

Warum entwickle ich objektorientiert?

Emotionale Gründe zur Anwendung der objektorientierten Entwicklung

Die objektorientierte Programmierung – auf dem PC schon fast eine Selbstverständlichkeit – setzt sich auch im embedded Bereich immer mehr durch. Allerdings kommt dieser Wandel nicht immer von den Entwicklern selbst, er wird oft einfach „per Order de Mufti“ vom Management angeordnet. Durch fehlende Motivation und die prinzipielle Abneigung gegen Befehle (mögen Software-Entwickler gar nicht) geht die Umstellung auf die objektorientierte Vorgehensweise schleppend voran. Ich möchte mit diesem Artikel aus meiner ganz persönlichen Sicht zeigen, warum objektorientiert denken und entwickeln Vorteile bietet und (mir) Spaß macht.

Vor einiger Zeit wurde ich gefragt, warum ich objektorientiert entwickle. Damals habe ich geantwortet „Das macht man heute so“ – tolle Begründung, damit hätte ich mich auch nicht zufrieden gegeben. Also wurde noch einmal nachgehakt. Trotzdem konnte ich auf Anhieb keine zufriedenstellende Antwort geben. Deshalb habe ich mich mit dem Thema auseinandergesetzt, um herauszubekommen, was mich zur objektorientierten Programmierung gebracht hat.

Zum Einstieg ist es sicher nicht schlecht, erst einmal zu wissen, was objektorientierte Programmierung denn überhaupt ist. In den Zeiten des Internets schaut man erst mal bei Wikipedia nach:

„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.“

Es werden auf der Webseite auch noch einige Feinheiten erklärt, aber über die Vorteile des Einsatzes der OOP sagt Wikipedia nichts. Also habe ich eine allseits bekannte Suchmaschine bemüht, um herauszubekommen, warum andere Entwickler objektorientiert arbeiten. Wie sich herausstellte, bin ich nicht der Einzige, der seine Nöte mit der Erklärung hat. Die Mehrzahl der Treffer führte zu Forenbeiträgen, wo jemand genau diese Frage nach dem Sinn der OOP gestellt hatte. Die Antworten wiesen meistens in die richtige Richtung:

„Die Programme sind flexibler, leichter an Veränderungen anzupassen.“

„Durch die Aufteilung in Klassen gibt es einen höheren Grad der Wiederverwendung, das resultiert in Zeitersparnis.“

„Die Programme sind besser erweiterbar und skalierbar.“

„OOP ist leichter zu lernen, da Daten und Funktionalität gemeinsam betrachtet werden.“

Am besten gefallen hat mir: „Durch die Verwendung von Polymorphie wird die Wartbarkeit der Software verbessert.“ Unbekannte Sachverhalte mit unbekannten Begriffen erklären ist immer überzeugend. Da sieht auch der letzte Entwickler ein, dass er einfach objektorientiert entwickeln muss.

Einige wenige Webseiten wiesen dann auch auf den Kernpunkt der OOP hin: die Orientierung an der menschlichen Wahrnehmung. Wir sehen Objekte, wenn wir die Welt betrachten. Diese Sichtweise wird durch die OOP auf die Entwicklung übertragen.

Da sind so viele vernünftige Gründe für die OOP, das Ganze ist auch noch menschlich, warum arbeiten dann nicht alle so? Die Antwort liegt in der menschlichen Natur: Unser Denken und Handeln wird von Hormonen gesteuert, d.h. Emotionen haben den Vorrang. Unser Gehirn wurde über Jahrtausende geprägt, da haben die paar Jahre technischen Fortschritts noch keine Spuren hinterlassen. Auch die Orientierung der OOP an der menschlichen Denk- und Sichtweise hilft da nicht in jedem Fall. Wer einmal etwas anderes gelernt hat, muss sich wieder umstellen. Auch wenn es hinterher einfacher wird, die Umstellung ist mit Mühe verbunden.

Allerdings muss zur Ehrenrettung vieler Entwickler gesagt werden, dass sie oft bereits objektorientiert entwickeln, ohne es bewusst wahrzunehmen– auch das ist wieder typisch menschlich. Die Erfahrungen aus dem Leben fließen in die Arbeit mit ein.

Ein paar Gründe für den Einsatz der OOP wurden bereits genannt. Allerdings ist für mich der einzige stichhaltige Grund die Orientierung an der menschlichen Wahrnehmung. Davon ausgehend habe ich einige rein subjektive Gründe gefunden, warum ich seit Jahren objektorientiert Software entwickle:

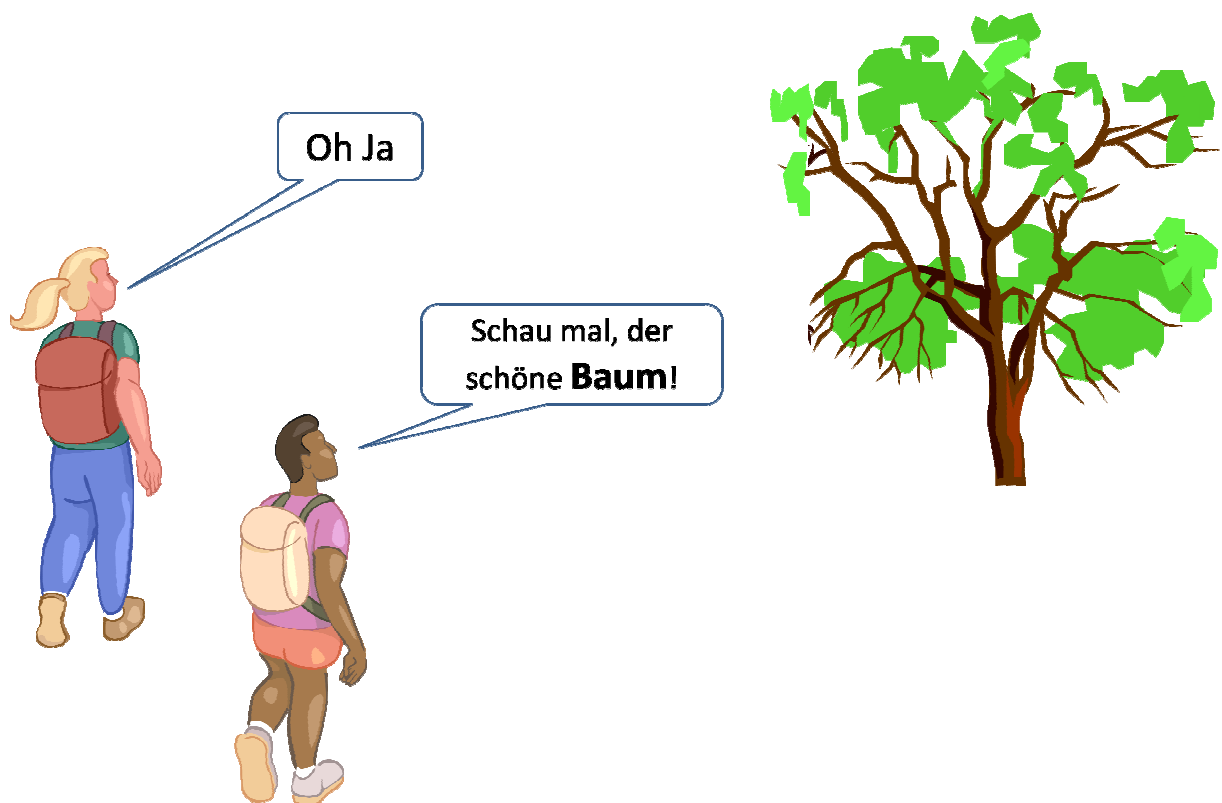
Ich möchte verstanden werden.

Im normalen Leben wie im Beruf: Ich mag es, wenn mein Gegenüber weiß, was ich meine. In meiner Tätigkeit als Dozent ist das sicher besonders wichtig, aber auch als Entwickler sollte man sich mitteilen können. Dazu gehört auch ein verständlicher Code. Ich habe es schon oft erlebt, dass zwei Diskussionspartner lange Zeit aneinander vorbeidiskutiert haben.

Beispiel für nicht-objektorientierte Kommunikation:



Nun das Ganze objektorientiert:



Auch Programmcode wird durch die Benutzung von verständlichen Einheiten mit verständlichen Namen (Objekten) anstatt einer Sammlung vieler Einzelteile besser verständlich.

Ich möchte verstehen.

Das ist der gerade genannte Grund aus der anderen Blickrichtung. Der Wunsch des Verstehens bezieht sich nicht nur auf Code von anderen Entwicklern, der, objektorientiert geschrieben, natürlich auch besser zu verstehen ist. Es ist das eine oder andere Mal auch ganz nett, seinen eigenen Code zu verstehen. Nachdem ich am Anfang meiner Entwicklerlaufbahn meinen vor zwei Wochen geschriebenen Code (der mir bei der Erstellung sonnenklar war) nicht mehr verstanden habe, lernte ich im Laufe der Zeit klare Strukturierung, sprechende Namen und sauber designte Objekte zu schätzen.

Ich bin faul.

Ich möchte schnell verstehen (siehe oben). Auf der anderen Seite möchte ich nicht so viel Code eintippen. Also erst mal in Ruhe nachdenken, ehe die Finger bewegt werden. Sorgfältiges Nachdenken sorgt auch dafür, dass die Finger nicht so oft bewegt werden müssen.

Ich mag Abwechslung.

Stupide Routine liegt mir nicht. Ich möchte Abwechslung im Entwickleralltag. Auch da hilft mir die objektorientierte Vorgehensweise: Durch Vererbung und Kapselung kann ich die Wiederverwendung meines Codes fördern und Doppelimplementierung vermeiden. Mein Ziel ist es, Probleme einmal zu lösen und nicht immer wieder.

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. Also erst nachdenken, mit den Kollegen sprechen und Ergebnisse dokumentieren, dann erst codieren. Solch lästige Sachen, wie Diskussionen mit Kunden und Kollegen oder Dokumentation, schiebt man gern auf die lange Bank. Man kann ihnen aber nicht enttrinnen. Also lieber gleich erledigen, dann tut es nicht so weh. Viele kleine Probleme, die man vor sich herschiebt, werden irgendwann sehr groß.

Ich habe gern Erfolgserlebnisse.

Entwickler werden selten gelobt – meistens gibt es nur einen Tritt, wenn mal etwas nicht geht. Also muss jeder selbst für seine Erfolgserlebnisse sorgen. Ein erledigter Arbeitsabschnitt schafft Zufriedenheit und motiviert. 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 und die Arbeit macht gleich mehr Spaß.

Achtung – 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“ – Das wird im Allgemeinen sicher auf viele von uns zutreffen. Nur wenige sehr introvertierte Menschen pflegen aktiv ihr Leid, nicht verstanden zu werden. In der Software-Entwicklung wird allerdings auch manchmal absichtlich gegen das Verstehen gearbeitet – um sich unabkömmlich zu machen. Diese Unsitte ist in den letzten Jahren zwar etwas zurückgegangen, aber man begegnet ihr leider immer noch.

„Ich möchte verstehen“ – Da haben wir alle unsere Aussetzer. Es gibt bei jedem Momente, in denen er mit abgeschaltetem Gehirn durchs Leben geht (oder fährt). Oft sind es äußere Einflüsse, z.B. drohende Entlassungen, die die Motivation drastisch senken. In solchen Fällen wird das Interesse, Software zu verstehen, auf Sparflamme zurückgeschraubt.

„Ich bin faul“ – Da gibt es wohl keine abweichende Meinung. Das Problem ist, dass sich die Faulheit auf verschiedene Art äußert. Der eine ist zu faul zum Tippen und denkt lieber länger, der andere ist zu faul zum Denken und tippt lieber mehr. Oder es wird aus Faulheit nicht sauber dokumentiert (das kostet mich auch immer viel Überwindung).

„Ich mag Abwechslung“ – Tagein, tagaus immer wieder dasselbe machen gefällt nicht jedem – aber der Hamster steigt freiwillig ins Rad. Viele Menschen führen lieber immer wieder dieselbe Tätigkeit aus (auch wenn sie keinen Spaß macht), als über Veränderungen nachzudenken. Die Angst vor Veränderung ist oft sehr ausgeprägt.

„Ich habe lieber kleine Schmerzen als richtig große“ – Hier ist die Natur des Menschen sehr unterschiedlich. Einige sind in der Lage, aus schmerzlichen Erfahrungen zu lernen und auf kleine Problemindikatoren zu achten bzw. vorausschauend zu denken, bei anderen muss ein Problem erst unüberwindbar werden, ehe es wahrgenommen wird. Ein gutes Beispiel aus dem Leben ist der Zahnarztbesuch. Geht man regelmäßig zur Untersuchung, sind nur sehr kleine Schäden zu beheben, und es tut nicht weh. Manche Menschen haben eine so große Angst vor dem Zahnarzt, dass sie erst gehen, 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.

Jeder Entwickler ist für die objektorientierte Entwicklung geeignet. Er muss nur an der richtigen Stelle eingesetzt werden. Nicht jeder kann und will Systemarchitekt sein. Innerhalb von Objekten gibt es immer noch sequentielle Programmierung und gerade in der embedded Programmierung auch genug Bit-Pfrieemelei für die, die das mögen. OOP-Muffel werden meistens nur an der falschen Stelle eingesetzt.

Fazit

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.

Quellen:

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

Autorenprofil:

Dipl.-Ing. **Frank Listing** ist seit 2002 Trainer und Projektcoach bei der MicroConsult GmbH mit dem Schwerpunkt Microsoft-Plattformen, objektorientierte Programmierung und Testen von Embedded Systemen und u.a. fachlich für das Thema .NET verantwortlich. Sein Wissen gibt er immer wieder auch in Publikationen und Fachvorträgen weiter.

