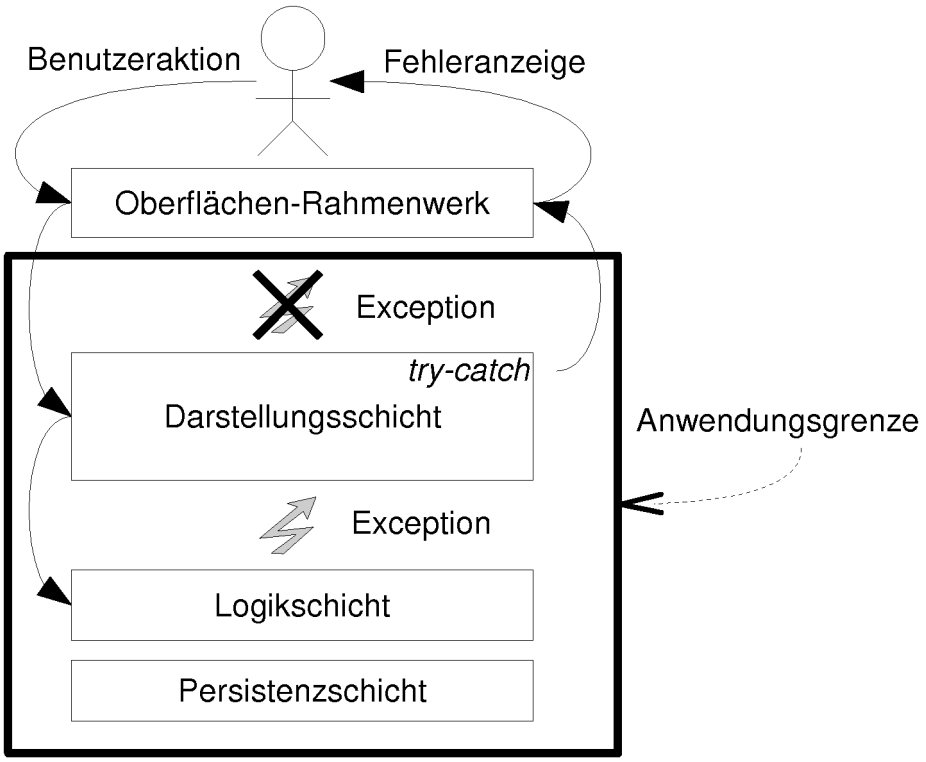
Oberflächentechnologie neutrale Lösung mit Aspektorientierung

Die Umsetzung von zentraler Ausnahmebehandlung hängt stark von dem verwendetem Oberflächenrahmenwerk ab. Im günstigen Fall bietet das Rahmenwerk entsprechende Mechanismen an, im ungünstigen Fall aber nicht. In diesem Abschnitt wird ein Ansatz vorgestellt, welches Aspektorientierung verwendet und sich für jedes Rahmenwerk einsetzten lässt.

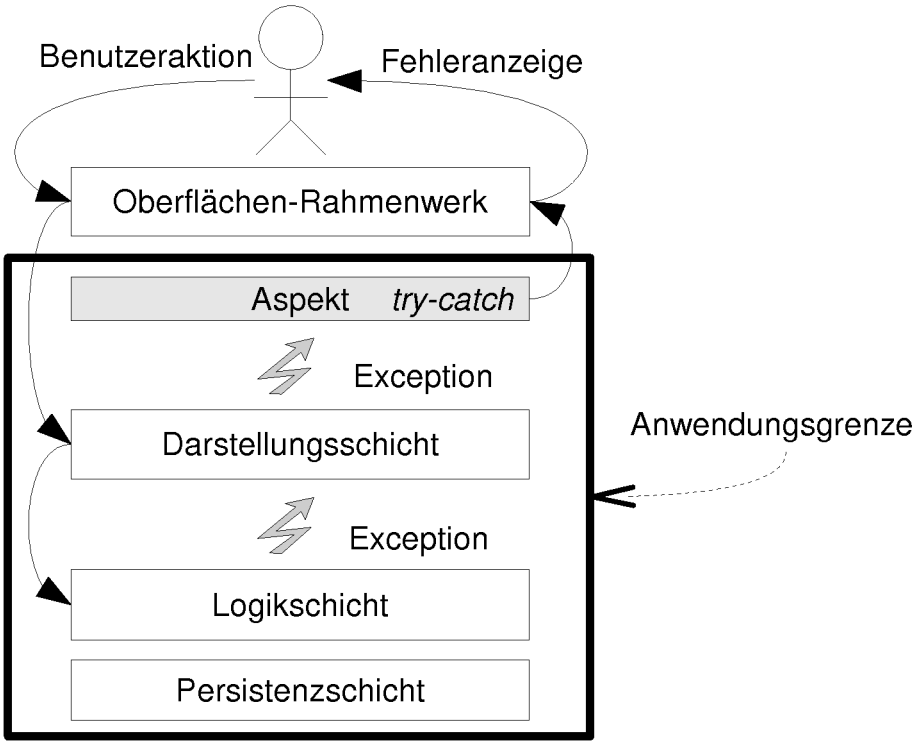
In Abbildung 3 sind die typischen 3-Schichten einer Anwendung dargestellt. Eine Benutzeraktion wird von Oberflächenrahmenwerk an die Darstellungsschicht der Anwendung weitergereicht. Diese wiederum ruft aus der Logikschicht eine Aktion auf, welche in einer Exception resultiert. Um diese Exception in Form einer Fehlerseite anzuzeigen, kann diese Exception nicht an das Oberflächenrahmenwerk weiter gereicht werden, da diese unter Umständen in einen undefinierten Zustand überführt werden würde. Stattdessen muss diese Exception abgefangen und wie beschrieben gemeldet werden. Dieses Verhalten muss für jede Interaktion sichergestellt werden und stellt daher einen querschnittliches Anliegen dar.



>>haschemi_aspekt_1.tif<<
Abb. 3: Benutzeraktion und Exception-Handling

Unser Ansatz ist es, dieses Anliegen in einem Aspekt zu kapseln und damit sowohl eine Trennung von technischem (zentrale Ausnahmebehandlung) und fachlichem Code, als auch eine Minimierung der Redundanz zu erreichen.

Mit dem Hinzufügen des Aspekts kann man in Abbildung 4 sehen, wie sich jener Aspekt zwischen der Darstellungsschicht und dem Oberflächen-Rahmenwerk eingliedert, alle Exceptions abfängt und sie meldet.



>>haschemi_aspekt_2.tif<<
Abb. 4: Benutzeraktion und Exception-Handling mit einem Aspekt

Um diesen Aspekt umsetzten zu können, müssen alle Stellen in der Darstellungsschicht identifiziert werden, die potenziell Exceptions produzieren können. Als konkretes Beispiel werden wir die zentrale Ausnahmebehandlung des Swing-Beispiels (mit dem Ansatz des Entwurfsmusters *template method*) mit einer aspektorientierten Version ersetzten.

Listing 2

```
public aspect SwingExceptionAspect {
  pointcut pointOfInterest(ActionEvent ev)
    : execution(void AbstractAction+.actionPerformed(ActionEvent))
       && args(ev);

  void around(ActionEvent ev) : pointOfInterest(ev) {
    try {
      proceed(ev);
    } catch (Exception exception) {
      CentralExceptionReporter.reportException(ev, exception);
    }
  }
}
```

Ende Listing 2

In Listing 2 ist der Aspekt, der die zentrale Ausnahmebehandlung durchführt, aufgelistet. Die verwendete Programmiersprache ist *AspectJ* [5][6], eine aspektorienierte Erweiterung von Java.

In den Zeilen 2 bis 4 wird der *Pointcut pointOfInterest* beschrieben, der alle *actionPerformed* Methoden der Klasse *AbstractAction* (oder Unterklassen von *AbstractAction*) referenziert. In den

Zeilen 6 bis 12 wird um die Ausführung aller *actionPerformed* Methoden ein try-catch-Block konstruiert, der alle Exceptions abfängt und meldet.

Somit stellen sich gegenüber dem ersten Ansatz folgende Vorteile ein:

- die Verwendung der Action-Unterklasse *ExceptionReportingSwingAction* und damit das Vermischen von technischem und fachlichen Code wurde verhindert
- TODO

Die wichtigsten Konzepte von AspectJ

Aspekt

Ein Aspekt ist eine modulare Umsetzung eines querschnittlichen Anliegens. Es bildet den Rahmen für die Implementation von Anforderungen und ist somit vergleichbar mit einer Klasse in der Objektorientierung.

Advice

In einem Advice wird die Zusatzfunktionalität implementiert. Hier steht der Programmtext der Anforderung.

Join Point

Ein Join Point ist spezifische Stelle innerhalb des Ablaufs einer Applikation, an denen ein *Advice* ausgeführt werden kann. Beispiele für Join Points sind:

- Methodenaufruf
- Methodenausführung
- Konstruktoraufruf
- Konstruktorausführung
- Bevor ein Objekt initialisiert wird
- Während ein Objekt initialisiert wird
- Bevor ein Objekt statisch initialisiert wird
- Wenn auf eine Klassenvariable zugegriffen wird.

Pointcut

Mit Hilfe eines Pointcut kann man das Interesse an einem *Join Point* im System verkünden, um an dieser Stelle einen *Advice* auszuführen. Die Auswahl eines *Join Point* geschieht über ein Muster, indem Wildcards verwendet werden können.

Ende Kastentext

TODO Wenn das Darstellungsframework die Änderung der Signatur der Methoden für die Ereignisbehandlung

nicht zulässt, dann unchecked-Exceptions benutzen.

Siamak Haschemi hat im Oktober 2006 das Studium der Medieninformatik abgeschlossen und ist seit dem Softwareingenieur für die Entwicklung von individuellen Softwarelösungen bei der sd&m AG – software design & management Berlin. Seine Schwerpunkte umfassen Enterprise Java (J2EE), Modellgetriebene Softwareentwicklung (MDSD) und Rich-Internet-Applications (RIAs).

Links & Literatur

[5] AspectJ www.eclipse.org/aspectj

[6] AspectJ in Action: Practical Aspect-Oriented Programming (Paperback) , Ramnivas Laddad, ISBN 1-930-11093-6