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. Dazu haben wir zuerst nach wissenschaftlichen Arbeiten gesucht, welche sich schon einmal mit diesem Thema befasst haben. Da wir dadurch aber nicht ausreichend fündig wurden, haben wir danach nach Arbeiten gesucht, die sich mit ähnlichen Fragestellungen befasst haben und Kriterien dieser auf unser Thema übertragen. 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.

1.3 Anforderungen

Die nun genannten Anforderungen sind durch die Recherche und Sensibilisierung mit der Forschungsfrage entstanden. Grundlegend sollte es sich hierbei mindestens um eine achte oder neunte Klasse handeln, da viele Programmierausdrücke in der Programmiersprache sehr ähnlich oder gar gleich zur Mathematik sind. Eine gewisse Reife ist Vorauszusetzen, um gewährleisten zu können, dass der/die Schüler/-in sich die Auswirkung aufs zukünftige Berufsleben in Hinsicht der erworbenen Qualifizierung bewusst ist um somit auch ein Interesse an der Thematik zu entwickeln. Außerdem

sollte das Prinzip von Logik zumindest vorhanden sein. Vorkenntnisse im Bereich der Entwicklung von Anwendung wäre wünschenswert aber keine Anforderung an den Schüler. Allerdings sollte die Bedienung eines Computers keine Probleme darstellen und grundlegende Steuerung wie zum Beispiel mit der Maus sollte bereits beherrscht werden.

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).

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)

[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]

2.3.1 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.

Durchaus positiv sind die herrschenden Konventionen in *„Ruby“*. Hierbei lässt die Sprache selbst sehr viel Spielraum und Akzeptanz bei der Syntax und Semantik. Dies äußert sich sehr stark bei der Verwendung von Semikolon - während es bei vielen etablierten Programmiersprachen strikt und nur eine Richtlinie gibt, wird bei *„Ruby“* ein ganz anderer Weg eingeschlagen. Es ist möglich, aber nicht nötig Semikolons zu setzen. Es ist auch nicht nötig sich für einen Stil zu entscheiden, so kann jeder seinen eigenen Programmierstil in dem Lehrprozess entwickeln. Wie in Abschnitt 2.3.3 bereits erwähnt, ist das Konzept der Objektorientierung nicht zwingend zum anwenden von *„Ruby“* erforderlich. Dadurch ist es von *„Ruby“* selbst gegeben, bereits mit nur **einer**

Zeile Code eine Ausgabe auf der Kommandozeile[2] zu wiedergeben. Dieser Effekt wird erzielt weil Klassen in *„Ruby“* nicht erforderlich sind und somit sehr viel Quellcode bei einfachen Anwendungen eingespart werden kann.

In den meisten Fällen haben Programmiersprachen nur einen oder wenig Anwendungsgebiete. *„Ruby“* sticht hierbei etwas heraus, denn die Vielseitigkeit der Programmiersprache ist enorm. *„Ruby“* besitzt die Möglichkeit als Bash-Anwendung[3] und separat in der Webentwicklung zu fungieren. Dies ist ein exemplarisches Beispiel wie stark der Kontrast beziehungsweise die Einsatzmöglichkeiten von *„Ruby“* sind.

Das erweitern von Bibliotheken beziehungsweise Paketen gestaltet sich in *„Ruby“* denkbar einfach. Hier punktet *„Ruby“* mit einem zentralen Paketverwaltungssystem (Link: Ruby Gems) in dem zusätzlich für jedes Paket eine Installationsanleitung sowie eine Dokumentation und Quellcodebeispiele zum verdeutlichen der Arbeitsweise des Paketes beinhaltet. Hierbei muss jedoch beachtet werden, dass jeder die Möglichkeit hat Pakete selbst zu veröffentlichen und somit nicht verifiziert werden kann, dass jedes Paket fehlerfrei beziehungsweise auf einem gewissen Niveau entwickelt wurde. Allerdings ist ein hoher Download-Aufruf ein Indikator dass das Paket einem gewissen Standard entspricht. Hierbei kann **nur** von einer Korrelation zwischen Funktionalität des Quellcodes und Download-Anzahl ausgegangen werden!

Negativ anzumerken ist, dass Programmiersprachen die einen Interpreter verwenden (so wie *„Ruby“*) in der Ausführung langsamer sind als Programmiersprachen mit einem Kompilierer (engl. Compiler).

Trotz der Tatsache dass die Sprache eine hohe Toleranz in Bezug auf *Skalierbarkeit* aufweist, trifft das Kriterium wegen der eventuellen schlechten Qualität der zu verfügbar gestellten Pakete und aufgrund der Nutzung eines *Interpreters* in Zusammenhang des Geschwindigkeitsdefizit nur teilweise zu.

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.

Die Programmiersprache *„Ruby“* besitzt eine sehr hohe Skalierbarkeit. Auch wenn *„Ruby“* eine tief integrierte Objektorientierung (nachstehend „OO“) besitzt, ist es nicht zwingend erforderlich das Konzept der eigentlichen OO zu verstehen. In *„Ruby“* ist es auch nicht nötig einen expliziten Einstiegspunkt zu definieren, wie es zum Beispiel in Java mit der klassischen Main-Methode der Fall ist. Was das erlernen der

[2] Kommandozeile: Hierbei handelt es sich um eine grundlegende Anwendung die meist durch Texteingaben gesteuert wird.
[3] Bash-Anwendung: automatisiertes Programmieren auf der Kommandozeile

Programmiersprache *„Ruby“* auch deutlich vereinfacht, ist dass *„Ruby“* an dem Englischen angelehnt ist und der gesprochenen Sprache sehr ähnelt. Dies äußert sich öfter in Quellcodebeispielen, bei denen man einzelne Abschnitte wie gesprochene Sätze ablesen kann. Der *Interpreter* hilft aktiv bei der Fehlersuche im Quellcode und somit ist der Grad der Frustration bei nichtfunktionierenden Code geringer als bei Programmiersprachen die auf einen *Kompilieren* (engl. Compiler) setzen.

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

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.

Die *„Ruby-Doc“* (Abkürzung von „Ruby-Dokumentation“) (Hier einsehbar) besitzt eine sehr ordentliche Strukturierung und weist ein sehr klares und verständliches Layout auf. Gerade für Anfänger ist es wichtig eine sauber ausgearbeitete Dokumentation vorzufinden, damit der Lehrprozess hier nicht durch Verständnisprobleme wegen mangelnder Erläuterung unterbrochen wird. Hierbei finden sich in der *„Ruby-Doc“* Anwendungsbeispiele die zur Unterstürzung des Verständnis beitragen.
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 (Hier einsehbar). 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. Anfänglich lässt sich sagen, dass *„Ruby“*, wie auch schon im Kriterium der Skalierbarkeit erklärt, grundsätzlich vielseitig einsetzbar ist. Auch lässt die Statistik von „RedMonk“ ([BP 01]), die die größten Programmiersprachen nach Beliebtheit auf „Stack Overflow“ (Anzahl Tags) und GitHub (Anzahl Projekte), in der *„Ruby“* in beiden Bereichen gut abschneidet, vermuten, dass die Sprache zumindest eine große Community besitzt. Trotz dieser vielseitigen Einsatzmöglichkeit sieht es in der Praxis aber doch meist so aus, dass *„Ruby“* hauptsächlich in Verbindung mit dem Framework „Ruby on Rails“ zur Entwicklung von Webanwendungen benutzt wird. Trotzdem, nimmt sie im Vergleich zu Sprachen wie „PHP“ oder „Asp.net“ nur einen kleinen Teil der Webseiten ein ([RW 01]) . Auf der anderen Seiten wird die Sprache aber in relativ regelmäßigen Abständen seit 2002 weiterentwickelt ([RR 01]). Zusammenfassed lässt sich also festhalten, dass *„Ruby“* zwar eine aktive Community hat, in der Praxis aber nicht allzu oft verwendet wird.

Dementsprechend trifft das gewählte Kriterium der *Verbreitung der Sprache* teilweise zu.

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.

„Ruby“ ist eine frei verfügbare Sprache und ist mit keinerlei Lizenzkosten an den/die Entwickler/-in geknüpft. Durchaus positiv ist auch die simple Verwaltung beziehungsweise Wartung von *„Ruby“*. Hierbei wird kein externer Server benötigt, wofür weitere Fachkenntnisse erforderlich wären. Somit sind die Kosten und der Aufwand für die Benutzung von *„Ruby“* niedrig, da kein weiteres Equipment außer der Computer erforderlich ist. Da *„Ruby“* wie bereits in 2.1 erwähnt auf größere Programmiersprachen basiert beziehungsweise von diesen inspiriert wurde, besteht eine größere Chance eine Lehrkraft zu finden die bereits die Grundlagen beherrscht. Somit kann man davon ausgehen, dass die Lehrkraft einen schnellen Einstieg nach kurzer Einarbeitungsphase in die Besonderheiten von *„Ruby“* absolvieren und somit die Schüler unterrichten kann.

Dementsprechend trifft das gewählte Kriterium der *Ausstattung der Schule* voll zu.

3 Schlussbemerkungen

4 Literaturverzeichnis

[WL01] Wirtschaftslexikon Gabler

5 Quellen

[RW 01]Ruby in Webanwendungen
https://i1.wp.com/www.pixelcrayons.com/blog/wp-content/uploads/2016/01/Most-popular-server-side-programming-languages.png?w=415&ssl=1)

[BP 01]Beliebtheit von Programmiersprachen
https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/

[RR 01]Ruby Releases
https://www.ruby-lang.org/en/downloads/releases