Paradise lost

Über die beschwerliche Existenz von Softwareentwicklern ließe sich so manche Anekdote erzählen, wäre das Thema nicht so betrüblich. Eingeklemmt zwischen Chaostheorie und Unschärferelation, fristen Entwickler ein Dasein am Rande der Gesellschaft. Obwohl die Medien neuen Errungenschaften, wie dem iPhone5 oder Windows 8, mit überschwänglichem Enthusiasmus begegnen, stehen gerade diejenigen nicht im Rampenlicht, die als Softwareentwickler – im Schweiße ihres Angesichts – derartige Gadgets erst zum Leben erwecken. Dabei ist Softwareentwicklung kein Zuckerschlecken, sondern eher mit Qualen Sisyphus'schen Ausmaßes verbunden.

Es kann daher nicht verwundern, dass einige berühmte Persönlichkeiten sich schon intensiv mit dieser Problematik beschäftigt haben. Dem unvergesslichen E. W. Dijkstra zufolge besteht die primäre Herausforderung eines Informatikers letztendlich darin, sich nicht von der Komplexität der eigenen Machwerke verwirren zu lassen. Und auch C. A. R. Hoare schlägt in dieselbe Kerbe, wenn er meint: „Es gibt zwei Wege, einen Softwareentwurf zu erzeugen. Einer besteht darin, ihn so einfach zu gestalten, dass es offensichtlich keine Mängel gibt, und der andere darin, den Entwurf so komplex zu strukturieren, dass es keine offensichtlichen Mängel gibt." Leider neigen Entwicklungsprojekte eher zum letzteren Extrem, wie wir alle wissen.

Als Beleg für diese Aussage möchte ich Bjarne Stroustrup, den Schöpfer von C++, zu Worte kommen lassen: „Ich habe mir immer gewünscht, dass sich Computer so einfach benutzen lassen wie Telefone. Mein Wunsch hat sich erfüllt. Ich weiß nicht länger, wie ich mein Telefon bedienen muss." Aus dem Munde von Bill Clinton war in diesem Zusammenhang zu vernehmen, dass „in Anbetracht des augenblicklich traurigen Zustands unserer Softwareprogramme, Softwareentwicklung eher einer schwarzen Kunst gleicht als einer Ingenieursdisziplin". Eine Lösung hierfür könnte freilich sein, dass die Informatik mehr Genies heranzüchtet, um solche Probleme zu vermeiden – also ganz frei nach Einstein: „Experten lösen Probleme; Genies vermeiden sie."

Leider erweisen sich auch produktivere Programmiersprachen wie Java nicht als hilfreich, denn „echte Programmierer können in jeder Programmiersprache Assemblercode schreiben." – Larry Wall. Und das alles, obwohl wir doch immer die Hoffnung hegten, mit besseren Werkzeugen und Automatisierung ließe sich das Problem in den Griff bekommen. Nur leider schlägt an dieser Stelle das Booch'sche Gesetz zu, das da lautet: „a fool with a tool is still a fool".

Obwohl wir also noch nicht einmal einzelne Applikationen effektiv und effizient entwickeln können, träumt unsere Disziplin bereits von systematischer Wiederverwendung mittels Plattformen und Produktlinien. Ralph Johnson hat aber recht, wenn er artikuliert: „Bevor Software wiederverwendbar ist, muss sie erst einmal verwendbar sein."

Können wir die Qualitätsproblematik also systematisch lösen? Natürlich darf an dieser Stelle das berühmte Zitat von Lenin nicht fehlen: „Vertrauen ist gut, Kontrolle ist besser." Ein Weg sind sicherlich regelmäßige Reviews, denn schon der Philosoph und Theologe Kierkegaard bemerkte treffend: „Das Leben lässt sich nur rückwärts verstehen, muss aber vorwärts gelebt werden."

Und gerade im agilen Zeitalter sollte beim Thema Qualität schon reflexartig das Stichwort „Testen" aus dem Munde jedes Entwicklers schallen. Leider stellen für manche Zeitgenossen Testaktivitäten eher böhmische Dörfer dar. Warum sonst hätte sich Testguru James Bach zu der Aussage hinreißen lassen: „Testen ist der Prozess, das Unsichtbare mit dem Mehrdeutigen zu vergleichen, um zu vermeiden, dass dem Anonymen das Undenkbare passiert."

Unverbesserlichen Formalfetischisten sei an dieser Stelle entgegengeschmettert, dass auch Programmverifikation kein Ersatz für ausgiebige Tests sein kann, was uns kein geringerer als Donald Knuth ins Stammbuch geschrieben hat: „Der obige Code ist mit Vorsicht zu genießen; ich habe zwar bewiesen, dass er korrekt ist, ihn aber nicht getestet."

Wir sollten auch dem Thema Anforderungen die gebührende Aufmerksamkeit widmen, denn es gilt nach Cem Kaner: „Eine exzellente Softwareentwicklung auf Basis lausiger Anforderungen resultiert in einer lausigen Applikation." Da nützen auch die besten Entwicklungsgurus wenig. Schließlich sieht Niklaus Wirth uns nicht zuletzt aufgrund der Benutzerwünsche in der Komplexitätsfalle, zumal „ein primärer Grund für Komplexität darin besteht, dass Softwarehersteller unkritisch jedes Feature integrieren, welches Benutzer wünschen".

Nicht zu vergessen ist freilich an dieser Stelle auch die unterschätzte Bedeutung von sozialen Fähigkeiten. Alistair Cockburn betrachtet Softwareentwicklung folgerichtig als „collaborative Game". Und Gerald Weinberg meinte nicht zu Unrecht: „Jedes Problem ist im Endeffekt ein Kommunikationsproblem."

Schließen möchte ich diese selbstkritische Reflexion mit einer guten Nachricht für alle Produktivitätsfanatiker: Es gibt, im Gegensatz zur Behauptung von Fred Brooks, doch eine Silberkugel („Silver Bullet") für die Softwareentwicklung. Die Kunst besteht darin, die richtige Person zu erschießen.

In diesem Sinne viel Spaß mit der vorliegenden Ausgabe

Ihr Prof. Dr. Michael Stal