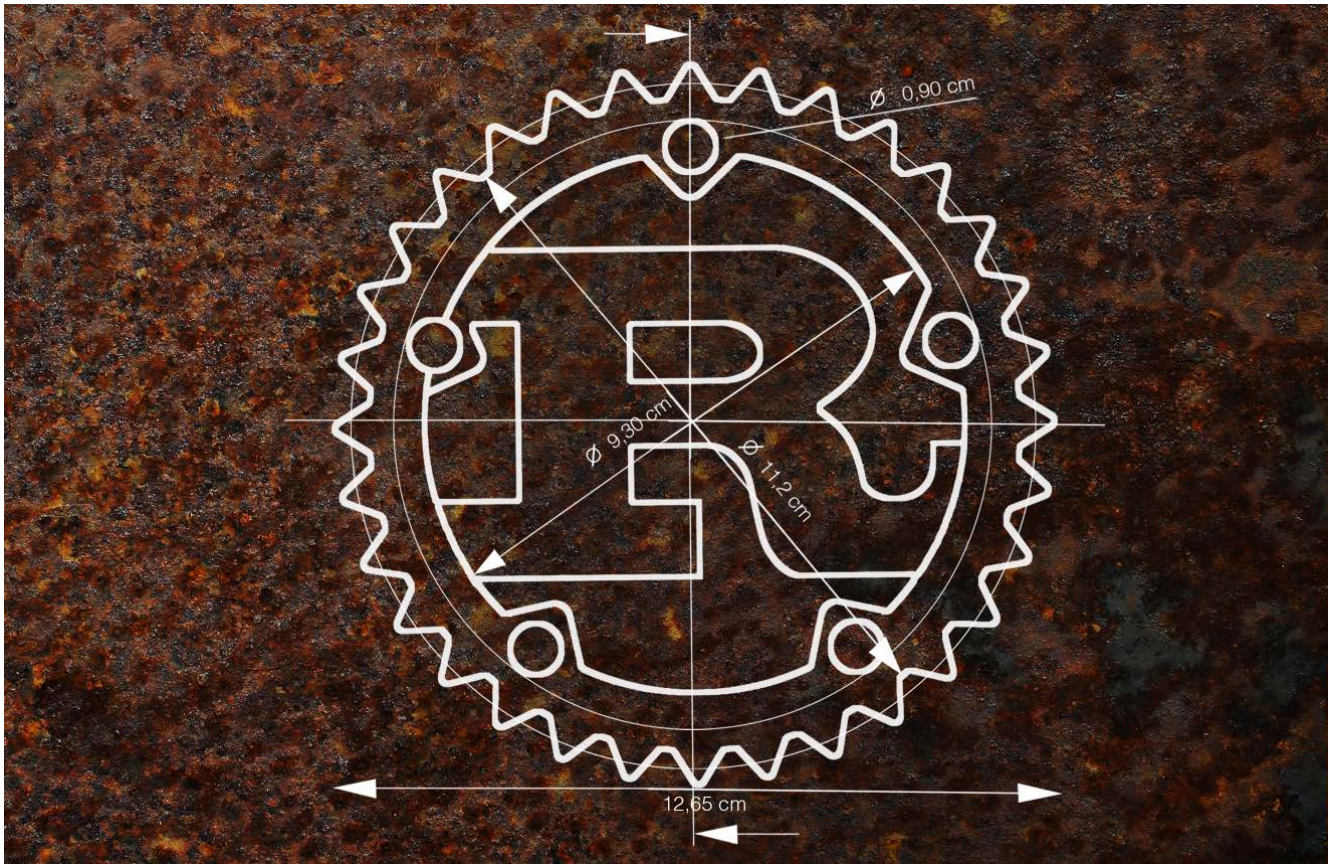


Was Rust hat, das andere nicht haben

# Neuling im Programmiersprachenwald

**Georg Nold**

Rust räumt Hindernisse aus dem Weg, an denen Sprachen wie C und C++ bisher gescheitert sind. Dabei verspricht es vor allem Speichersicherheit und baut seine Anhängerschaft stetig aus.

Wer die Wirtschaftspresse verfolgt, stellt fest, dass der digitale Wandel weite Teile der Industrie in Deutschland erreicht hat. Es sind nicht mehr ausschließlich die Protagonisten der IT-Branche, die Software als zentrales Mittel zur Entwicklung neuer Produkte nutzen. Mit der Automobilindustrie sind auch die Schwergewichte der deutschen Wirtschaft vom Wandel betroffen. In globalisierten Märkten wächst der Druck zur Automatisierung und zur Datenökonomie; der Umstieg auf nachhaltige Energienutzung treibt die Digitalisierung zusätzlich an. Statt elektromechanischer Getriebe und Steuerungen setzt man Computer in Verbindung mit Sensoren und Aktoren ein, um „intelligente“ Geräte und Abläufe zu schaffen. Die Produkte werden komplexer und die Software beansprucht einen wachsenden Anteil an der Wertschöpfung.

Da trifft es sich schlecht, dass der sichere Einsatz softwaregestützter Produkte für Konsumenten und Unternehmen mit wachsenden Schwierigkeiten verbunden ist. Regelmäßig tauchen Berichte über Sicherheitslücken in Software auf, die es Angreifern erlauben, Computer und Geräte zu übernehmen, und der Aufwand für Gegenmaßnahmen steigt. IT-Sicherheitsfachleute von Microsoft und Google berichten übereinstimmend, dass 70 Prozent aller Schwachstellen ihrer Produkte im Zusammenhang mit Speicherverwaltungsfehlern entstehen (mehr dazu unter ix.de/zn6d). In der Regel entwickelt man diese Produkte mit C oder C++. Offensichtlich sind die Anstrengungen der zurückliegenden Jahre gescheitert, die Speicherverwaltung von C/C++ zu verbessern.

Schon vor zehn Jahren begann Graydon Hoare als Mitarbeiter der Mozilla Founda-

tion die Arbeit an Rust. Aus den Erfahrungen mit dem Firefox-Browser reifte die Überzeugung, dass eine neue Compilersprache nötig sei, die sich an C- und C++-Entwickler richtet und dazu dienen sollte, Teile des Firefox neu zu implementieren. Sein zentrales Ziel lautete, mittels Automatisierung die Speicherverwaltung anzugehen, den Hauptgrund für Schwachstellen in systemnaher Software. Die Herausforderung hier ist, auf hardwarenahe Ressourcen bei ungebremster Geschwindigkeit zuzugreifen. Programmiersprachen mit Laufzeitumgebung (Lisp, Smalltalk, Python, Ruby, Java, C#, Go …) lösen das Speicherverwaltungsproblem mit einem Garbage Collector (GC), der über alle Variablen und Speicherbereiche Buch führt und sie nach Gebrauch bereinigt. Das Laufzeitsystem bremst dabei aber die Anwendung.

## Den Rücken freihalten

In Rust kümmert sich der Compiler während der Übersetzung darum, dass keine Operationen auf Variablen die Speichersicherheit gefährden. Um keine Ausführungsgeschwindigkeit einzubüßen, kommen, wann immer möglich, Zero Cost Abstractions zum Einsatz. Das bedeutet, der Compiler hat den Aufwand, nicht das übersetzte Programm während der Laufzeit. Grundlage dafür ist ein wohlüberlegtes Ownership-Konzept, das im Kern aus drei Regeln besteht: Jeder Wert in Rust hat eine Variable als ihren Besitzer (Owner). Es kann zu jedem Zeitpunkt immer nur einen Besitzer für Werte geben. Und sobald der Besitzer (Owner Variable) den Gültigkeitsbereich (Scope) verlässt, wird der Wert gelöscht.

Weitere darauf aufbauende Regeln schreiben vor, wie sich Zeiger nutzen lassen. So ist nur ein Zeiger je Wert erlaubt, der Änderungen zulässt. Zugriffe auf Arrays außerhalb des Wertebereichs werden zur Laufzeit unterbunden und führen zum Programmabbruch. Zusätzlich vermeidet Rust Tony Hoares „Billion Dollar Mistake“: Statt Null-Zeigern verwendet man Option-Typen, die aus dem Enum `None` und `Some(T)` bestehen. Statt eines Null-Werts weißt man Option-Variablen ohne Wert `None` zu. Variablen mit einem regulären Wert sind vom Typ `Some(T)`. Der Compiler erledigt dann den Rest und meldet fehlerhafte Zuweisungen. Das Beste daran ist allerdings, dass Rust mit diesen Regeln für Multi-Threaded-Anwendungen ebenfalls Speichersicherheit gewährleistet. Rust verspricht also nichts weniger als eine waschechte Zeitenwende in der Softwareentwicklung für hardware- und systemnahe Programmierung.

### iX-TRACT

- Rust kommt ohne Garbage Collector aus. Der Compiler gewährleistet während der Übersetzung Speichersicherheit, ohne dass es zu Geschwindigkeitseinbußen zur Laufzeit kommt.
- Rust-Code lässt sich in Verbindung mit C- und C++-Bibliotheken einsetzen.
- Mit der Gründung der Rust Foundation gewinnt die Sprache an Unabhängigkeit und bekommt ein sicheres Dach für ihre Weiterentwicklung.

## Auf dem Weg Richtung Zukunft

Seit Anfang Februar kümmert sich die Rust Foundation um alle Belange der Programmiersprache. Die gemeinnützige Organisation soll die Infrastruktur des Rust-Projekts beherbergen, die Community betreuen und die künftigen Geschicke der Sprache und ihres Ökosystems steuern. Mozilla engagiert sich als Platinsponsor, zu den weiteren Gründungsmitgliedern zählen AWS, Huawei, Google und Microsoft sowie fünf Teammitglieder des Rust-Projekts. An der bisherigen technischen Governance will die Stiftung nicht rütteln, auch die aktuellen Communitystrukturen sollen bestehen bleiben.

Ashley Williams, Interim Executive Director und Mitglied des Rust-Kernteams, sieht im Start der Foundation einen wichtigen Meilenstein in der Etablierung der Sprache: „Dies ist ein großer Schritt in der Entwicklung von Rust auf mehreren Ebenen; nicht zuletzt eine formale, finanzielle Zusage einer Reihe von weltweit führenden Unternehmen, die die Ankunft von Rust als produktionsreife Technologie in Unternehmen ankündigt."

Schon im vergangenen Jahr diskutierten die Linux-Kernel-Entwickler auf der Linux Plumbers Conference ernsthaft über die Idee, Rust im Kernel einzusetzen. Dabei ist die Idee allerdings nicht, den Kernel von Grund auf neu zu schreiben. Die Entwicklerinnen überlegen lediglich neuen Code hinzuzufügen, der in Rust geschrieben ist und sich sauber in die bestehende Kernel-Infrastruktur einfügt.

Linux-Schöpfer Linus Torvalds hat sich vor ein paar Wochen vorsichtig optimistisch zu Rusts Zukunft im Kontext des Open-Source-Betriebssystems geäußert. In einem kürzlich bei zdnet.com erschienenen Artikel (der vollständige Beitrag findet sich unter ix.de/zn6d) teilt Torvalds mit: „Ich bin an dem Projekt interessiert, aber ich denke, es wird von Leuten vorangetrieben, die sehr begeistert von Rust sind, und ich möchte sehen, wie es dann tatsächlich in der Praxis funktioniert." Torvalds schließt also mittel- bis langfristig keineswegs aus, dass die neue Programmiersprache zukünftig für den Linux-Kernel eine Rolle spielen könnte.

Mozillas erste Erfahrungen mit Rust mit der CSS-Komponente für Firefox ergaben eine Reduktion von Schwachstellen um über 70 Prozent. Auch Microsoft verfolgt die Entwicklung von Rust mit großem Interesse (etwa Gavin Thomas vom Microsoft Security Response Center, siehe ix.de/zn6d) und verspricht sich davon eine wesentliche Verbesserung der IT-Security.

## Rust bringt alle Werkzeuge mit

Jenseits der Speichersicherheit hat Rust alles an Bord, was die moderne Informatik im Werkzeugkasten für Programmiersprachen anbietet. Die Sprache erlaubt es, parallele Algorithmen zu entwickeln, unterstützt objektorientierte Programmierung, Traits, Closures und Pattern Matching. Zudem sieht Rust automatisierte Tests in Verbindung mit dem Package-Manager Cargo vor. Für systemnahe Einsatzzwecke können Entwicklerinnen den Sicherheitsgurt lösen und mit Unsafe-Code-Blöcken Zeiger auf Speicheradressen dereferenzieren. Zusätzlich können sie so C-Funktionen verwenden, für die der Compiler keine Speichersicherheit gewährleistet. Hier sind Entwickler gefordert, wie bisher von Hand die Speichersicherheit herbeizuführen.

Safe Code ist Standard und es bedarf des bewussten Opt-out, um ohne die Unterstützung des Compilers Daten zu verwalten. Vielen Entwicklern gefällt das, deshalb steht Rust seit fünf Jahren an der Spitze der „Most Loved Programming Language" auf Stack Overflow. Es wird also Zeit, dass Rust in der gesamten Industrie zum Einsatz kommt, um das Potenzial für qualitativ bessere Software zu heben. Rust kommt dabei zugute, dass man nicht sämtliche Investitionen verwerfen muss, die im C- und C++-Code stecken. Rust-Code kann C- und C++-Bibliotheken nutzen und C- und C++-Code kann Rust-Code verwenden. Damit gelingt eine schrittweise Einführung von Rust und gleichzeitig kann man sich zunächst auf kritische Bereiche im Softwareprojekt konzentrieren, wo der Nutzen am stärksten ist.

Einsteiger berichten von der steilen Lernkurve beim Verinnerlichen des Ownership-Konzepts und sind außerdem begeistert von der Qualität der Fehlermeldungen des Compilers, die helfen viele Stolpersteine wegzuräumen, bevor sie zur Laufzeit Probleme verursachen. Die Community um Rust hat sich innerhalb der letzten Jahre seit Rusts zweiter Edition von 2018 weiter professionalisiert. Das mündete im Februar 2021 in der Gründung der Rust Foundation. Damit hat sich das Projekt von Mozilla emanzipiert. In diesem Zuge gewann die Sprache mit Amazon AWS, Google, Huawei und Microsoft weitere gewichtige Unterstützer. Das schafft Kontinuität und sichert die Entwicklung für die Zukunft.

## Mit voller Kraft voraus entwickeln

Derweil arbeitet das Kernteam an der neuen Edition von Rust. Zu den wesentlichen Zielen gehören beispielsweise die ISO-Standardisierung und Zertifizierung des Compilers oder Verbesserungen des Cargo- und Build-Systems. Außerdem wollen die Sprachentwicklerinnen private Crates einbinden, um Bibliotheken im Unternehmenskontext besser zu integrieren, und planen, weitere Verbesserungen im Ökosystem vorzunehmen. Auch an diesem Punkt beweist das Rust-Kernteam Weitsicht, denn die Änderungen an zukünftigen Releases sollen die Kompatibilität nicht beeinträchtigen und folglich die Investitionssicherheit sicherstellen.

Rust ist inzwischen bereit für kritische Projekte, bei denen es sowohl auf Geschwindigkeit als auch auf Sicherheit ankommt. Amazon bietet schon erste kommerzielle Dienste an. Amazon Web Services (AWS) etwa hat vor Kurzem Bottlerocket Linux für Container veröffentlicht, das eine Menge Rust enthält. Linus Torvalds kann sich Rust sogar im Linux-Kernel vorstellen (siehe Kasten „Auf dem Weg in Richtung Zukunft").

Darüber hinaus wäre es wünschenswert, dass Rust in den Produkten der deutschen und europäischen Industrie stärkere Beachtung findet. Die Programmiersprache bietet vor allem die Chance, Qualität von Anfang an in die Software einzubauen, statt sie, wie bisher gängige Praxis, mit Patches erst im Nachhinein zu flicken. In diesem Kontext stünde es Unternehmen gut zu Gesicht, sich in der Rust Foundation zu engagieren. Frei nach dem Motto „Wes Brot ich ess, des Lied ich sing" lässt sich auf diese Weise die Zukunft von Rust mitgestalten. (csc@ix.de)

### Quellen

Links zu Hintergrundinformationen unter ix.de/zn6d

**Georg Nold**

ist Chief Technology Officer bei Heise Medien und verfolgt Rust aus der Engineering-Perspektive.