
Testen mit Perl

Qualitätssicherung für Module

Renée Bäcker, ***Smart-Websolutions***

Tests schreiben...



Mein Programm ...

```
#!/usr/bin/perl

use strict;
use warnings;

my $tmp;

print 'Zahl eingeben: ';
my $number = <STDIN>;

for my $f(1..$number-1){
    next unless (($number/$f) == int($number/$f));
    $tmp += $f;
}

print "Perfekte Zahl ? ", $tmp == $number ? "Ja" : "Nein";
```

... macht was es soll

■ Programm für perfekte Zahlen

- Wenn alle Faktoren bis zur Zahl addiert die Zahl ist
- 6 ist perfekt: $1 + 2 + 3 = 6$
- 8 ist nicht perfekt: $1 + 2 + 4 = 7$

```
C:\community>test_example.pl
```

```
Zahl eingeben: 6
```

```
Perfekte Zahl ? Ja
```

```
C:\community>test_example.pl
```

```
Zahl eingeben: 8
```

```
Perfekte Zahl ? Nein
```

Eine kleine Änderung ...

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

```
use strict;
```

```
use warnings;
```

```
my $tmp;
```

```
while(1){
```

```
    print 'Zahl eingeben: ';
```

```
    my $number = <STDIN>;
```

```
    for my $f(1..$number-1){
```

```
        next unless (($number/$f) == int($number/$f));
```

```
        $tmp += $f;
```

```
    }
```

```
    print "Perfekte Zahl ? ", $tmp == $number ? "Ja" : "Nein";
```

```
    print "\n";
```

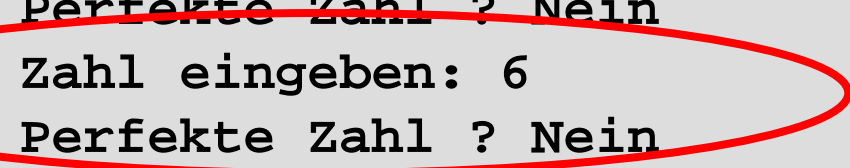
```
}
```

... und es funktioniert noch ...

```
C:\community>test_example.pl  
Zahl eingeben: 6  
Perfekte Zahl ? Ja  
Zahl eingeben: 8  
Perfekte Zahl ? Nein  
Zahl eingeben:
```

... oder auch nicht!

```
C:\community>test_example.pl  
Zahl eingeben: 6  
Perfekte Zahl ? Ja  
Zahl eingeben: 8  
Perfekte Zahl ? Nein  
Zahl eingeben: 6  
Perfekte Zahl ? Nein  
Zahl eingeben:
```





Quelle: www.radgraphics.net

Tests sind wichtig weil...

- Menschen machen Fehler – und wir sind Menschen
- Bugs kosten Zeit & Geld
- Bugs nerven



Tests sind wichtig weil...

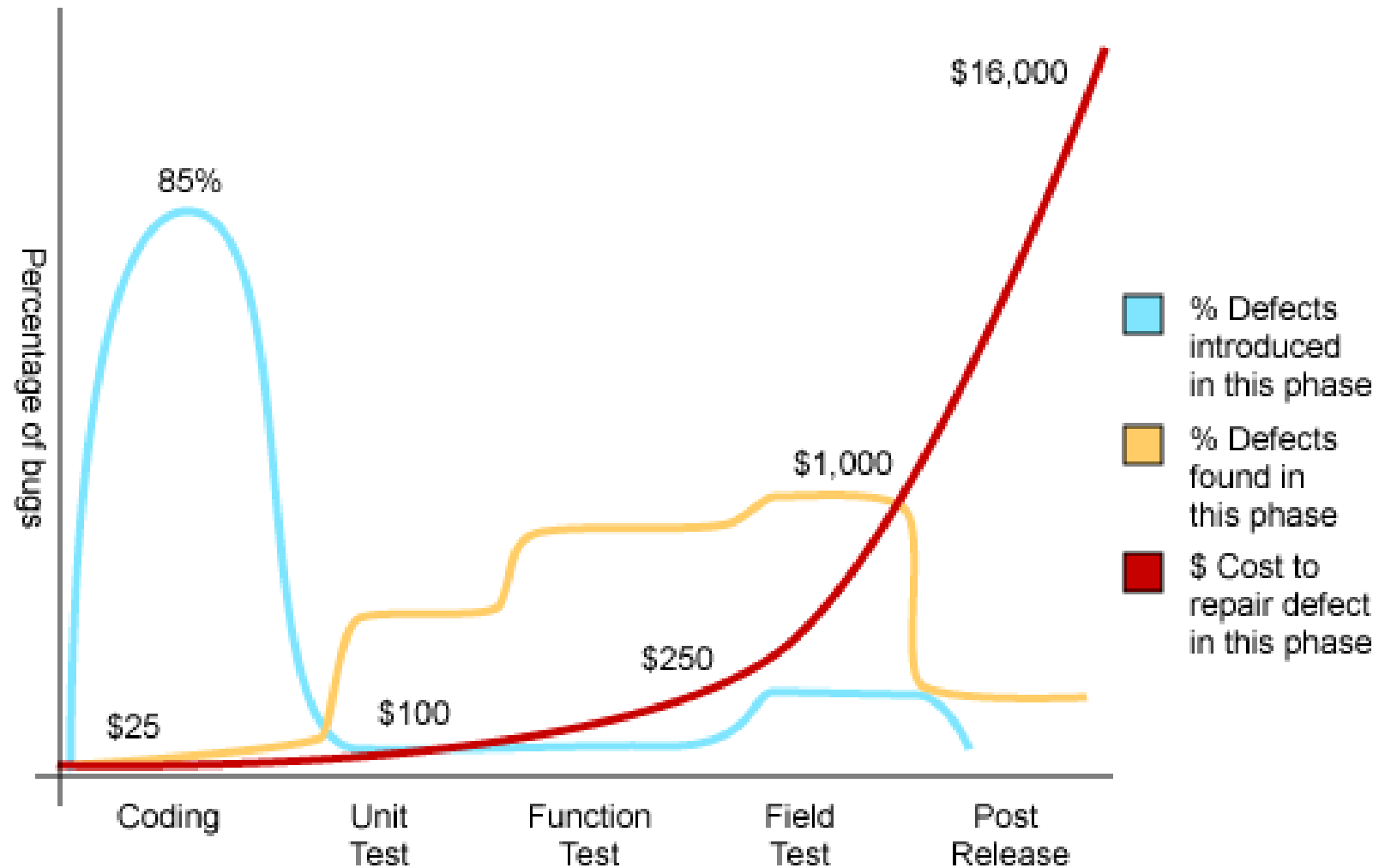
- Erhöhen Qualität des Programms
- Senken die Fehlerwahrscheinlichkeit
- Sichern die Funktionalität des Programms
- ... es das Ranking bei CPANTS erhöht



Tests machen glücklich ;-)



Tests sparen Geld weil...



Source: *Applied Software Measurement*, Capers Jones, 1996

Philosophie der Testautomatisierung

- Es müssen nicht immer 1.000 Tests sein
 - Ein Test ist besser als gar keiner
 - Fang' mit einem Test an
 - Tests können nach und nach erweitert werden
 - Erweitern der Tests
 - Wenn Features hinzugefügt werden
 - Wenn ein Bug entdeckt wird
 - Danach muss der Code neuen Test erfüllen
 - Alter Code darf neuen Test nicht erfüllen
-

Test-driven development

- Tests vor Code
 - Nachträgliche Tests haben Nachteile
 - Beeinflussung der Tests (was wird geprüft)
 - Lückenhafte Tests (Coverage)
 - Erfordert Disziplin
 - Einzelne Testfälle → Testsuite
 - Frühzeitige Gedanken über Design
 - Schnellere Fehlerbehebung möglich
-

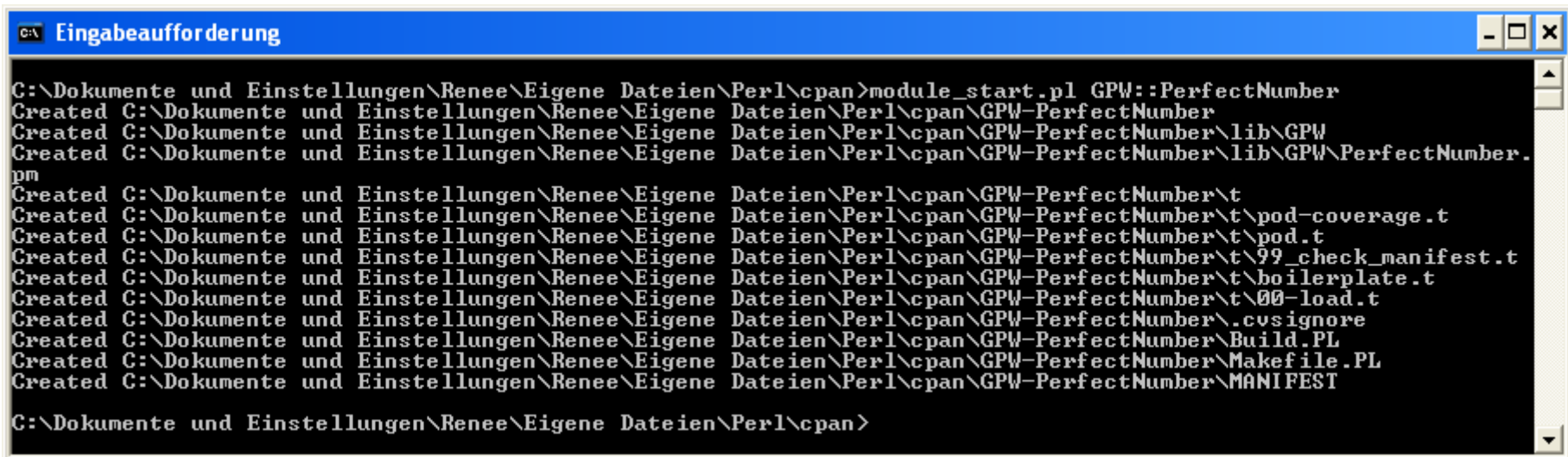
Module für Tests

- Test::Harness
- Test::Simple
- Test::More
- Test::Pod
- Test::Pod::Coverage
- Test::Class
-



Beispiel „perfekte Zahlen“

- Module::Starter Grundgerüst
 - Verzeichnisstruktur
 - Modul-Datei
 - Basis-Testskripte
 - Allgemeine Dateien (README, MANIFEST,...)



```
C:\> Eingabeaufforderung
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan> module_start.pl GPW::PerfectNumber
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\lib\GPW
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\lib\GPW\PerfectNumber.pm
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t\pod-coverage.t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t\pod.t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t\99_check_manifest.t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t\boilerplate.t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\t\00-load.t
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\cvsignore
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\Build.PL
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\Makefile.PL
Created C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber\MANIFEST
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan>
```

Festlegen der Tests und des Designs

- Methode „is_perfect“ überprüft Zahl
 - Rückgabe „1“ wenn perfekte Zahl
 - Rückgabe „0“ wenn nicht perfekte Zahl oder falsche Eingabe
 - Nur natürliche Zahlen
 - Perfekte Zahlen:
 - 6, 28, 496, 8128, 33550336
 - Jede Menge „nicht-perfekte“ Zahlen...
-

Schreiben des Tests (01_is_perfect.t)

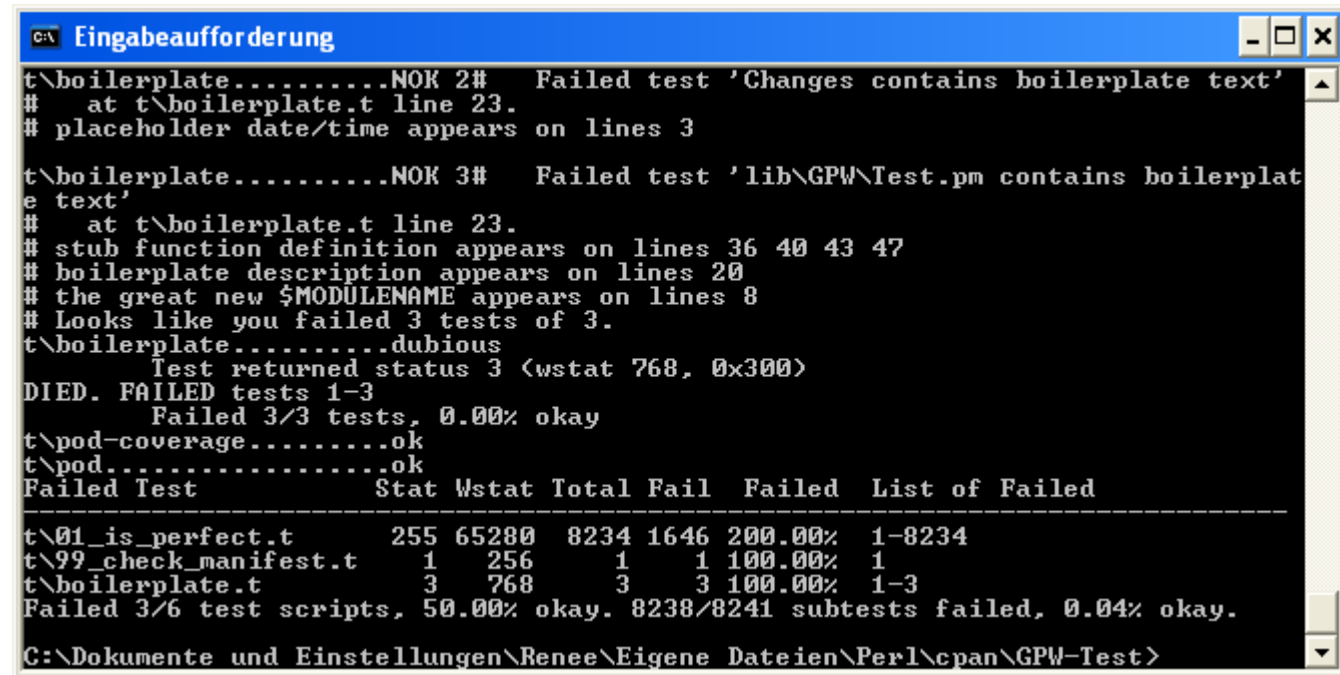
```
#!/usr/bin/perl

use strict;
use warnings;
use Test::More tests => 8234;
use GPW::PerfectNumber qw(is_perfect);

my @array = qw(6
  28
  496
  8128
  33550336);

for my $number(-100..8128,@array){
  my $expected = (grep{ $_ == $number }@array) ? 1 : 0;
  is(is_perfect($number),$expected, "check $number");
}
```

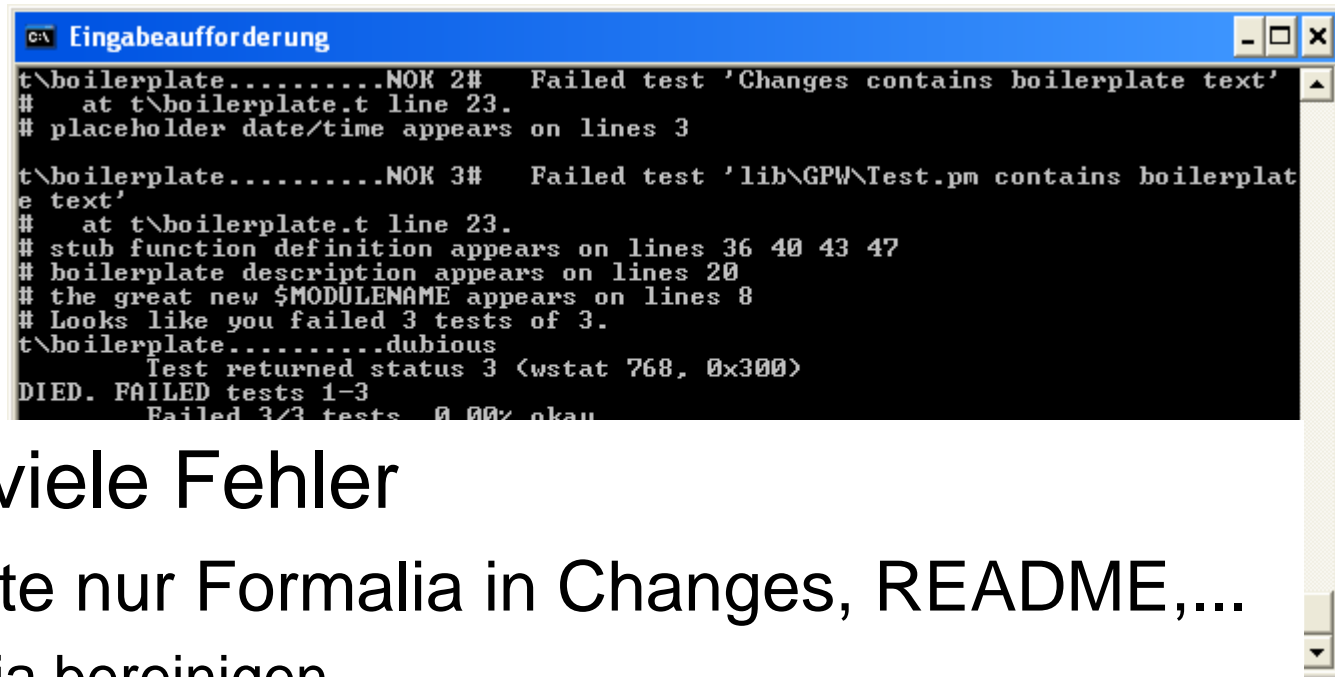
Den Test laufen lassen



```
C:\ Eingabeaufforderung
t\boilerplate.....NOK 2# Failed test 'Changes contains boilerplate text'
# at t\boilerplate.t line 23.
# placeholder date/time appears on lines 3

t\boilerplate.....NOK 3# Failed test 'lib\GPW\Test.pm contains boilerplate
text'
# at t\boilerplate.t line 23.
# stub function definition appears on lines 36 40 43 47
# boilerplate description appears on lines 20
# the great new $MODULENAME appears on lines 8
# Looks like you failed 3 tests of 3.
t\boilerplate.....dubious
Test returned status 3 (wstat 768, 0x300)
DIED. FAILED tests 1-3
Failed 3/3 tests, 0.00% okay
t\pod-coverage.....ok
t\pod.....ok
Failed Test Stat Wstat Total Fail Failed List of Failed
-----
t\01_is_perfect.t 255 65280 8234 1646 200.00% 1-8234
t\99_check_manifest.t 1 256 1 1 100.00% 1
t\boilerplate.t 3 768 3 3 100.00% 1-3
Failed 3/6 test scripts, 50.00% okay. 8238/8241 subtests failed, 0.04% okay.
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-Test>
```

Den Test laufen lassen



```
C:\> t\boilerplate.....NOK 2# Failed test 'Changes contains boilerplate text'
# at t\boilerplate.t line 23.
# placeholder date/time appears on lines 3

t\boilerplate.....NOK 3# Failed test 'lib\GPW\Test.pm contains boilerplate
e text'
# at t\boilerplate.t line 23.
# stub function definition appears on lines 36 40 43 47
# boilerplate description appears on lines 20
# the great new $MODULENAME appears on lines 8
# Looks like you failed 3 tests of 3.
t\boilerplate.....dubious
Test returned status 3 (wstat 768, 0x300)
DIED. FAILED tests 1-3
Failed 3/3 tests, 0.00% okay
```

- Ziemlich viele Fehler
 - boilerplate nur Formalia in Changes, README, ...
 - Formalia bereinigen
 - check_manifest fehlt der Test...
 - Test-Datei in MANIFEST eintragen
 - Tests der Zahlen schlägt fehl

Implementieren der Funktion

```
use base qw(Exporter);
our @EXPORT = qw(is_perfect);

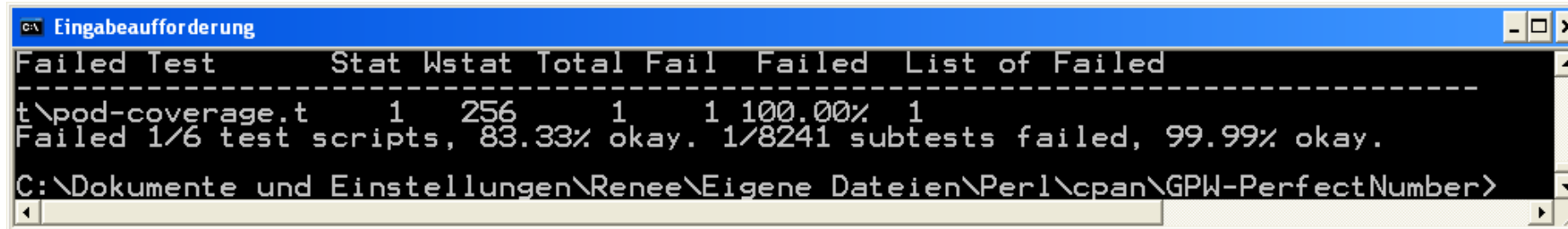
sub is_perfect{
    my ($nr) = @_;
    my $tmp = 0;

    return 0 unless $nr =~ /^[1-9]\d*$/;

    for my $f(1..$nr-1){
        next unless (($nr/$f) == int($nr/$f));
        $tmp += $f;
    }

    return $tmp == $nr ? 1 : 0;
}
```

Test laufen lassen



```
C:\> Eingabeaufforderung
Failed Test      Stat Wstat Total Fail  Failed  List of Failed
-----
t\pod-coverage.t  1    256     1    1 100.00%  1
Failed 1/6 test scripts, 83.33% okay. 1/8241 subtests failed, 99.99% okay.
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber>
```

- Test der Zahlen erfolgreich
- Fehler bei der Doku
 - Doku zu der Funktion fehlt

Neue Funktion hinzufügen

- Eine Funktion „find_perfects“ liefert alle perfekten Zahlen innerhalb eines gegebenen Zahlenbereichs



Test dazu schreiben

```
#!/usr/bin/perl

use strict;
use warnings;
use Test::More tests => 1;
use GPW::PerfectNumber qw(find_perfects);

my $expected = [qw(6 28 496 8128)];

is_deeply(
    [find_perfects(-100, 8128)],
    $expected,
    "check $number");
```

Test ausführen

```
t\02_find_perfects....."find_perfects" is not exported by the
      GPW::PerfectNumber module
Can't continue after import errors at t\02_find_perfects.t line 6
BEGIN failed--compilation aborted at t\02_find_perfects.t line 6.
t\02_find_perfects.....dubious
      Test returned status 255 (wstat 65280, 0xff00)
DIED. FAILED test 1
      Failed 1/1 tests, 0.00% okay
```

- Funktion im Modul einfügen
-

Die Funktion

```
sub find_perfects{
  my ($min,$max) = @_ ;

  my @found;

  while($min++ < $max){
    push @found,$min if is_perfect($min);
  }
}
```

Test ausführen

```
C:\> perl.exe
t\00-load.....ok
t\01_is_perfect.....ok
t\02_find_perfects.....
t\02_find_perfects.....NOK 1# Failed test 'check find_perfects'
#   in t\02_find_perfects.t at line 10.
#   Structures begin differing at:
#     $got->[0] = ''
#     $expected->[0] = '6'
# Looks like you failed 1 test of 1.
t\02_find_perfects.....dubious
Test returned status 1 (wstat 256, 0x100)
DIED. FAILED test 1
Failed 1/1 tests, 0.00% okay
t\99_check_manifest....ok
t\boilerplate.....ok
t\pod-coverage.....ok
t\pod.....ok
Failed Test          Stat Wstat Total Fail  Failed List of Failed
-----
t\02_find_perfects.t    1    256     1     1 100.00% 1
Failed 1/7 test scripts, 85.71% okay. 1/8242 subtests failed, 99.99% okay.
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber>
```

- is_perfect ist schon getestet
 - Fehler kann nur in find_perfect liegen
 - Fehlerbereich sehr eingeschränkt

Die Funktion (geändert)

```
sub find_perfects {
    my ($min, $max) = @_;

    my @found;

    while($min++ < $max) {
        push @found, $min if is_perfect($min);
    }

    return @found;
}
```

Tests starten

```
c:\> Eingabeaufforderung
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber>
.\Build test
Deleting blib\lib\GPW\PerfectNumber.pm
lib\GPW\PerfectNumber.pm -> blib\lib\GPW\PerfectNumber.pm
t\00-load.....ok 1/1# Testing GPW::PerfectNumber 0.01, Perl 5.008008, C
:\usr\bin\perl.exe
t\00-load.....ok
t\01_is_perfect.....ok
t\02_find_perfects.....ok
t\99_check_manifest.....ok
t\boilerplate.....ok
t\pod-coverage.....ok
t\pod.....ok
All tests successful.
Files=7, Tests=8242, 215 wallclock secs ( 0.00 cusr + 0.00 csys = 0.00 CPU)
C:\Dokumente und Einstellungen\Renee\Eigene Dateien\Perl\cpan\GPW-PerfectNumber>
```

- Jetzt läuft alles perfekt

Zusammenfassung

- Kleine Tests sind wenig Aufwand
 - Bei „Test first“
 - Lässt sich Fehlerbereich verkleinern
 - Schnellere Entwicklung
 - Überblick leichter zu behalten
 - Tests erhöhen Qualität

 - Folien und Code unter
<http://renee-baecker.de/vortraege.html>
-