beziehungen zwischen den Systemteilen ermöglicht. Die einzelnen Bausteine und Teilsysteme kommunizieren und kooperieren über die Schnittstellen miteinander. Auch die Kommunikation mit der Außenwelt findet über Schnittstellen statt.

Quelle Basiswissen für Softwarearchitekten

## Warum sind Schnittstellen so wichtig?

Von technischen Details abgesehen, stecken die wichtigsten Aspekte fur Architekten in den Schnittstellen und Beziehungen zwischen den Bausteinen ([Rechtin2000]). Das hat mehreren Gründen:

* Die einzelnen Bausteine oder Teilsysteme kommunizieren und kooperieren über Schnittstellenmiteinander. Diese Zusammenarbeit bildet die Basis des Gesamtsystems – ohne Schnittstellen funktioniert nichts!
* Einzelne Bausteine konzentrieren sich auf ihre (lokalen) Probleme. Sie betrachten die übrigen Systeme oder Bausteine als „Peripherie“. Sie als Architekt halten die Faden des gesamten Systems zusammen – eben über die Schnittstellen.
* Über Schnittstellen findet die Kommunikation mit der Ausenwelt statt – ohne die jedes System nutzlos bleibt.
* Schließlich betreffen Fehler an Schnittstellen immer mindestens zwei Bausteine, nämlich sowohl Anbieter als auch Nutzer der Schnittstelle. Solche Probleme haben dadurch größere Konsequenzen – es sind nämlich mehr Leute betroffen, als wenn „nur“ ein einzelner Baustein defekt ist.

Quelle Effektive Softwarearchitekturen 8. Auflage, Innerhalb Kapitel 4.2 Heuristiken

# Abtrennung von Schnittstellen

Bei vielfacher Nutzung einer umfangreichen Schnittstelle kann es sinnvoll sein, diese in mehrere spezifischen Schnittstellen zu zerlegen, z.B.:

* Zerlegung nach semantischem Zusammenhang
* Zerlegung nach Verantwortungsbereich

Eine solche Zerlegung reduziert die Anzahl der abhängigen Benutzer und damit, auch mögliche Folgeänderungen. Außerdem sind kleine, fokussierte Schnittstellen leichter implementierbar und wartbar.

Quelle Basiswissen für Softwarearchitekten

# Anforderungen an Schnittstellen

Mir Consumer erden all diejenigen Klassen bzw. Objekte genannt, die diese Schnittstellen nutzen. Als Provider werden alle Klassen bzw. Objekte genannt, die die Schnittstelle implementieren.

Typische Qualtitätsmerkmale für Schnittstellen sind

* Die Anforderungen der Consumer erfüllen. Das ist gerade bei öffentlichen Schnittstellen sehr schwierig, da der Provider die Consumer meist nicht kennen.
* Einfach zu erlernen und zu benutzen.
* konsistent zu anderen Schnittstellen im System, also den „Gepflogenheiten“ des Systems entsprechen. Verwendet ein System durchgängig JSON, sollten Sie nicht an Ihrer Schnittstelle (überraschenderweise) XML nutzen
* Der Quellcode von Consumern (Clients, Benutzern der Schnittstelle) sollte verständlich und leicht wartbar sein. Machen Sie sich als Anbieter von Schnittstellen Gedanken, wie der entsprechende Consumer/Client-Code aussehen muss, um Ihre API zu verwenden.
* leicht erweiterbar. Consumer und Anwender kommen ständig auf neue Ideen und benötigen an Schnittstellen neue oder erweiterte Funktionalität – darauf sollte eine API vorbereitet sein

# Speziell zu Beachten

Für Softwareentwickler ist sofort klar, dass sie bei der Definition von Schnittstellen auf Datenformate und -typen achten, die Signatur von Methoden oder Funktionen klaren und darüber hinaus den Übertragungswagen zwischen Consumer und Provider entscheiden oder sicherstellen müssen.

Zusätzlich sollten Sie an Schnittstellen auf angemessene Fehlerbehandlung achten

Richtig kompliziert wird es allerdings, wenn wir noch weitere Anforderungen an Schnittstellen

hinzunehmen – ich stelle Ihnen eine kleine (schreckliche!) Auswahl vor:

* Sicherheit – Teil 1: Die Nutzung der Schnittstelle ist nur autorisierten oder authentisierten Nutzern (Consumern) erlaubt. Vor der Nutzung der Schnittstelle sind sicherheitsrelevante Vorarbeiten nötig.
* Sicherheit – Teil 2: Die Kommunikation über diese Schnittstelle ist vertraulich. Wir müssen Fragen der Verschlüsselung (Festlegung von Verschlüsselungsverfahren, Schlüssel- oder Zertifikatsbehandlung, Nichtabstreitbarkeit etc.) klaren.
* Qualitat: Die Schnittstelle soll bestimmte Garantien bezüglich Durchsatzes, Performance oder Verfügbarkeit geben. Jetzt müssen Sie erstens diese sogenannten *Qualities of Service* (QoS) beobachten oder messen und zweitens Maßnahmen ergreifen, falls die QoS nicht mehr eingehalten werden.
* Kosten: Die Benutzung der Schnittstelle kostet Geld. Beispiel: pro Aufruf oder pro Datenvolumen. Als Provider mochten Sie ein verlässliches Protokoll, welcher Consumer sie aufgerufen hat, als Consumer mochten Sie gleichermassen den Provider kontrollieren – und zwar buchhaltungs- oder fälschungssicher.

Quelle Effektive Softwarearchitekturen 8. Auflage Kapitel 4.4.2

# Tipps zum Entwurf von Schnittstellen

Einige Ratschlage24 zum Entwurf von Schnittstellen mochte ich Ihnen noch mitgeben – ohne Anspruch auf methodische Vollständigkeit:

* Benutzen Sie Ihre eigenen Schnittstellen, beispielsweise indem Sie Tests gegen diese Schnittstelle implementieren. Das hat gleich zwei Vorteile: Erstens merken Sie, ob sich dieArbeit mit der Schnittstelle aus Sicht eines Consumers *gut anfühlt* – und zweitens prüfen Sie deren Korrektheit. Auch bekannt als „eat-your-own-dogfood“.
* Klaren Sie die Anforderungen an Schnittstellen grundsätzlich mit Consumer und Provider, weil die beiden manchmal komplett unterschiedliche Sichtweisen oder Anforderungen haben. Das gilt gleichermassen für interne wie externe Schnittstellen.
* Verbergen Sie Implementierungsdetails von Bausteinen, beispielsweise intern genutzte Datenstrukturen. Damit beachten Sie das Geheimnisprinzip und können Ihre Bausteine unabhängig optimieren oder weiterentwickeln.
* An Schnittstellen sind sprechende Namen und deren konsistente Benutzung besonders, wichtig. Begriffe aus dem Domänenmodell bieten sich an
* . Minimieren Sie Überraschungen oder Seiteneffekte: Benutzer Ihrer Schnittstelle sollten immer genau (!) das bekommen, was sie erwarten (bzw. was die Namen von Methoden oder Schnittstlen suggerieren). Das ist eine Konkretisierung des Grundprinzips „Separation of Concerns“.
* Stellen Sie an Schnittstellen „atomare“ Informationseinheiten bereit, anstelle Informationen in Zeichenketten zu kombinieren. Nutzer der Schnittstelle sollten einfachen Zugriff auf Informationen erhalten, statt Strings parsen zu müssen.
* Dokumentieren Sie öffentliche Schnittstellen. Denken Sie dabei insbesondere an Sonder- und Fehlerfalle. Als einfache Form der Dokumentation schlage ich Unit-Tests vor entwicklerfreundlich, ausfuhrbar und Sie sollten die ohnehin haben.

Quelle Effektive Softwarearchtitekturen 8. Auflage Kapitel 4.4.3

# Spezielle Behandlung von externen Schnittstellen

Treffen Sie mit externen Systemen moglichst exakte Absprachen hinsichtlich der auszutauschenden Daten. Beachten Sie dabei neben der reinen Syntax und Semantik der ausgetauschten Daten auch Formate (Zeichen- und Zahlencodierung, Schlusselwerte etc.), Muss-/Kann-Belegungen, Wertebereiche, Ausnahme- oder Sonderfalle, Grenzfalle (Nullwerte, leere Dateien, überlange Zeichenketten,etc.), Ubertragungkanäle und -protokolle, Häufigkeit des Datenaustausches, eventuelle Berechtigungs- oder Sicherheitsaspekte, Transaktions- und Zustandsbehandlung, eventuell notwendige Transformation, Synchronität, Statusverfolgung und andere. Kann das liefernde System bereits gelieferte Daten nachträglich für ungültig erklären und durch eine neue Version ersetzen?

Quelle Effektive Softwarearchitekturen 8. Auflage Kapitel 7.10.3