Von Generation zu Generation

Am Anfang war der Code und der Code war in Assembler geschrieben, wenn man die Veteranen der Programmierung vor dem Computerzeitalter von Ada Lovelace bis zu Konrad Zuse außen vor lässt. Schnell kamen die ersten höheren Sprachen wie FORTRAN und COBOL auf, später BASIC und C. Und immer wieder findet ein Generationswechsel statt, sei es durch neue Konzepte wie die objektorientierte Programmierung unter anderem in C++ oder ein neues Umfeld wie das Internet, das die Tür für JavaScript öffnete.

Die Welt der Softwareentwicklung ist im steten Wandel. Die heute dominierenden Sprachen Java, Python, JavaScript und C# wurden vor dreißig Jahren noch nicht an den Unis unterrichtet. Kaum ein Developer wird über das gesamte Berufsleben bei denselben Sprachen bleiben. Und das ist gut so. Neue Patterns bringen frische Konzepte und öffnen die Tür für neue Sprachen. Die prozedurale Programmierung und später die Objektorientierung befreiten uns vom Spaghetti-Code, die Java Virtual Machine vereinfachte die plattformübergreifende Programmierung und der Garbage Collector verhinderte typische Speicherfehler.

Wer auf eine neue Sprache wechseln will, muss den richtigen Zeitpunkt abpassen, den Sweetspot im Hype-Zyklus, wenn das Tal der Enttäuschung durchschritten und das Plateau der Produktivität oder zumindest der Pfad der Erleuchtung erreicht ist.

Vier Sprachen haben aus unserer Sicht gute Chancen, die nächste Generation zu bilden: TypeScript, Kotlin, Rust und Go. Sie sind reif genug und haben unterschiedliche geistige Vorgänger, die sie um eigene Konzepte erweitern: TypeScript bringt Typsicherheit zu JavaScript, Kotlin vermischt funktionale Konzepte mit objektorientierter Programmierung auf der JVM. Gegenüber C bringt das Ownership-Konzept von Rust Speichersicherheit ohne den Overhead eines Garbage Collector, und Go zielt mit Blick auf Cloud-Computing und Anwendungen im Cluster auf nebenläufige Programmierung. Und das ist nur die Oberfläche der Neuerungen der Next Generation.

Anders als beim Turmbau zu Babel ist die Sprachenvielfalt in der Softwareentwicklung kein Nachteil. Projekte lassen sich sprachübergreifend umsetzen und die meisten Programmiersprachen bieten Konzepte für das geschmeidige Zusammenspiel. Damit kommt es nicht zu der babylonischen Sprachverwirrung, die seinerzeit verhinderte, dass die übermütigen Menschen den Turm bis in den Himmel bauten. Die Spitze des Programmiersprachenturms ist noch nicht erreicht. Go, Rust, Kotlin und TypeScript entstanden in den Jahren 2009 bis 2012 und sind gut ein Jahrzehnt gereift. Welche Sprachen die nächste Generation der 2030er-Jahre bilden, wird sich zeigen.

RAINALD MENGE-SONNENTAG