

Auf dem Weg zu idealen Programmierwerkzeugen – Bestandsaufnahme und Ausblick

Johannes Brauer · Christoph Crasemann · Hartmut Krasemann

Einleitung

Geprägt durch jahrzehntelange Erfahrungen in der Praxis der Entwicklung und dem Betrieb großer Softwaresysteme diskutieren die Autoren, inwieweit die nach wie vor mannigfaltigen Probleme der Produkt- und Prozessqualität bei Software ihre Ursache in den Programmierwerkzeugen haben. Die Programmiersprache als das grundlegendste Werkzeug steht dabei zwar im Vordergrund, aber für den effektiven Einsatz einer Sprache in der Praxis sind Entwicklungsumgebung und Laufzeitsystem ebenso erforderlich.

Wenn im Zusammenhang mit Problemen oder Unzulänglichkeiten von Programmiersprachen häufiger Java zitiert wird, geschieht dies stellvertretend für alle industriell relevanten Sprachen, wie z.B. C#, C++ oder auch COBOL.

Die Autoren glauben nicht, dass die in der Praxis so „populäre“ Programmiersprache Java einen Endpunkt der Entwicklung dieses Teilgebiets der Informatik markiert und nur noch an der Weiterentwicklung dieser Sprache gearbeitet wird. Im Gegenteil befinden sich die gängigen Sprachen u.E. eher auf toten Ästen des Programmiersprachenstammbaums, die möglicherweise längst abgebrochen wären, wenn sie nicht durch immer neue „Stützungsmaßnahmen“ daran gehindert würden. So könnte z.B. der stete Strom neuer Rahmenwerke und Werkzeuge, die offenbar erforderlich sind, um mit Java produktiv arbeiten zu können, als Indiz für die Unreife dieser Sprache angesehen werden. Diese Entwicklung führt zur Steigerung der Kompliziertheit [1] von Java-basierten Systemen und damit auch zu einer zunehmenden Steilheit der Lernkurve, was

die Praxistauglichkeit eher abnehmen lässt. Auch wenn sich z.B. „Eclipse“ heute symbiotisch anbietet, zahlreiche Werkzeuge für die Java-Entwicklung zu integrieren, so ist doch zu beobachten, dass nur noch ganz wenige Entwickler mit dieser Kompliziertheit souverän und zielführend umgehen können.

In der „Programmierung im Großen“ fehlt nach wie vor, Anwendungen durch „Zusammenstecken“ wiederverwendbarer Komponenten entwickeln zu können. Das liegt u.a. daran, dass es keine normierten Schnittstellenspezifikationen gibt, die diesen Namen verdienen. Meist wird darunter nur die Festlegung der Aufrufsyntax von Diensten verstanden, während deren Semantik nicht präzise definiert werden kann. Dies wird auch an den Standards für Web-Services – wie z.B. WSDL – deutlich, die lediglich die Aufrufsyntax und die Konnektivität betreffen, aber nichts über die Semantik der Dienste auszudrücken erlauben (vgl. auch [12]).

In der „Programmierung im Kleinen“ behindern unsere Programmiersprachen oft den Entwickler, z.B. dadurch dass sie ihn zwingen,

DOI 10.1007/s00287-007-0211-3
© Springer-Verlag 2007

Prof. Dr. Johannes Brauer
FB Informatik, NORDAKADEMIE Hochschule der Wirtschaft,
Köllner Chaussee 11, 25337 Elmshorn
E-Mail: brauer@nordakademie.de

Dr. Christoph Crasemann
IC&C GmbH,
Papenhöhe 14, 25335 Elmshorn
E-Mail: christoph.crasemann@icc-gmbh.de

Dr. Hartmut Krasemann
IT-Architekt,
Königsberger Str. 41c, 22869 Schenefeld
E-Mail: krasemann@acm.org