



Agile

Agile Softwareentwicklung hat sich als Stand der Technik weitgehend durchgesetzt – kann aber auch schiefgehen. Eine dem jeweiligen Zweck angepasste Auswahl an Methoden, die richtigen Werkzeuge für verteilte Retrospektiven und solides Schätzen des Aufwands weisen den Weg zum Erfolg.

ab Seite 7

Qualitätssicherung

Künstliche Intelligenz unterstützt Entwicklerinnen und Entwickler in immer mehr Bereichen vom Schreiben des Codes bis zu dessen Analyse. Mit KI aufgerüstete Entwicklungsumgebungen und Unit-Testing-Tools erleichtern den Programmieralltag. Automatisierte Qualitätskontrolle gelingt aber auch ohne KI.

ab Seite 31



Agile

Werkzeuge: Die passende agile Methode finden	8
Marktübersicht: Tools für verteilte Retrospektiven	12
Agile Entwicklung: Gutes Schätzen geht auch remote	23
Schöner scheitern: Auf welche Arten Agilität schiefgehen kann	27

Qualitätssicherung

Künstliche Intelligenz in der Softwareentwicklung	32
KI-gestütztes Entwickeln mit IntelliJ IDEA, Visual Studio IntelliCode und Tabnine	36
Frameworks für verhaltensgetriebenes Testen	42
Künstliche Intelligenz im Unit-Testing	52
Marktübersicht: Werkzeuge zur automatischen Codeanalyse	58
End-to-End-Tests für Web-Frontends	66
Container-Images: Abschied vom Dockerfile	70
Sichere Software entwickeln mit OWASP SAMM	80

Programmiersprachen

Was Rust hat, das andere nicht haben	86
Rust-Tutorial	
Teil 1: Sprachkonstrukte, Ownership und asynchrone Programmierung	88
Teil 2: Parallele Programmierung, Speicherverwaltung und Crates	92

Tutorial Clean Code mit C++20	
Teil 1: Effizientere Vergleiche	96
Teil 2: Code lesbarer gestalten	99
Teil 3: Weitere Features zur Codeoptimierung	102

GitOps und Containerisierung

GitOps läutet die Ära des automatisierten IT-Betriebs ein	106
GitOps in der Praxis	113
Optimierung Container-basierter Java-Anwendungen	118
Continuous Deployment: Kubernetes in der GitOps-Welt	122

Extras

Status quo komponentenbasierter Softwaretechnik	130
Why Reactive: Reaktive Architekturen und ihre Geschichte	134
Pragmatische Küchentricks für RESTful HAL APIs	139
Bytecode im Browser: Mit WebAssembly und Rust zur Web-Anwendung	144
Value Design als Leitlinie moderner Softwareentwicklung	148
Copyright in der Softwareentwicklung	152

Sonstiges

Editorial	3
Impressum	133



Programmiersprachen

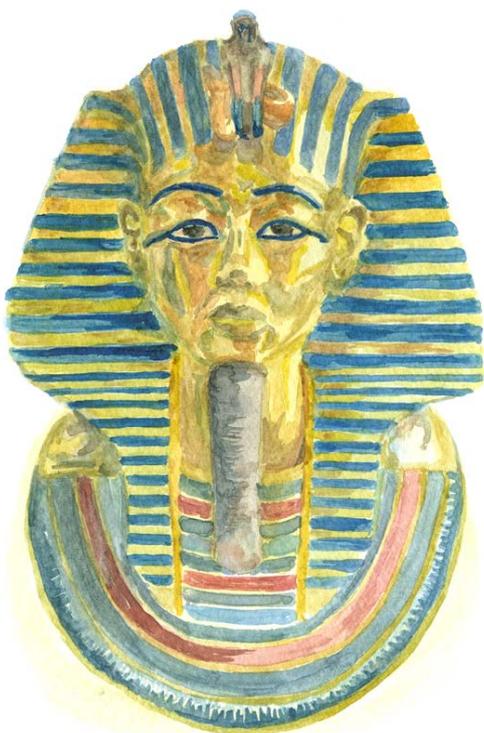
Im schier unüberschaubaren Programmiersprachenwald tritt Rust an, einiges besser zu machen als etablierte Sprachen, beispielsweise durch sicherere Speicherverwaltung und Codeprüfung zur Kompilierzeit. Mit den Prinzipien des Clean Code lässt sich aber auch C++20 optimieren und auf das Wesentliche reduzieren, um den Code lesbarer zu gestalten und Fehler zu vermeiden.

ab Seite 85

GitOps und Containerisierung

Als neues Paradigma will GitOps die Ära des automatisierten IT-Betriebs einläuten. Von der Infrastructure as Code bis zum vollständigen Lebenszyklus einer Anwendung soll mehr Automatisierung helfen, typische Fehler zu vermeiden, und so zu höherer Qualität beitragen. Das macht ein Kubernetes-Praxisbeispiel aus der Versicherungsbranche ebenso deutlich wie ein Tutorial zur Optimierung containerisierter Java-Anwendungen.

ab Seite 105



Extras

Mit Value Design lassen sich moderne Softwareentwicklung und Ethik in Einklang bringen. In Cloud-nativen Architekturen helfen die aus dem Reactive Manifesto abgeleiteten Prinzipien, die hohen Anforderungen an Elastizität und Resilienz zu erfüllen. Für das Arbeiten mit RESTful APIs existieren viele Ansätze. Wie es mit der Hypertext Application Language geht, zeigt ein Erfahrungsbericht.

ab Seite 129