

Warum entwickle ich objektorientiert?

Emotionale Gründe zur Anwendung der objektorientierten Entwicklung

Frank Listing
f.listing@microconsult.com

Was ist objektorientierte Programmierung?

Wie wird der Einsatz der OOP begründet?

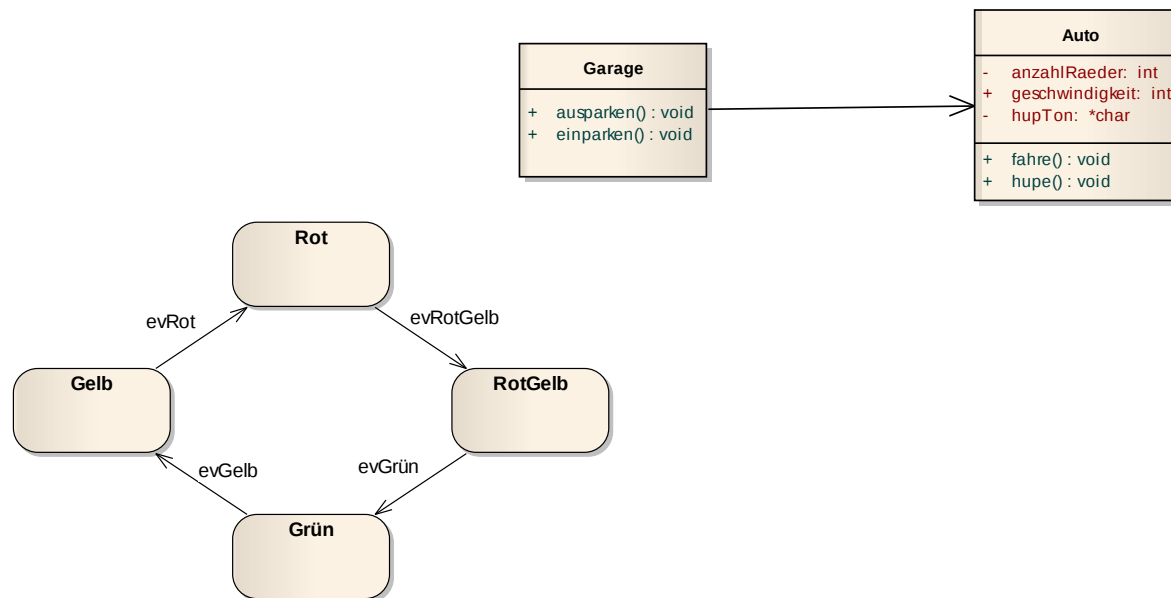
Welches sind meine ganz persönlichen Gründe?

Können diese Gründe verallgemeinert werden?

Objektorientierte Programmierung

- auf dem PC schon fast eine Selbstverständlichkeit
- setzt sich auch im embedded Bereich immer mehr durch
- Umsetzung oft schleppend
- fehlende Akzeptanz bei den Entwicklern

```
class Auto
{
public:
    Auto(void);
    ~Auto(void);
};
```



Problem:

- Einführung der OOP "von oben"
- Entwickler werden zu wenig involviert
- Motivation wird vernachlässigt

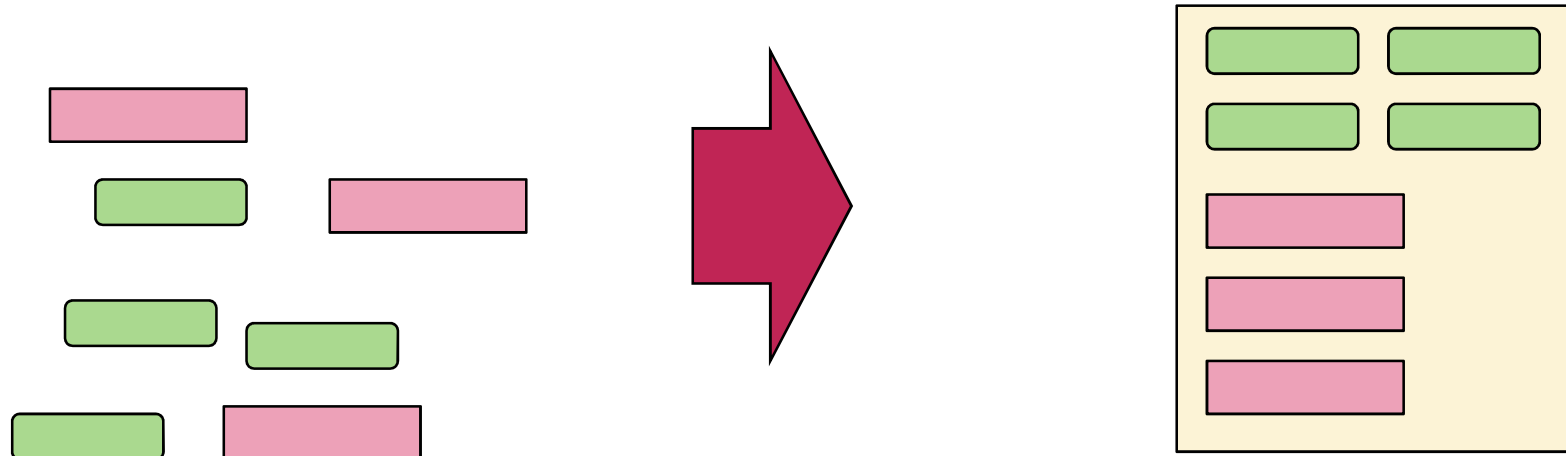
→ Abneigung vieler Entwickler gegen OOP, ohne sich näher damit zu beschäftigen.



Was ist OOP?

Wikipedia:

"Die objektorientierte Programmierung (kurz OOP) ist ein auf dem Konzept der Objektorientierung basierender Programmierstil. Die Grundidee dabei ist, Daten und Funktionen, welche auf diese Daten angewandt werden können, möglichst eng in einem sogenannten Objekt zusammenzufassen und nach außen hin zu kapseln, so dass Methoden fremder Objekte diese Daten nicht versehentlich manipulieren können."



Quelle: http://de.wikipedia.org/wiki/Objektorientierte_Programmierung

Über die Vorteile der OOP sagt Wikipedia nichts. Aber eine Recherche mit einer bekannten Suchmaschine brachte einige Ergebnisse:

- Die Programme sind flexibler, leichter an Veränderungen anzupassen.
- Durch die Aufteilung in Klassen gibt es einen höheren Grad der Wiederverwendung → Zeitersparnis.
- Die Programme sind besser erweiterbar und skalierbar.
- OOP ist leichter zu lernen, da Daten und Funktionalität gemeinsam betrachtet werden.

Mein Lieblingsgrund:

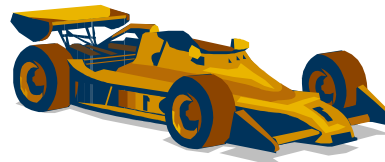
Durch die Verwendung von Polymorphie wird die Wartbarkeit der Software verbessert.

Der Hauptgrund für die Entwicklung der objektorientierten Programmierung ist ein sehr menschlicher und wurde noch nicht erwähnt:

Wir sehen Objekte, wenn wir die Welt betrachten.

Die objektorientierte Vorgehensweise wurde aus den Erfahrungen und dem täglichen Umgang des Menschen entwickelt.

Die Abstraktion geschieht ganz automatisch, wir haben das von Kind an so gelernt.



Die angeführten Gründe, die OOP einzusetzen, klingen plausibel (nicht in jedem Fall verständlich). Aber warum setzen viele Entwickler sie nicht ein?

- Die Steuerung des menschlichen Gehirns erfolgt auf Basis von Emotionen und wurde über Jahrtausende geprägt.
- Die Zeit der technischen Ära ist so kurz, dass sie dabei keinen Einfluss hatte.



Trotz der besser an den Menschen angepassten Vorgehensweise fällt vielen Entwicklern der Umstieg zur OOP schwer.

- Sie haben es anders gelernt - Umlernen ist immer ein schwieriger Prozess.
- Die zu programmierenden Aufgaben kommen auch nicht aus dem normalen menschlichen Umfeld.
- Viele Menschen denken, dass komplizierte Aufgaben nur mit komplizierten Lösungen gelöst werden können.

Aber (menschlich):

Viele Entwickler wissen nicht, was OOP ist, machen es aber trotzdem.

Es gibt die vorgeschobenen sachlichen Gründe, warum die OOP besser ist als die funktionale Programmierung.

Ein wirklich triftiger Grund ist die Orientierung an der menschlichen Wahrnehmung der Welt.

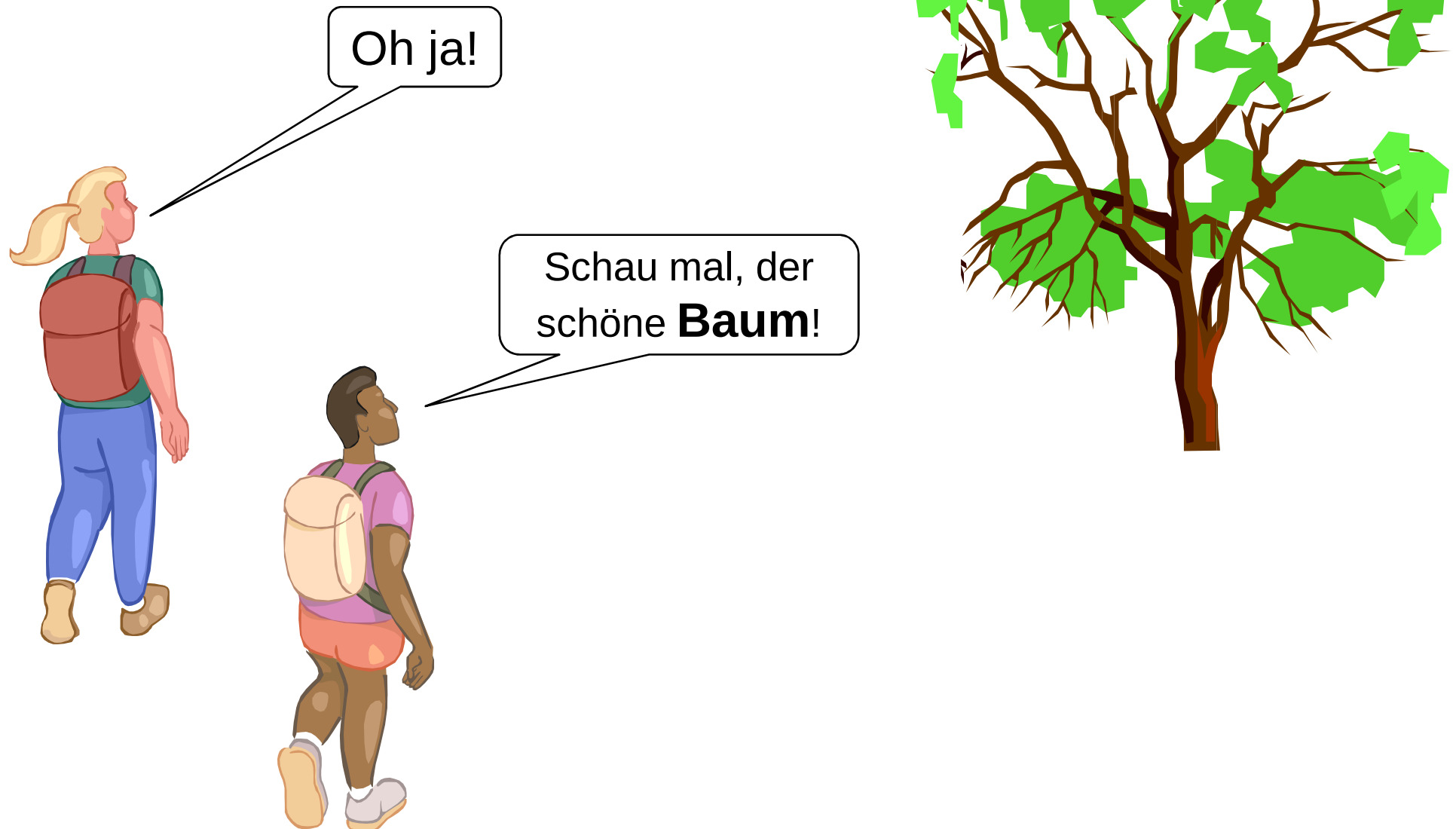
Trotzdem:

Warum entwickle ich objektorientiert?

Eine rein subjektive Sicht auf das Thema OOP:

Ich möchte verstanden werden





Durch die Benutzung von verständlichen Einheiten mit verständlichen Namen (Objekten) anstatt einer Sammlung von vielen Einzelteilen kann ich anderen meinen Code besser erklären.

```
int farbe;  
int richtung;  
int spannung1;  
int spannung2;  
  
void Ein();  
void Start();  
void Aus();  
void Stop();
```

```
class Lampe  
{  
private:  
    int farbe;  
    int spannung;  
  
public:  
    void Ein();  
    void Aus();  
};
```

```
class Motor  
{  
private:  
    int richtung;  
    int spannung;  
  
public:  
    void Start();  
    void Stop();  
};
```

Ich möchte verstehen

"Ich möchte verstanden werden" aus der entgegengesetzten Blickrichtung:

Mit der objektorientierte Vorgehensweise kann ich mir einen groben Überblick über ein System verschaffen.

Ich werde nicht von zu vielen Details erschlagen und kann gezielt tiefer in bestimmte Programmteile einsteigen – die Verbindungen zu anderen Programmteilen sind gut sichtbar.

→ Besseres Verstehen von fremden aber auch eigenem Code.

Ich bin faul

Wenn ich in meinen Code schaue, möchte ich schnell verstehen, was ich dort programmiert habe.

- Einteilung in Objekte
- Verwendung von verständlichen Namen
- Details lenken nicht ab und werden erst bei näherem Hinschauen sichtbar.

Ich möchte nicht immer wieder die notwendigen Informationen mühsam zusammensuchen.



Ich mag Abwechslung

Weniger Langeweile im Entwickleralltag:

- Vermeiden von stupider Routine
- Probleme nur einmal lösen, nicht immer wieder
- Nutzen von Vererbung und Kapselung, um vorhandenen Code wiederzuverwenden, anstatt ihn noch einmal neu zu schreiben
- Zusammenhänge herstellen und begreifen



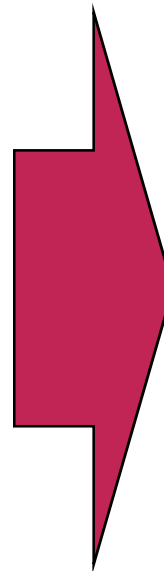
```
extern int g_wert;
```

```
...  
g_wert= 3;  
WertHatSichGeaendert();
```

```
...  
g_wert= 6;  
WertHatSichGeaendert();
```

```
...  
g_wert= 31;  
WertHatSichGeaendert();
```

```
...  
g_wert= 25;  
WertHatSichGeaendert();
```



```
class X  
{  
    ...  
    void SetzeWert(int neu)  
    {  
        m_wert=neu;  
        WertHatSichGeaendert();  
    }  
}
```

```
...  
g_werte.SetzeWert(3);
```

```
...  
g_werte.SetzeWert(6);
```

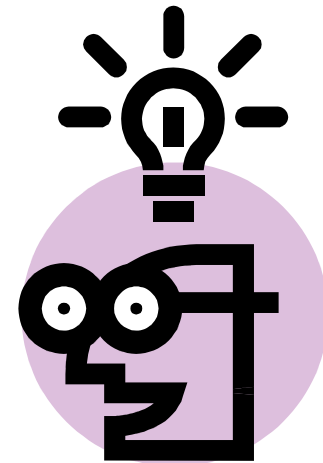
```
...  
g_werte.SetzeWert(31);
```

```
...  
g_werte.SetzeWert(25);
```

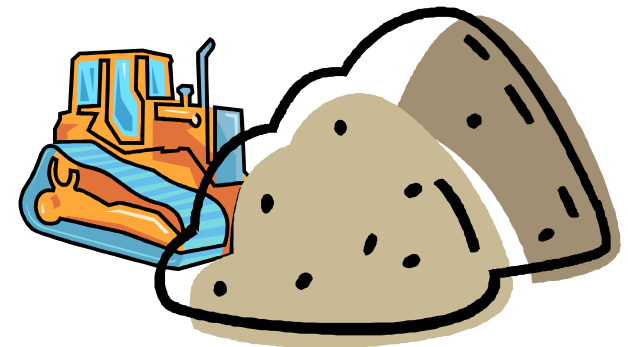
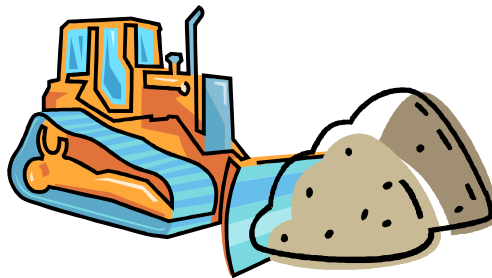
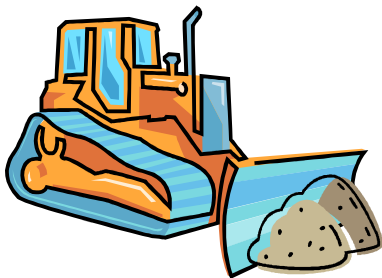
Ich habe lieber kleine Schmerzen als richtig große

Aus meiner Erfahrung heraus weiß ich:

- Am Anfang des Projektes ein wenig unangenehme Arbeit zu erledigen (z.B. Nachdenken) schützt vor sehr viel unangenehmer Arbeit gegen Ende des Projektes.
- Gut durchdachte Schnittstellen erleichtern die Zusammenarbeit mit den Kollegen und erleichtern den Test.
- Eine gute Architektur, die auch Änderungen berücksichtigt, beugt bösen Überraschungen vor. Ihr Kunde hat garantiert kurz vor Abgabe noch eine tolle Idee.



Viele kleine Probleme, die man vor sich herschiebt, werden irgendwann sehr groß.



Ich habe gern Erfolgserlebnisse

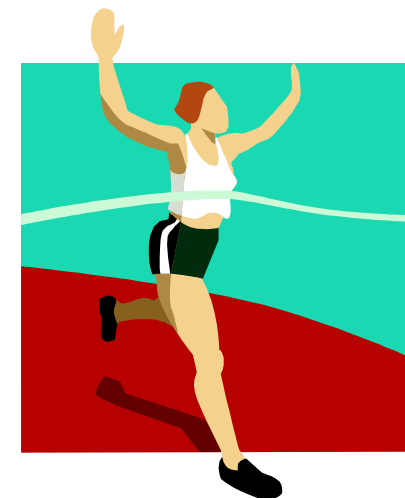
Fertig programmierte Klassen sind Teilerfolge.

Mit Testtreibern und -stubs sind sie meistens gut separat testbar und können damit abgehakt werden.

Nahe kleinere Ziele beflügeln mehr als ein weit entferntes großes.

Mehrere kleinere Etappen als eine große – das nächste Ziel ist in Sichtweite.

Nicht immer ist der Weg das Ziel – Ankommen ist auch schön.



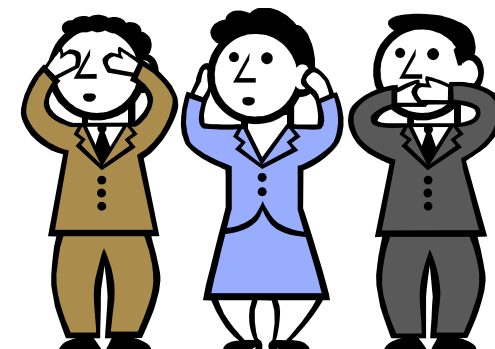
Vorsicht vor Verallgemeinerung!

Das sind **meine** persönlichen Gründe.

Diese Gründe mögen auch bei anderen Entwicklern zutreffen – vielleicht auch nur teilweise – aber Menschen sind zu unterschiedlich, als dass meine Argumentation für jeden zutreffen kann.

Ich möchte verstanden werden:

- Trifft sicher auf viele zu.
- Wenige sehr introvertierte Menschen pflegen aktiv ihr Leid, nicht verstanden zu werden.
- In der Software-Entwicklung wird manchmal auch absichtlich gegen das Verstehen gearbeitet – um sich unabkömmlich zu machen.



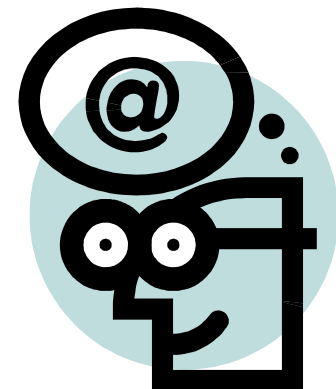
Ich möchte verstehen:

- Wir haben alle unsere Aussetzer.
- Es gibt bei jedem Momente, in denen er mit abgeschaltetem Gehirn durchs Leben geht (oder fährt).
- Oft senken äußere Einflüsse, z.B. drohende Entlassungen, die Motivation.



Ich bin faul:

- Da gibt es wahrscheinlich keine abweichende Meinung.
- Die Faulheit äußert sich auf verschiedene Art.
 - Einer ist zu faul zum Tippen und denkt lieber länger.
 - Ein anderer ist zu faul zum Denken und tippt lieber mehr.



Ich mag Abwechslung:

- Tagein, tagaus immer wieder dasselbe machen gefällt nicht jedem.
- Aber: Der Hamster steigt freiwillig ins Rad.
Viele Menschen machen lieber immer wieder dieselbe Tätigkeit, als über Veränderungen nachzudenken.
- Die Angst vor Veränderung ist oft sehr ausgeprägt.

Ich habe lieber kleine Schmerzen als richtig große:

- Die Natur des Menschen ist sehr unterschiedlich.
 - Einige sind in der Lage, aus schmerzlichen Erfahrungen zu lernen und auf kleine Problemindikatoren zu achten.
 - Bei anderen muss ein Problem erst unüberwindbar werden, ehe es wahrgenommen wird.

Ein gutes Beispiel aus dem Leben ist der Besuch beim Zahnarzt.

- Geht man regelmäßig zur Untersuchung, tut es selten weh.
- Manche Menschen gehen erst, wenn die Schmerzen unerträglich werden – und werden dann natürlich in ihrer Angst bestärkt.



Ich habe gern Erfolgserlebnisse:

Selbst bei einem an sich schönen Grund werden mir nicht alle zustimmen:

Der Märtyrer will leiden.

Ich habe für mich genug Gründe gefunden, weiterhin objektorientiert Software zu entwickeln.

Diese sind nicht immer vernünftig, sie entsprechen aber meinen Bedürfnissen als Individuum und sorgen dafür, dass mir meine Arbeit Spaß macht.