

UML + Perl = ?

Renée Bäcker, **Smart-Websolutions**

Warum UML?

- Vereinfachung von Dokumentation
 - Auch „Managertauglich“
- Modellierung von Anwendungen
 - Austausch zwischen Programmierern wird einfacher
- Perl für Unternehmensanwendungen
 - UML oft eine Voraussetzung

Kurze Einführung in UML



- UML = Unified Modeling Language
 - Mittlerweile Version 2.0
 - Seit 1997 Standard der OMG
- Entwickelt von „drei Amigos“
 - James Rumbaugh
 - Ivar Jacobson
 - Grady Booch

Kurze Einführung in UML



- Sprache zur Modellierung
 - legt Beziehungen fest
 - graphische Elemente
- Modellierung über Diagramme
 - Strukturdiagramme
 - Verhaltensdiagramme

Klassendiagramm

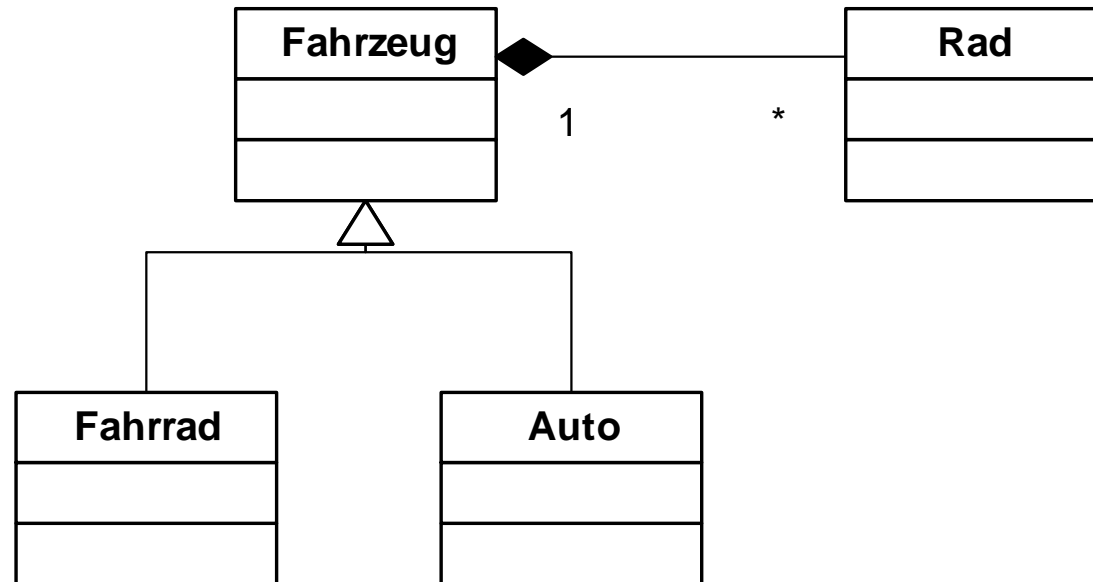


- graphische Darstellung Klassen und Beziehungen
- Teil von Objektorientierter Analyse und Design

Klassendiagramm - Bestandteile

- Klassen
 - Klassenname
 - Attribute
 - Methoden
- Beziehungen
 - Aggregation und Komposition
 - Assoziation
 - Generalisierung / Spezialisierung

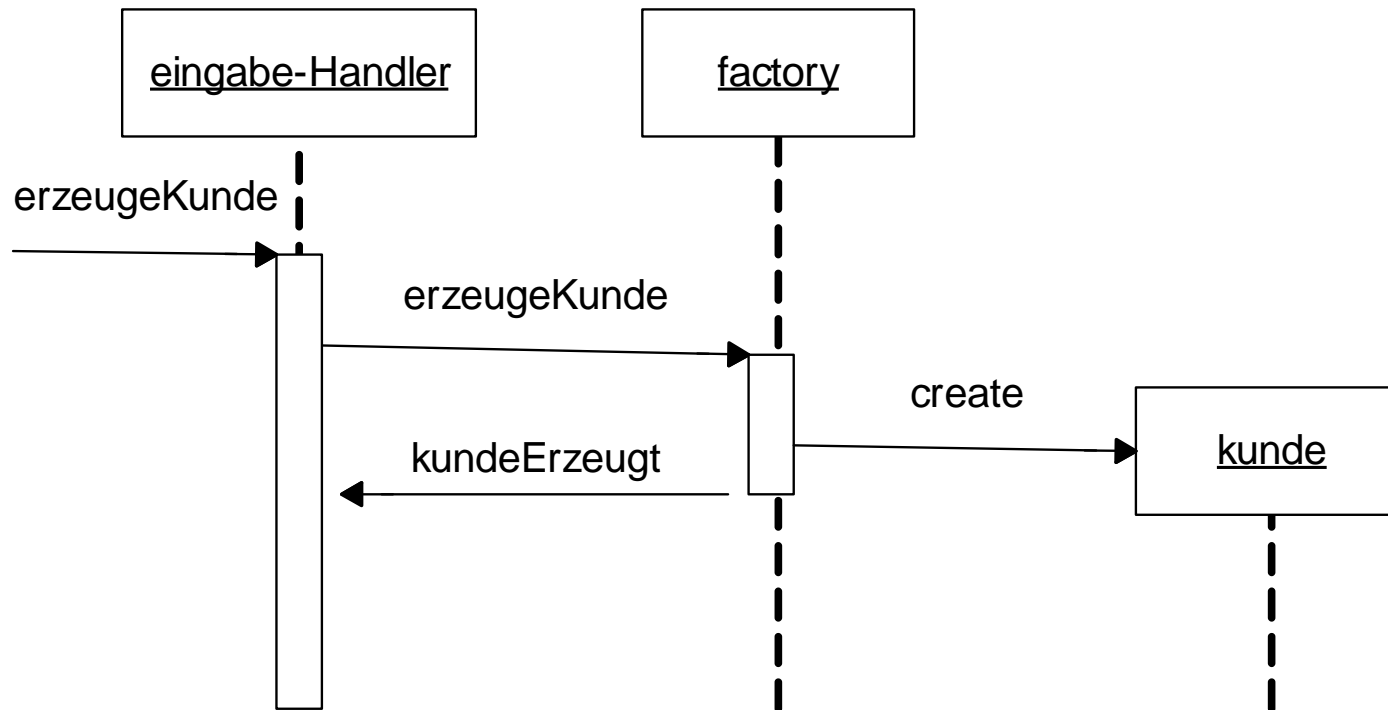
Beispiel: Klassendiagramm



Sequenzdiagramm - Bestandteile

- Zeigt Nachrichten und Interaktionen
 - Verlorene und gefundene Nachrichten
- Darstellung über Lebenslinien
- Ablauf des Programms

Beispiel: Sequenzdiagramm



UML und Perl verheiraten



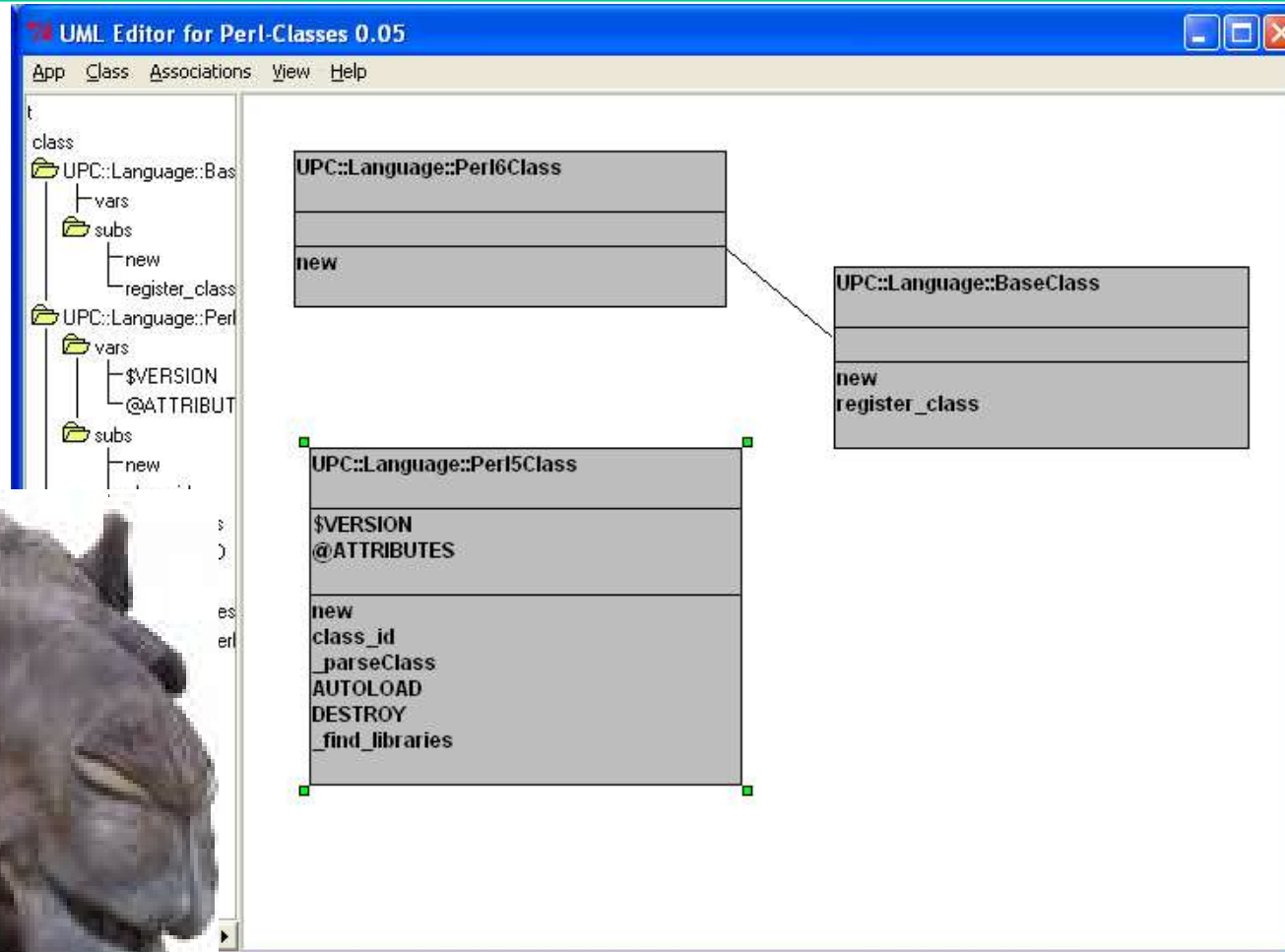
- UPC
 - Klassendiagramm
- UML::Sequence
 - Sequenzdiagramm
- autodia.pl
 - Klassendiagramm
- UML::Class::Simple
 - Klassendiagramm
- UMMF
 - Parsen von Meta-Informationen



UPC – UML-Editor für Perl

- Parsen von Perl mittels PPI
- Erstellt Klassendiagramme
- Ist noch in der Entwicklung
- Graphische Oberfläche
- Roundtrip Engineering soll ermöglicht werden

UPC - Ein UML-Editor für Perl



23.02.2007

UML + Perl = ?

UML::Class::Simple

- Erzeugt Klassendiagramme
- Aus bestehendem Code
- Verwendet GraphViz für Graphik
- Parsing mit PPI
- Einfache Bedienung
- Kein Roundtrip Engineering

UML::Class::Simple

```
package MyTest;  
  
our $VERSION = 0.01;  
  
sub my_test{}  
  
package MyTest::Child;  
  
use base 'MyTest';  
  
sub child_test{};
```

UML::Class::Simple

```
#!/usr/bin/perl
```

```
use strict;
```

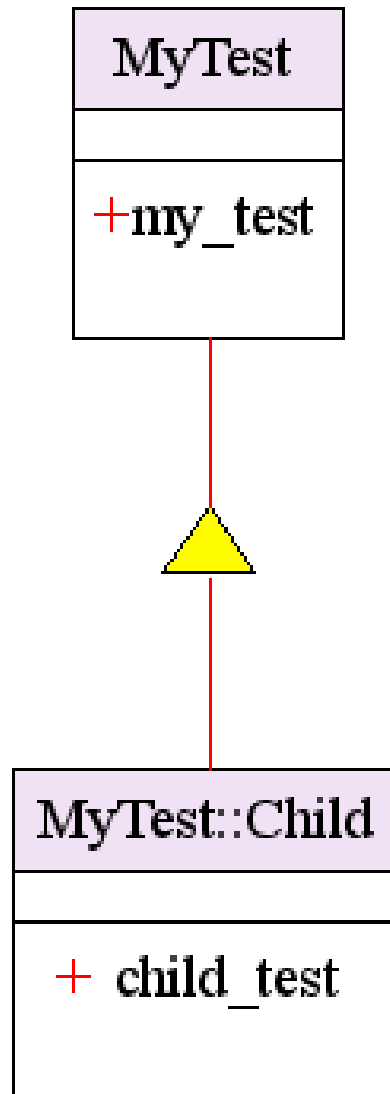
```
use warnings;
```

```
use UML::Class::Simple;
```

```
my @classes = classes_from_runtime(  
    "MyTest",  
    qr/^MyTest/);
```

```
my $obj = UML::Class::Simple->new(\@classes);  
$obj->as_png('MyTest.png');
```

UML::Class::Simple



23.02.2007

UML::Sequence

- Erzeugt Sequenzdiagramme
- Durch Ablauf des Programms
- Nicht so gut dokumentiert
- Ausgabe in SVG oder PNG
- Umständlich von Code zu Graphik
- Bearbeiten nicht möglich bei PNG
- Dennoch gutes Modul für Verständnis

UML::Sequence – ein Beispiel

- Einfach ein paar Sub-Aufrufe
- Aufruf von Subs eines packages

```
#!/usr/bin/perl
```

```
package Test;
```

```
sub testsub1{}
```

```
sub testsub2{ testsub3() }
```

```
sub testsub3{}
```

```
package main;
```

```
test1(); test3(); test7();
```

```
Test::testsub1();
```

```
Test::testsub2();
```

```
sub test1{ test2() }
```

```
sub test2{ }
```

```
sub test3{ test4(); test5() }
```

```
sub test4{ }
```

```
sub test5{ test6() }
```

```
sub test6{ }
```

```
sub test7{ }
```

UML::Sequence – ein Beispiel

```
#!/usr/bin/perl
```

```
use UML::Sequence;
```

```
use UML::Sequence::PerlSeq;
```

```
use UML::Sequence::Raster;
```

```
use constant USP => 'UML::Sequence::PerlSeq';
```

```
my $m_file = 'test.methods';
```

```
my $script = 'sequence_test.pl';
```

```
my $test = USP->grab_outline_text($m_file,$script);
```

```
my $methods = USP->grab_methods($test);
```

```
my $parse_method = USP->can('parse_signature');
```

```
my $grab_method = USP->can('grab_methods');
```

```
shift @$methods; # wichtig!
```

UML::Sequence – ein Beispiel

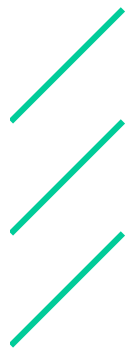
```
my $tree = UML::Sequence->new(
    $methods,
    $test,
    $parse_method,
    $grab_method);

# XML-Datei
my $xml_file = 'test_sequence.xml';
open my $fh, '>', $xml_file or die $!;
print {$fh} $tree->build_xml_sequence('Test');
close $fh;

# Erzeugung der PNG-Datei
my $png_file = 'test_sequence.png';
my @args     = ('-o', $png_file);

seq2raster(@args, $xml_file);
```

UML::Sequence – ein Beispiel



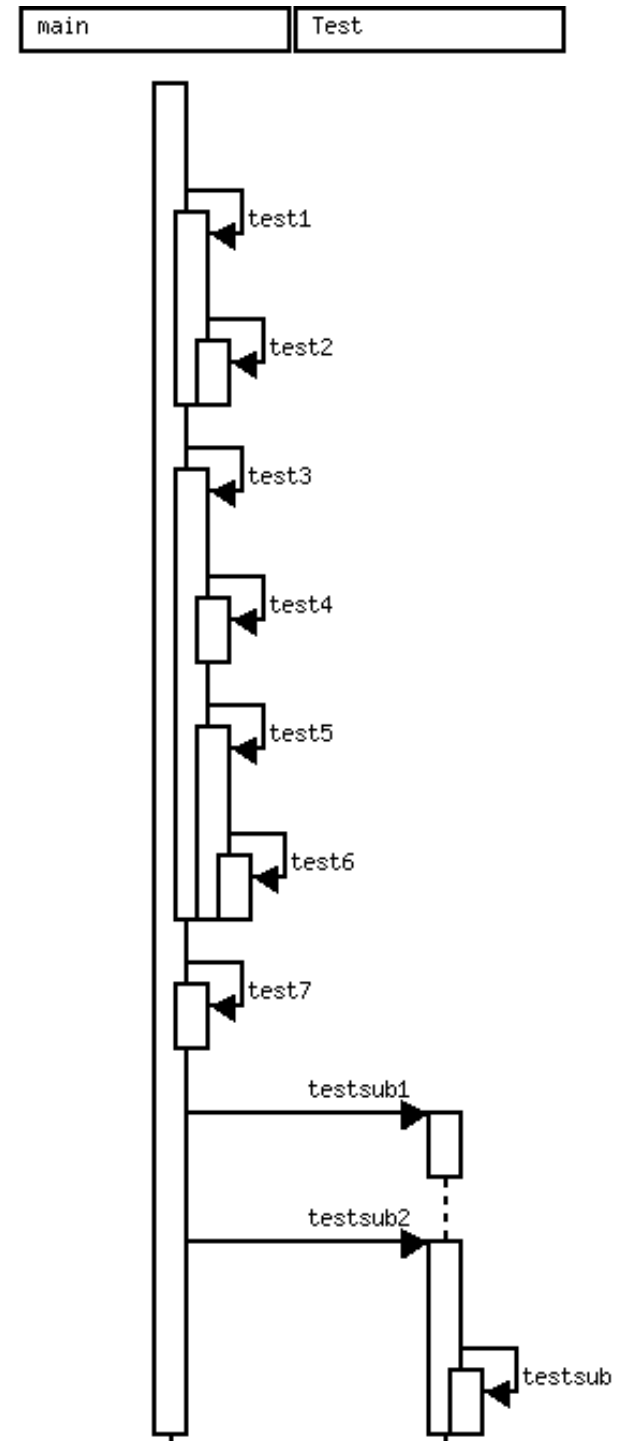
```
Test::testsub1  
Test::testsub2  
Test::testsub3  
Test1  
Test2  
Test3  
Test4  
Test5  
Test6  
test7
```

UML::Sequence

- Bereiche klar getrennt
- Ablauf sauber dargestellt

23.02.2007

UML + Perl = ?



Autodia

- Erzeugt Klassendiagramme
- Parsen mit Heuristiken
- Bestehender Code
- Berücksichtigt nur angegebene Dateien
- Bei Tests Probleme mit Vererbung
- autodia.pl als praktisches Skript

Autodia – Ein Beispiel

```
C:\programs>autodia -i "mytest.pl MyTest.pm MyTest\Child.pm" -z  
getting handlers..  
getting pattern for perl
```

```
AutoDia - version 2.03(c) Copyright 2003 A Trevena
```

```
using language : perl
```

```
..using Autodia::Handler::Perl
```

```
opening mytest.pl
```

```
opening MyTest.pm
```

```
opening MyTest\Child.pm
```

```
Diagram.pm : Inheritances : no Inheritances to be printed -  
ignoring..
```

```
written outfile : autodia.out.xml successfully
```

```
complete. (processed 3 files)
```

Autodia – Ein Beispiel

```
#!/usr/bin/perl

use strict;
use warnings;
use MyTest;
use MyTest::Child;
use NonExistantPackage;

print 'done';
```

```
package MyTest::Child;

use Exporter;
our @ISA = qw(MyTest);

sub child_test{};
```

```
package MyTest;

use vars qw($VERSION);
$VERSION = 0.01;

sub my_test{}
```

Autodia – ein Beispiel

