Berufliches Schulzentrum für Bau und Technik Dresden

Fachoberschule

Fachrichtung Technik

Facharbeit

im Fach Informatik

Wie würde eine Programmiersprache der nächsten Generation aussehen?

Verfasser: Simon Sommer

Klasse: FOS21B

Schuljahr: 2022/23

Betreuer: Herr Rottmann

Ort, Datum: Dresden, 14.09.2022

Inhaltsverzeichnis

1 Einleitung

Jedes Jahr werden viele neue Programmiersprachen veröffentlicht. Viele von ihnen werden anfänglich als Teil einer wissenschaftlichen Arbeit entwickelt. Oft arbeiten Privatpersonen sowie Studenten in ihrer Freizeit an ihren eigenen Sprachen, um Themen im Bereich der Implementation dieser kennenzulernen. Teilweise kommt es auch vor, dass Firmen oder Organisationen ein bestimmtes Problem lösen möchten, jedoch die passende Sprache dazu noch nicht existiert.[1] Mit den meisten von ihnen hat der durchschnittliche Entwickler nie etwas zu tun und doch bringen sie im Allgemeinen alle Programmiersprachen zusammen voran.

Eine neue Sprache hat den Vorteil, von speziellen mit der Zeit gewachsenen Gegebenheiten nicht behindert zu werden, wie es bei vielen älteren Vorgängern der Fall ist. Da man bei der Umsetzung von null starten kann, jedoch Konzeptuell alles Wissen aus vorherigen Versuchen benutzt wird, können wesentlich einfacher und schneller vollkommen neue Ideen ausprobiert werden. Bei einer Sprache, die von Tausenden Personen genutzt wird und möglichst stabil bleiben soll, ist dieser Prozess schwieriger.

Ein weiterer Vorteil von kleinen neuen Projekten ist, dass es weniger Bürokratie und Abstimmungen gibt, wodurch es möglich wird, iterativ schnell viele Konzepte auszuprobieren. Diese Prozesse sind bei größeren Projekten durchaus sehr wichtig, da oft viele Personen gleichzeitig an ihnen arbeiten und es sonst kaum Fortschritte geben würde.

Erscheinen die resultierenden Ansätze der „Neulinge” sinnvoll und werden großflächig adaptiert, resultiert daraus oft eine Integration in ältere, größere Sprachen. Genau solch eine Sprache soll im Rahmen dieser Arbeit durch die Analyse bewährter Technologien und Kombination mit neuen Ideen konzipiert und umgesetzt werden.

Diese Arbeit beschäftigt sich deshalb mit der Analyse vergangener sowie aktueller Programmiersprachen, um herauszufinden, in welche Richtung sich diese im Moment entwickeln. Aus den daraus gewonnenen Erkenntnissen soll danach eine neue Sprache entworfen werden, welche eine mögliche Antwort zur Frage: **Wie würde eine Programmiersprache der nächsten Generation aussehen?** wäre.

[1] Vgl. Stansifer, Ryan (1995): Theorie und Entwicklung von Programmiersprachen eine Einführung. München: Prentice Hall, S. 25-53.

2 Analyse aktuell benutzter Programmiersprachen

Der erste Schritt zur Beantwortung dieser Frage ist die Analyse aktueller Programmiersprachen. Mittlerweile selten genutzte Sprachen, aus deren Funktionalität jedoch immer noch sinnvolle Schlüsse gezogen werden können, sollen an anderer Stelle betrachtet werden.

Um die Anzahl der Sprachen in einem bearbeitbaren Rahmen zu halten, werden die unter Benutzern populärsten Programmiersprachen des Jahres 2022 analysiert. Die primäre Quelle für diese Daten bildet der StackOverflow Developer Survey 2022. Diese Umfrage der StackOverflow Plattform findet jedes Jahr statt und ist an alle programmierenden Personen gerichtet.

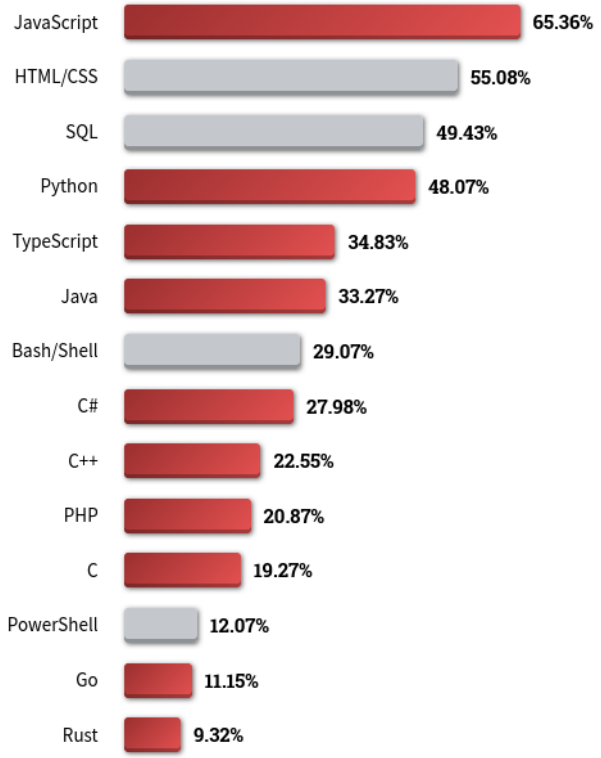

Abb. 1: Populärste Markup, Skript- und Programmiersprachen im Jahr 2022

Aus dem StackOverflow Developer Survey 2022 gehen JavaScript (1995), Python (1991), Typescript (2012), Java (1995), C# (2000), C++ (1985), PHP (1995), C (1978), Go (2009) und Rust (2010) als die zehn am meisten genutzten Programmiersprachen hervor.[2,3] HTML, CSS, SQL, Bash/Shell und Powershell werden hier ausgeschlossen, da sie alle stark spezialisiert und teilweise nicht Turing Vollständig[4] sind. Im Durchschnitt sind diese Sprachen demnach 25 Jahre alt.

---

[2] Vgl. N. N. (2022): StackOverflow Developer Survey 2022. https://survey.stackoverflow.co/2022/#programming-scripting-and-markup-languages, aufgerufen am 12.10.2022.
[3] Vgl. Benjamin QoChuk: Alter der Programmiersprachen. https://iq.opengenus.org/age-of-programming-languages, aufgerufen am 27.09.2022.
[4] Eine Turing Vollständige Sprache kann jeden erdenklichen Algorithmus implementieren.

Analyse

- unterstützte und bevorzugte Programmierparadigmen
- Syntax
- Types
    - core Types:
    - Literal Types (LiteralBool, LiteralNumber, LiteralString)
    - type definitions
- Execution: kompiliert / interpretiert / Mischung
- Mängel welche oft von Nutzern geäußert werden
- Besonderheiten / Features
        - const/mut und daraus folgende Optimierungsmöglichkeiten
    - type functions?
- Vergleich ältere Sprachen (Java, C, C++) mit neueren (Go, Rust)

Syntax

- genaue Details eher unwichtig (begründen)
- Syntax von Rym wegen persönlichen Präferenzen stark an Rust (ist immerhin in Rust geschrieben) angelehnt
- ein paar Beispiele jedoch begründen
    - weshalb curly braces sinnvoll sind
    - mit Autoformatierung begründen
    - genaue Formatierung, im speziellen Indentation, letzen endes weniger wichtig
    - darauf hinweisen das vieles expressions sind und diese schlecht ohne Anfangs- / Endtoken funktionieren

3 Vor- und Nachteile populärer Programmierparadigmen

- Populäre Programmierparadigmen analysieren um herauszufinden was
    - sich bei ihnen (nicht) bewährt hat [Quellen]
    - interessante Ideen sind

3.1 Imperative Programmierung

- Im Moment für traditionelle Programmiersprachen am meisten benutztes Paradigma [Quelle]
- Prozedurale Programmierung
    - Verkettung von Funktionen [Quelle]
- Objektorientierte Programmierung

3.2 Deklarative Programmierung

- Funktionale Programmierung
    - pure Funktionen
    - keine Seiteneffekte
    - wie funktioniert Eingabe / Ausgabe wenn es keine Seiteneffekte gibt

- Ziele der Sprache
    - Kombination von Elementen der typischen Imperativen Programmierung mit denen der Funktionale Programmierung
    - Seiteneffekte klar nachvollziehbar darstellen
    - siehe: https://gist.github.com/CreatorSiSo/55d293f165257cc6df74b150d67d390f

4 Abbildungsverzeichnis

5 Anlagenverzeichnis

6 Literaturverzeichnis

Stansifer, Ryan (1995): Theorie und Entwicklung von Programmiersprachen eine Einführung. München: Prentice Hall.

7 Quellenverzeichnis

StackOverflow Developer Survey 2022. https://survey.stackoverflow.co/2022/#programming-scripting-and-markup-languages, aufgerufen am 12.10.2022.

Alter der Programmiersprachen. https://iq.opengenus.org/age-of-programming-languages, aufgerufen am 27.09.2022.

8 Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und ohne fremde Mittel verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort/Datum: ……..…… Unterschrift: ……..……