Auch morgen noch

Nein, er hatte es nicht studiert.
Er war Ägyptologe und über Umwege
zur Softwareentwicklung gestoßen.

Faszinierend fand er die Welt der Bits und Bytes schon immer, und über die Programmierung, die er sich selbst angeeignet hatte, konnte er wenigstens in einem Lebensbereich detailliert bestimmen, was passieren soll. „Du kannst das doch, das mit Computern“, war der Startschuss für seinen ersten Auftrag für eine kleine selbstgeschriebene Software. So fing alles an.

In seiner langen Tätigkeit als Softwareentwickler hatte er dann immer wieder bemerkt, dass er einige Dinge besser konnte als die, die Informatik von der Pike auf erlernt hatten. Er hatte ein Gespür für Code, und je mehr er sich mit der Thematik auseinandersetzte, desto mehr stellte er fest, dass er instinktiv Architektur, Abläufe und Routinen richtig aufsetzte. Sein Maßstab war die logische Umsetzung von Vorgängen.

Frei nach dem Motto:
Morgen muss ich den Code immer noch genauso gut verstehen wie heute.

Die Denkstrukturen, die heute zum Code geführt haben, sollten so fest gefügt sein, dass sie auch morgen noch zum gleichen Code führen würden. Wie bei einer Akten- oder Dateiablage: Die Logik muss immer den gleichen Ordner finden – ob heute, morgen oder übermorgen.

Um das zu erreichen, ging er immer von den Daten aus: Daten einlesen, Daten verarbeiten, Daten ausgeben. Damit war zu jeder Zeit klar, was das Programm tat. So konnte er nicht nur einen Konverter für XML-Dateien, den er vor vielen Jahren geschrieben hatte, noch einwandfrei verstehen. Nein, sogar Kollegen waren in der Lage, seine Software zu lesen und nachzuvollziehen. Vielleicht war diese Sichtweise genau die, die Menschen am nächsten lag.

Statt mit komplexen Klassen, die tief drin irgendwelche Eigenschaften veränderten, programmierte er mit Funktionen, die Daten verarbeiteten. Parameter gaben die Daten herein, der Rückgabewert war das Ergebnis. Diese einfache Struktur, die schon seit Beginn der Softwareentwicklung in Programmiersprachen zur Verfügung stand, schien die zu sein, die die meisten verstehen konnten.

Inspiriert durch einen Vortrag von Brandon Rhodes [1], der auch für .NET-Entwickler ein wunderbarer Denkanstoß sein kann.

Viel Spaß mit der dotnetpro

Tilman Börner
Chefredakteur dotnetpro

[1] Brandon Rhodes, The Clean Architecture in Python, www.dotnetpro.de/SL2102Edi1

Jan Tittel

informiert über Aus- und Weiterbildung für Softwareentwickler (S. 15, 20)

Christian Havel

berichtet über seine Erfahrungen bei der Einführung von Event Storming (S. 28)

Daniel Weber

überträgt das bewährte OR-Mapper-Konzept auf Graphdatenbanken (S. 85)