Kapitel 3 Fehlernummernkreise im UML-Repository

Dieses Dokument definiert die Fehlernummernkreise bei der Implementierung des UML-Repositories.

3.1 Allgemeines Schema für Fehleridentifikatoren

Die Fehleridentifikatoren im UML-Repository werden in einer baumartigen Struktur vergeben. Jede Ebene des Baumes definiert eine Menge von Fehlernummern. Ein Pfad von der Wurzel des Baumes zu einem Blatt ergibt dann einen Fehleridentifkator.

Die textuelle Darstellung eines Fehleridentifikators setzt sich aus der Zeichenfolge '[SERUM: ' gefolgt vom Fehleridentifikator zusammen. Dieser wird wie ein Dateisystempfad dargestellt, also als Folge von Nummern getrennt durch ein '/'. Nach der letzten Nummer folgen erst die Zeichen ' [', dann ein sinnvoller Fehlertext und abschließend die Zeichen ']]'.

Anmerkung: *Die Klammerung '[SERUM: ...]' erlaubt es, noch weitere Informationen anzuhängen. Funktionen, die den Fehlertext analysieren dürfen nicht davon ausgehen, daß nur der Fehleridentifikator im Text steht!*

Durch die Baumstruktur kann jeder in seinem Bereich unabhängig von anderen Fehlernummern vergeben. Diese werden dann zentral gewartet, wenn die entsprechenden Pakete freigegeben sind.

Fehler werden aus historischen Gründen mit der Ifx.-Fehlernummer 746 angezeigt. Im nachfolgenden Text werden unsere Fehlernummern kodiert.

3.1.1 Funktionen zum Auslösen eines Fehlers.

Mit folgenden beiden Funktionen können Fehler angezeigt werden:

```
SERUM_raise_exception( error_number_path LIST(INTEGER), explanation LVARCHAR );
SERUM_raise_exception( error_name_path LIST(VARCHAR), explanation LVARCHAR );
```

Anmerkung: *Die Realisierung soll in 'Java' erfolgen.*

Mit diesen Funktionen können Entwickler 'SERUM'-Fehlernummern erzeugen. Die oberste Variante nimmt den vollständigen Pfad als Folge von Integer-Werten entgegen, während die untere die Angabe der Fehlernummer als Folge von Fehlernummernamen erlaubt. Die interne Darstellung erfolgt immer als Folge von Integer-Werten.

Anmerkung: *Was bedeutet, daß es eine interne Tabelle mit den Fehlernummern, ihren Namen und der Baumstruktur geben muß. Über diese wird auch kontrolliert, ob der Fehleridentifikator überhaupt korrekt ist!*

3.1.2 Funktionen zum Behandeln von Fehlern

Mit folgenden Funktionen können Anwendungsprogramme Fehler abfangen und behandeln.

SERUM_get_error_explanation (error_string LVARCHAR) RETURNING LVARCHAR;

Diese Funktion extrahiert aus der Zeichenkette der Ifx-Fehlermeldung die Erklärung.

SERUM_get_error_path(error_string LVARCHAR) RETURNING LVARCHAR;

Diese Funktion extrahiert aus der Zeichenkette der Ifx-Fehlermeldung den Fehlernummerpfad.

SERUM_get_root_error_path (error_numner_path LVARCHAR) RETURNING LVARCHAR;

Diese Funktion extrahiert aus dem Fehlernummerpfad die oberste Fehlernummer (die Wurzel).

SERUM_get_son_error_path (error_numer_path LVARCHAR) RETURNING LVARCHAR;

Diese Funktion liefert den Fehlernummerpfad unterhalb der Wurzel. Wenn es keinen solchen Pfad mehr gibt, so wird ‘’ als Ergebniszeichenkette geliefert.

SERUM_get_error_number(error_leaf LVARCHAR) RETURNING INTEGER;

Diese Funktion liefert den Integer-Wert zu einer Fehlernummer.

SERUM_get_error_name(error_leaf LVARCHAR) RETURNING LVARCHAR;

Diese Funktion liefert den Namen zu einer Fehlernummer.

Anmerkung: *Realisierung in ‘Java’. Diese Funktionen kommen nur in Anwendungsprogrammen, bzw. in integrierter Anwendungslogik zur Anwendung. Passendes Konzept?*

3.2 Nummerkreis: Ausnahmebehandlung

Dieser Nummerkreis definiert Fehleridentifikatoren für den Bereich Ausnahmebehandlung.

- Nummer: 100
- Name: ‘Exception’

3.2.1 Exception/Internal

Dieser Nummernkreis ist für Fehler vorgesehen, die aus internen Fehlerzuständen resultieren.

- Nummer: 10
- Name: ‘Internal’

3.2.2 Exception/External

Dieser Nummernkreis ist für Fehler vorgesehen, die aus externen Fehler resultieren, beispielsweise wenn ein falsche Fehlernummeridentifikator übergeben wird.

- Nummer: 20

- Name: ‘External’

Exception/External/InvalidErrorIdentifier

- Nummer: 10
- Name: ‘InvalidErrorIdentifier’
- Explanation: Hier wird der ungültige Fehleridentifkator eingetragen.

Sei ‘[SERUM: 500/45/98 [ein Test]]’ ein ungültiger Fehleridentifkator, so sieht hat der zu erzeugenden Identifkator die folgende textuelle Darstellung:
‘[SERUM: 100/20/10 [[SERUM: 500/45/98 [ein Test]]]’

3.3 Nummernkreis: SERUM-Primärschlüssel

Dieser Nummernkreis ist Fehlernummern aus dem Primärschlüssel vorbehalten.

- Name: ‘PrimaryKey’
- Nummer: 100

3.3.1 PrimaryKey/Internal

Dieser Nummernkreis ist für Fehlernummern, die aus internen Fehlern resultieren, vorgesehen.

- Nummer: 10
- Name: ‘Internal’

3.3.2 PrimaryKey/External

Dieser Nummernkreis ist für Fehlernummern, die aus externen Fehleren resultieren, vorgesehen.

- Nummer: 20
- Name: ‘External’

Exception/External/CheckId

Dieser Nummernkreis ist für Fehler vorgesehen, die daraus resultieren, daß die Überprüfung eines Objektidentifikators fehlschlägt. Die genaue Ursache (Tabelle existiert nicht, ...) wird durch entsprechende Fehlernummern verfeinert.

- Nummer: 10
- Name: ‘CheckId’

Exception/External/InvalidUserObjectId/InvalidTable

Die im Objektidentifikator kodierte Tabelle ist falsch.

- Nummer: 10
- Name: ‘InvalidTable’

Exception/External/InvalidUserObjectId/InvalidTable

Der im Objektidentifiaktor kodierte Typ ist falsch.

- Nummer: 20
- Name: ‘InvalidType’

3.4 Nummernkreis: Well-formedness Rules

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die ausgelöst werden, wenn eine im Standarddokument definierte WfRs nicht erfüllt ist.

- Name: ‘Well-formedness Rule’
- Nummer: 200

3.4.1 Core

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die aufgrund von nicht erfüllten WfRs im Package ‘Core’ resultieren. Für jede Klasse wird ein eigener Fehlernummernkreis vergeben.

- Name: ‘Core’
- Nummer: 10

3.4.2 ModelManagement

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die aufgrund von nicht erfüllten WfRs im Package ‘ModelManagement’ resultieren.

- Name: ‘ModelManagement’
- Nummer: 20

3.4.3 DataTypes

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die aufgrund von nicht erfüllten WfRs im Package ‘DataTypes’ resultieren.

- Name: ‘DataTypes’
- Nummer: 30

3.4.4 ExtensionMechanism

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die aufgrund von nicht erfüllten WfRs im Package ‘ExtensionMechanism’ resultieren.

- Name: ‘ExtensionMechanism’
- Nummer: 40

3.4.5 StateMachine

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die aufgrund von nicht erfüllten WfRs im Package ‘StateMachine’ resultieren.

- Name: ‘StateMachine’
- Nummer: 50

3.5 Nummerkreis: StateMachines

Dieser Nummernkreis ist Fehlermeldungen vorbehalten, die ausgelöst werden, wenn eine State-Machine eine Verletzung der semantischen Integrität feststellt.

- Name: ‘StateMachine’
- Nummer: 300