HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Ruby als Lernprogrammiersprache

Nicolai Böttger und Jon-Steven Streller

Seminar-Arbeit im Studiengang „Angewandte Informatik“

17. Juni 2019

Autor 1: Nicolai Böttger
1476431
nicolai.boettger@stud.hs-hannover.de
Verfasste Seiten/Abschnitte: ...

Autor 2: Jon-Steven Streller
1475759
steven.streller@stud.hs-hannover.de
Verfasste Seiten/Abschnitte: ...

Prüfer: Prof. Dr. Dennis Allerkamp
Abteilung Informatik, Fakultät IV
Hochschule Hannover
dennis.allerkamp@hs-hannover.de

Selbständigkeitserklärung

Mit der Abgabe der Ausarbeitung erklären wir, dass wir die eingereichte Seminar-Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die von uns angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht haben.

Hannover, den 17. Juni 2019

Inhaltsverzeichnis

1 Einführung

Die vorliegende Arbeit entstand im Rahmen des Seminars *„Programmiersprachen für den Einstieg“*. Ziel des Seminars war es mehrere Programmiersprachen auf ihre Tauglichkeit im Anwendungsbereich einer pädagogischen Lehrveranstaltung zu analysieren. Hierbei wurde jede Programmiersprache einzeln von verschiedenen Studierenden in Zweiergruppen vorgestellt und die jeweiligen Ergebnisse Ihrer Forschung präsentiert. Nachfolgend entstand ein Austausch zwischen den Studierenden über positive und auch negative Auffälligkeiten in der Präsentation.

1.1 Vorgehensweise

Um eine sinnvolle Bewertung der Sprache in Bezug zum Thema des Seminars zu erreichen haben wir zuerst nach Studien gesucht, welche Kriterien definieren, nach denen man eine Programmiersprache bewerten kann. Danach haben wir angefangen uns selbst in unsere vorzustellende Sprache *„Ruby“* einzuarbeiten. Nachdem wir ein grundsätzliches Verständnis der Sprache bekommen haben, haben wir angefangen *„Ruby“* in Bezug zu den vorher gewählten Kriterien zu bewerten.

1.2 Aufbau der Arbeit

Im folgenden **Hauptteil** wird die Programmiersprache *„Ruby“* vorgestellt. Hierbei werden auf besondere Merkmale der Sprache eingegangen und *Anwendungsbeispiele* visualisiert. Anschließend werden die Methodik und die Kriterien erläutert und definiert. Zum Abschluss werden die Forschungsergebnisse in Form einer Bewertung dargelegt.

2 Hauptteil

2.1 Was ist Ruby?

Im Jahr 1995 veröffentlichte *Yukihiro Matsumoto* die erste Version (Version 0.95) von *„Ruby“*. *Yukihiro Matsumoto* lies sich während der Entwicklung von *„Ruby“* von mehreren Programmiersprachen wie z.B. Perl, Smalltalk, Lisp, Ada inspirieren. *Ruby* besitzt eine tief integrierte Objektorientierung[1] (kurz OO).

[1] Ein System besteht in der Objektorientierung ausschließlich aus Objekten, die miteinander über Nachrichten kommunizieren. Jedes Objekt verfügt über Eigenschaften und Methoden. Die Eigenschaften beschreiben dabei über ihre Werte den Zustand eines Objektes, die Methoden die möglichen Handlungen eines Objektes. vgl. [WL01]

2.2 Arbeitsweise von Ruby

An dieser Stelle wird kurz auf die interne Arbeitsweise von Ruby eingegangen. Ruby-Code wird ein Statement nach dem anderen, von einem Interpreter übersetzt und ausgeführt. Im Gegensatz zum Compiler, analysiert dieser den Code schneller und das Schrittweise übersetzen erleichtert das Finden von Fehlern im Code, allerdings hat ein kompiliertes Programm meist eine bessere Performance.

2.3 Methodik

Um die Programmiersprache Ruby in Bezug zur Fragestellung, ob sie sich als Programmiersprache für den Einstieg eignet bewerten zu können, wurde als Methodik ein Kriterienkatalog erstellt, welcher die Sprache in hinsichtlich verschiedener Bereiche untersucht. Dieser Katalog besteht aus den Kriterien Einstiegsfreundlichkeit, Skalierbarkeit, Verständlichkeit, Dokumentation, Verbreitung der Sprache und Ausstattung der Schule. Eine besondere Gewichtung wurde hierbei auf die Kriterien Skalierbarkeit und Verständlichkeit gelegt, da diese die grundlegenden Aspekte zum Verstehen der Sprache sind. Nach einer kurzen Erläuterung der Kriterien, werden diese in Bezug zur Programmiersprache Ruby gesetzt und es wird erörtert, inwiefern das Kriterium zutrifft (trifft voll zu, trifft teilweise zu, trifft nicht zu)

2.3.1 Einstiegsfreundlichkeit

Das Kriterium der *Einstiegsfreundlichkeit* fast alles zusammen, was benötigt wird um mit dem Programmieren in der Sprache anzufangen. So sollte die Installation von möglichen Compilern oder Interpretern nicht zu aufwändig sein und ohne große Fachkentnisse vorgenommen werden können. Auch wäre es von Vorteil, wenn die Installationsbestandteile nicht allzu viel Speicher einnehmen. Ebenfalls wird hier berücksichtigt, ob beispielsweise mehrere Dateien angelegt, oder andere Vorgänge wie ein mögliches Linken von Dateien geschehen muss, ehe man ein geschriebenes Programm ausführen kann.

Die technischen Vorraussetzungen, die benötigt werden, um mit der Programmiersprache *„Ruby“* arbeiten zu können beschränken sich ausschließlich auf einen *Ruby-Interpreter*, welchen man auf der offiziellen Website von *„Ruby“* herunterladen kann. Auch die anschließende Installation läuft unter dem Betriebssystem *Microsoft Windows* so ab, wie man es von anderen heruntergeladenen Programmen gewohnt ist. Die Größe der Standardedition dieses Interpreters beläuft sich auf ca. 60MB und nimmt somit vergleichsweise wenig Speicher auf der Festplatte ein (Java SE Development Kit 12.0.1 160MB). Theoretisch könnte man in *„Ruby“* ein komplettes Programm in nur einer Datei schreiben, welches man ohne weitere Zwischenschritte von einem Terminal aufrufen könnte. Zusätzlich wird mit dem

Download des Interpreters das Programm *IRB* (Interactive Ruby) installiert, mit welchem man unmittelbares Feedback auf eingegebenen Ruby-Code erhält.

Dementsprechend trifft das gewählte Kriterium der *Einstiegsfreundlichkeit* voll zu.

2.3.2 Skalierbarkeit

Unter Skalierbarkeit wird alles bewertet, was mit den spezifischen Eigenschaften und Besonderheiten der Sprache zu tun hat. Wie hoch ist die Komplexität um eine Ausgabe zu erzeugen? Wie sehr bindet mich die Programmiersprache an bestimmte Syntax und Semantik? Ein weiterer Punkt der nicht zu vernachlässigen ist, ist die Plattformunabhängigkeit der Sprache. Hierzu wird im nachfolgenden Abschnitt näher drauf eingegangen.

2.3.3 Verständlichkeit

Bei der *Verständlichkeit* geht es vor allem, dass der geschriebene Quellcode möglichst simpel und verständlich wirkt. Ein abstrakter Quellcode wäre nicht fördernd um grundlegende Konzepte des Programmieren zu vermitteln. Außerdem ist ein schnelles und klares Feedback des Interpreters essentiell.

2.3.4 Dokumentation

Eine aktuelle und feinsäuberlich ausgearbeitete *Dokumentation* der Programmiersprache ist überlebenswichtig für jene. Beachtet werden muss auch zum Beispiel wie viel Lehr/-Sachbücher über die Programmiersprache publiziert wurden. Ergänzend hierzu ist es immer sehr positiv, wenn die Programmiersprache über viele Entwickler/-innen verfügt, da so ein großer Support bei technischen Fragen eher gegeben ist. Damit Schüler auch von Zuhause aus programmieren können, wäre es hilfreich, wenn spezifische Webseiten mit Tutorials oder Lernvideos zum Erlernen dieser Programmiersprache existieren würden.

Hinsichtlich Literatur, also Lehr- und Sachbüchern lässt die Dokumentation der Sprache *„Ruby“* zu wünschen übrig. Es gibt zwar einige Bücher, diese sind aber in der Regel schon etwas älter und dementsprechend nicht mehr kompatibel mit dem heutigen Stand der Sprache. Besonders schwierig ist es gute Literaturbücher in der deutschen Sprache zu finden (https://wiki.ruby-portal.de/Literatur.html). Im Gegensatz dazu gibt es aber einige gute Videotutorials, entweder kostenlos auf Youtube oder kostenpflichtig - teilweise mit Übungsaufgaben - auf Plattformen wie „Udemy“ um sich das Programmieren mit *„Ruby“* selbst beizubringen. Trotz der Tatsache, dass man die Grundlagen der Programmiersprache so auch gut über das Internet lernen kann, trifft das Kriterium einer ausgereiften *Dokumentation*, aufgrund fehlender schriftlicher Literatur nur teilweise zu.

2.3.5 Verbreitung der Sprache

Unbeliebte Programmiersprachen sind in der Regel auch nicht so stark in der Arbeitswelt gefordert, weshalb es sehr ernüchternd sein kann, eine Programmiersprache pädagogisch vermittelt zu bekommen die keine reelle Perspektive in der Zukunft besitzt. Zusätzlich ist eine Würdigung beziehungsweise eine Akzeptanz in der „Programmiergesellschaft“ wünschenswert, da es durchaus vorkommt, dass Programmiersprachen sich eher schwer durchsetzen, wenn diese nicht von der o.g Zielgruppe akzeptiert werden. Desweiteren ist eine aktive Weiterentwicklung der Programmiersprache eine elementare Voraussetzung um gegen andere aktive und etablierte Sprachen zu konkurrieren.

2.3.6 Ausstattung der Schule

Selbstverständlich sollten die Lehrmittel beziehungsweise Lizenzkosten im Optimalfall keine Kosten verursachen, um auch Schulen mit wenig Budget eine Perspektive bieten zu können. Ideal wäre ein/e Lehrbeauftragter/-in die bereits grundlegende Kenntnisse in der Programmiersprache beherrscht. Somit könnten Ausbildungsmaßnahmen für den/die Lehrbeauftragte/-n eingespart werden und die eingesparten Kosten in die Lehrveranstaltung investiert werden

3 Schlussbemerkungen

4 Literaturverzeichnis

[WL01] Wirtschaftslexikon Gabler