Fehlerbehandlung in Programmiersystemen

RALF HINZE

Institut für Informatik III, Universität Bonn
Römerstraße 164, 53117 Bonn, Germany
Email: ralf@informatik.uni-bonn.de
Homepage: http://www.informatik.uni-bonn.de/~ralf

17. März 2006

(Die Folien sind online verfügbar: .../~ralf/talks.html#T47.)

Überblick

Bisher:

- Softwarekomponenten (Halbfabrikate)
- Komponentenmodelle
  - JavaBeans
  - Component Object Model (COM/ActiveX)
    - binärer Standard
    - Bindungen an unterschiedliche Programmiersprachen

Heute:

- Fehlerbehandlung in COM
- Ausnahmebehandlung in ML und Java

Lernziele

- Die Fehlerbehandlung in COM erklären können.
- Den allgemeinen Mechanismus zur Ausnahmebehandlung in Programmiersprachen erläutern können.
- Ausnahmebehandlung in Java anwenden können.

Motivation

Ein wesentliches Qualitätsmerkmal von Software ist Zuverlässigkeit:

- Korrektheit: die Software tut, was sie soll.
- Robustheit: die Software kommt mit unerwarteten Situationen zurecht.

☞ Je höher der Grad der Wiederverwendbarkeit (Modul, Klasse, Paket, Komponente), desto höher der Stellenwert der Zuverlässigkeit: nichts ist fataler als die Wiederverwendung fehlerhafter oder fragiler Software.

Fehlerbehandlung in COM

Fehlerbehandlung in COM

Designprinzipien von COM:

- einfach,
- sprachunabhängig und
- ortstransparent.

Insbesondere die ersten beiden Designprinzipien bedingen, dass

- Ausnahmen nicht über Interfacegrenzen hinaus propagiert werden, da es kein allgemeines semantisches Modell von Ausnahmen gibt, das sprach- und betriebssystemunabhängig ist.

☞ Stattdessen werden Fehler über den Rückgabewert von Funktionsaufrufen kommuniziert.

Der Statuscode HRESULT

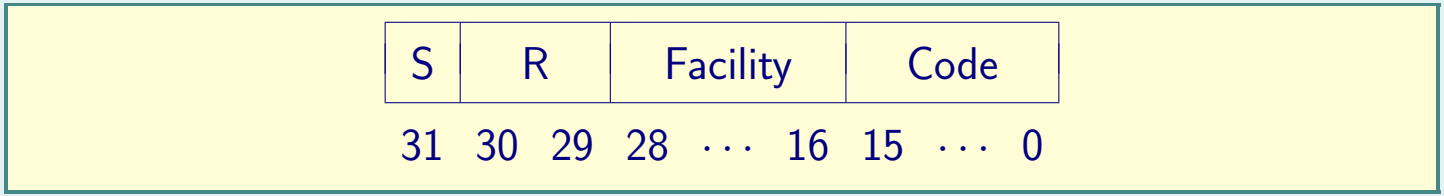
Zur Erinnerung: jede Komponente muss die Schnittstelle IUnknown implementieren:

```
[
 object,
 uuid(00000000-0000-0000-C000-000000000046),
 pointer_default(unique)
]
interface IUnknown
{
 HRESULT QueryInterface([in] REFIID iid, [out] void **ppv);
 ULONG   AddRef(void);
 ULONG   Release(void);
}
```

☞ Fehler beim Aufruf von COM API Funktionen oder von Interface Funktionen werden über den Rückgabewert vom Typ HRESULT bekanntgegeben.

Der Statuscode HRESULT — Fortsetzung

Ein Element vom Typ HRESULT ist ein einfacher 32-Bit Wert.



- S: severity ($0 \cong$ success, $1 \cong$ error),
- R: reserviert,
- Facility: spezifiziert die Gruppe des Statuscodes (ITC, RPC, STORAGE etc),
- Code: der eigentliche Rückgabewert bzw. der Fehlercode.

Sprachanbindung

Die COM Bibliothek stellt verschiedene Makros und Funktionen bereit, um Statuscodes zu manipulieren.

```
#define SUCCEEDED(Status)  ((HRESULT)(Status) >= 0)
#define FAILED(Status)     ((HRESULT)(Status) <  0)
```

☞ Abhängig von der Programmiersprache können die Statuscodes in lokale Ausnahmen überführt werden bzw. umgekehrt lokale Ausnahmen über 'error codes' kommuniziert werden:

- durch entsprechende Bibliotheken und/oder
- mittels geeigneter Werkzeuge (tool support).

☞ Zusätzliche Informationen zur Fehlerursache etc. können über das optionale `IErrorInfo` Interface ausgetauscht werden (`CreateErrorInfo`, `GetErrorInfo`). Fehler können darüber hinaus protokolliert werden (logging).

Ausnahmebehandlung in ML und Java

Ausnahmebehandlung in Programmiersprachen

Konstrukte zur Ausnahmebehandlung (engl. exception handling) bieten eine strukturierte Form von Sprüngen, um aus Blöcken oder aus Funktionsaufrufen „herauszuspringen“.

> Der Begriff „Ausnahme“ suggeriert, dass Ausnahmen nur in Ausnahmesituationen verwendet werden sollen. Dies ist intendiert, kann aber natürlich nicht von einer Programmiersprache erzwungen werden.

Viele Programmiersprachen bieten Konstrukte zur Ausnahmebehandlung an (z.B. Ada, C++, Java, ML). Diese umfassen mindestens

- einen Ausdruck oder eine Anweisung, um Ausnahmen auszulösen (engl. raise oder throw),
- einen Mechanismus, um Ausnahmen abzufangen und zu behandeln (engl. handle oder catch).

Ausnahmebehandlung in ML

In ML sind Ausnahmen Daten, Elemente des Typs `exn`.

```
exception Overflow;
exception NotFound of string;

raise Overflow;
raise (NotFound word);

lookup dictionary "computer"
  handle NotFound s => s ^ " is unknown"
       | Overflow   => raise Overflow;
```

☞ Im Fall einer Ausnahme wird die Auswertung abgebrochen und mit der Auswertung des nächsten passenden „exception handlers“ fortgefahren. Dieser wird dynamisch bestimmt.

Ausnahmebehandlung in Java

In Java sind Ausnahmen Objekte, Instanzen der Klasse `Throwable` oder einer ihrer Unterklassen (insbesondere `Exception`).

```
class Overflow extends Exception { }
class NotFound extends Exception {
  String key;
  NotFound (String k) {
    key = k;
  }
}

throw new Overflow();
throw new NotFound(word);
```

Ausnahmebehandlung in Java — Fortsetzung

Ausnahmen werden mit **try-catch-finally**-Blöcken abgefangen und behandelt:

```
try {
    dictionary.lookup("computer");
} catch (NotFound e) {
    System.err.println(e.key + " is unknown");
} catch (Overflow e) {
    throw e;
} finally {
    System.out.println("bye bye");
}
```

☞ Der optionale **finally**-Block wird unabhängig vom Ergebnis des **try**-Blocks auf jeden Fall ausgeführt.

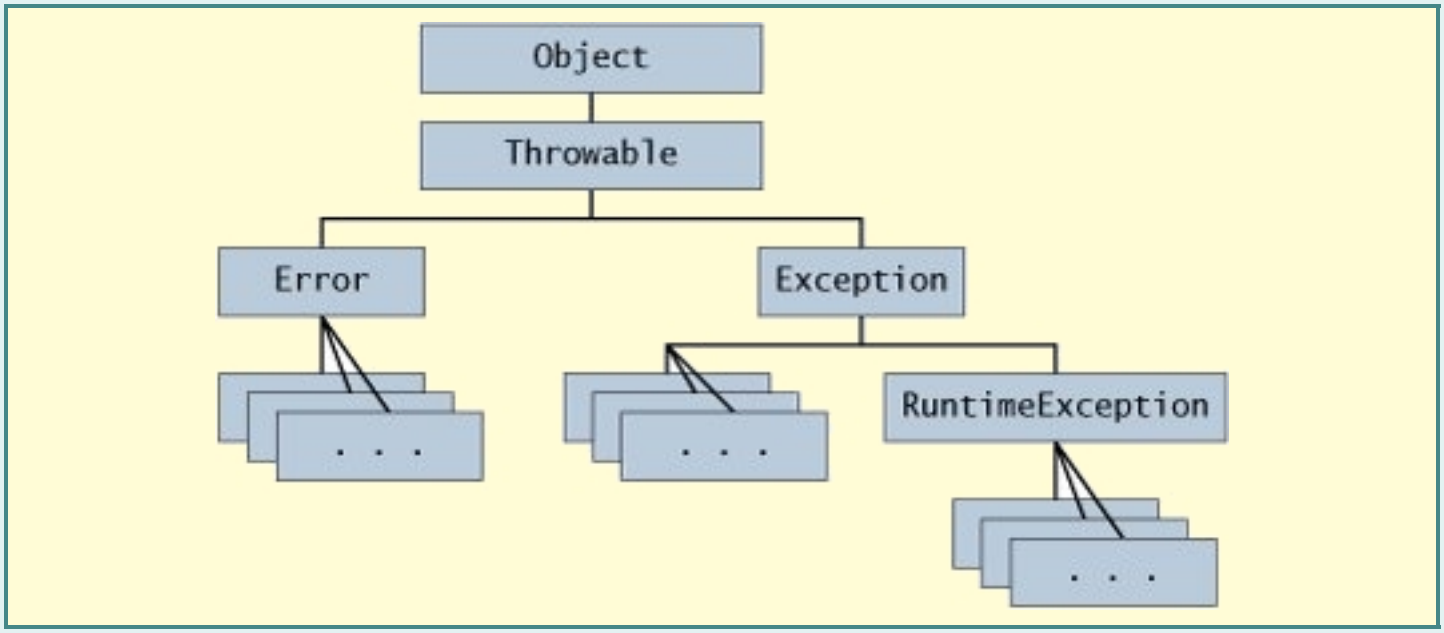
Statische Überprüfung von Ausnahmen

Zur Schnittstelle einer Methode gehören neben den Parametern und dem Rückgabewert auch eine Liste der Ausnahmen, die möglicherweise in der Methode ausgelöst werden („catch or specify“).

```
void translate() throws Overflow { }
```

☞ Auf diese Weise wird zur Übersetzungszeit sichergestellt, dass alle Ausnahmen auch tatsächlich behandelt werden (checked exceptions). Ausgenommen von der Prüfung sind lediglich `RuntimeExceptions`.

`Throwable` in der Klassenhierarchie



- `Error`: Systemfehler („hard failure“),
- `RuntimeException`: vom Laufzeitsystem ausgelöste Ausnahmen („soft failure“).

Zusammenfassung

- In COM werden Fehler über den Rückgabewert kommuniziert.
- Konstrukte zur Ausnahmebehandlung umfassen Anweisungen zum Auslösen von Ausnahmen und Anweisungen zum Abfangen von Ausnahmen.
- In Java sind Ausnahmen in die Klassenhierarchie eingebettet.