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“

21. 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 21. Juni 2019

1 Abstrakt

Dieser Abstrakt fasst den Inhalt und die Ergebnisse dieser Arbeit kurz zusammen. Diese Arbeit beschäftigt sich mit der Frage, ob sich die Programmiersprache *„Ruby“* als erste Programmiersprache für den Einsatz in einer Schule oder ähnlichem eignet. Die aufgestellten Kriterien - Einstiegsfreundlichkeit, Skalierbarkeit, Verständlichkeit, Dokumentation, Verbreitung der Sprache und Anforderungen an die Lehranstalt - welche zum Beantworten dieser Frage dienen sollten, wurden für *„Ruby“* überwiegend zutreffend bewertet, weswegen diese Arbeit zu dem Ergebniss kommt, dass sich *„Ruby“* grundsätzlich schon für diesen Zweck eignet. Trotzdem ist zu beachten, dass dies nicht heißen soll, dass Ruby die perfekte erste Programmiersprache wäre.

Inhaltsverzeichnis

2 Einführung

Die vorliegende Arbeit entstand im Rahmen des Seminars *„Programmiersprachen für den Einstieg“*. Ziel des Seminars war es eine Programmiersprache zu finden, die sich gut als erste Programmiersprache zum Lernen eignet. Dazu wurden mehrere Programmiersprachen auf ihre Tauglichkeit im Anwendungsbereich einer pädagogischen Lehrveranstaltung analysiert. Hierbei wurde jede Programmiersprache einzeln von verschiedenen Studierenden in Zweiergruppen vorgestellt und die jeweiligen Ergebnisse ihrer Recherche präsentiert. Nachfolgend entstand ein Austausch zwischen den Studierenden über positive und auch negative Aspekte in der Präsentation. Diese Arbeit beschäftigt sich mit der Frage, ob sich die Programmiersprache *„Ruby“* als Programmiersprache für den Einstieg eignet.

2.1 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 hatten, haben wir angefangen *„Ruby“* in Bezug zu den vorher gewählten Kriterien zu bewerten. Am Ende haben wir diese einzelnen Bewertungen noch zu einer endgültigen Bewertung zusammengefasst.

2.2 Aufbau der Arbeit

Die Arbeit beginnt damit, dass zuerst die Anforderungen an die Schule und die Schüler in Bezug zum Lernen einer Programmiersprache dargestellt werden. Als nächstes wird die Sprache *„Ruby“* an sich kurz vorgestellt und etwas auf ihre interne Arbeitsweise eingegangen. Danach wird die Methodik vorgestellt, nach der versucht wurde, die Fragestellung der Arbeit zu beantworten. Im Hauptteil werden dann die in Verbindung mit der Methodik entstandenen Kriterien definiert und die Programmiersprache *„Ruby“* anhand jener bewertet. Anschließend daran folgt im Schlussteil zum einen noch eine zusammenfassende Bewertung der Sprache im Bezug zum Thema und zum anderen einige Schlussbemerkungen, die es beim Interpretieren der Ergebnisse dieser Arbeit und des durchgeführten Seminars an sich zu beachten gibt.

2.3 Anforderungen

Die nun genannten Anforderungen sind durch die Recherche und Interpretierung der Forschungsfrage entstanden. Grundlegend sollte es sich bei der Zielgruppe um mindestens 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 sich der/die Schüler/-in der Auswirkung aufs zukünftige Berufsleben in Hinsicht der erworbenen Qualifizierung bewusst ist, um somit auch ein Interesse an der Thematik zu entwickeln. Außerdem sollte ein gewisses Verständnis für Logik und die grundsätzliche Fähigkeit zur Bedienung eines Computers vorhanden sein.

3 Hauptteil

3.1 Was ist Ruby?

Im Jahr 1995 veröffentlichte *Yukihiro Matsumoto* die erste Version (Version 0.95) von *„Ruby“*. *Yukihiro Matsumoto* ließ sich während der Entwicklung von *„Ruby“* von mehreren Programmiersprachen wie z.B. Perl, Smalltalk, Lisp, Ada inspirieren. *Ruby* besitzt zwar eine tief integrierte Objektorientierung[1] (kurz OO), kann aber auch mit Programmierparadigmen wie prozeduraler oder funktionaler Programmierung verwendet werden. Ruby wird ungefähr in einem Intervall von 3 Monaten mit Updates versorgt. Im Moment (17. April 2019) befindet sich Ruby in Version 2.6.3 Die Version 2.7.0 steht schon in einer Vorabversion zur freien Verfügung. Die Programmiersprache Ruby ist plattformunabhängig und somit auf kein bestimmtes System beschränkt.

3.2 Arbeitsweise von Ruby

An dieser Stelle wird kurz auf die interne Arbeitsweise von Ruby eingegangen. Die Übersetzung von Quellcode zu Maschinencode funktioniert in *„Ruby“* über einen *Interpreter*. Um ein besseres Verständnis zu vermitteln, wird hier auf die Unterscheidung zwischen Compiler und Interpreter eingegangen. Der *Interpreter* arbeitet Zeile für Zeile den Quellcode ab und stoppt sofort, wenn ein Fehler in einem Ausdruck gefunden wurde, während bei dem Compiler direkt der ganze Quellcode analysiert wird und erst am Ende alle Fehler aufgelistet werden. Dies erleichtert die

[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]

Arbeit beim Debugging[2] ungemein. Auf der anderen Seite haben Programme, die in einer Sprache, welche durch einen Compiler übersetzt wird, geschrieben werden, meist eine bessere Performance (vgl. [DI01]).

3.3 Methodik

Um die Programmiersprache Ruby in Bezug auf die Fragestellung, ob sie sich als Programmiersprache für den Einstieg eignet, bewerten zu können, wurde als Methodik ein Kriterienkatalog erstellt, welcher die Sprache hinsichtlich verschiedener Bereiche untersucht. Dieser Katalog besteht aus den Kriterien Einstiegsfreundlichkeit, Skalierbarkeit, Verständlichkeit, Dokumentation, Verbreitung der Sprache und Anforderung an die Lehranstalt. 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)

3.3.1 Einstiegsfreundlichkeit

Das Kriterium der *Einstiegsfreundlichkeit* fasst 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 Voraussetzungen, die benötigt werden, um mit der Programmiersprache *„Ruby“* arbeiten zu können beschränken sich ausschließlich auf einen *Ruby-Interpreter*, welchen man auf der offiziellen Webseite 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 aus 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 und welches

[2] *Debugging* bezeichnet den Prozess des Findens und Behebens von Fehlern im geschriebenen Code. vgl. [CD01]

sich demnach gut eignet, um die Funktion einzelner Code-Ausschnitte zu überprüfen und zu verstehen([RM01]).

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

3.3.2 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. Hierauf wird im nachfolgenden Abschnitt näher 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 nur eine Möglichkeit 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 während des Lernprozesses entwickeln. Auf der anderen Seite kann dies aber auch zu schlechten Angewohnheiten, wie einem unkonsisten Programmierstil führen, welcher den geschriebenen Code unlesbarer machen könnte. Wie in Abschnitt 3.1 bereits erwähnt, ist das Konzept der Objektorientierung nicht zwingend zum Programmieren mit *„Ruby“* erforderlich. Zum Beispiel kann man in *„Ruby“* mit nur **einer** Zeile Code eine Ausgabe im Terminal[3] tätigen. Dies ist möglich, weil auch Klassen in *„Ruby“* nicht erforderlich sind, und somit kann sehr viel Quellcode bei einfachen Anwendungen gespart werden.

Die Anwendungsgebiete zur Programmierung mit *„Ruby“* sind vielfältig. In Ruby geschriebene Programme können zum Beispiel als Bash-Anwendung[4] aber auch in der Webentwicklung eingesetzt werden. 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 (Ruby Gems), in dem zusätzlich für jedes Paket eine Installationsanleitung sowie eine Dokumentation und Quellcodebeispiele zum Verdeutlichen der Arbeitsweise des Paketes vorhanden sind. 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 natürlich ein Indikator dafür, dass das Paket einem gewissen Standard entspricht.

[3] Terminal: Programm zur Eingabe und Ausgabe von Daten, vgl. [WP01].
[4] Bash-Anwendung: Geschriebene Datei, die automatisch Kommandozeilenbefehle ausführt

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

Trotz der Tatsache, dass die Sprache eine hohe Toleranz in Bezug auf das Kriterium *Skalierbarkeit* aufweist, trifft dieses wegen der Gefahr der Angewöhnung eines schlechten Programmierstils nur teilweise zu.

3.3.3 Verständlichkeit

Bei der *Verständlichkeit* geht es vor allem darum, dass der geschriebene Quellcode möglichst simpel und verständlich wirkt. Ein abstrakter Quellcode wäre nicht fördernd um grundlegende Konzepte des Programmierens zu verstehen.

Die Programmiersprache *„Ruby“* besitzt eine sehr hohe *Verständlichkeit*. 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 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* führt den Quellcode Zeile für Zeile aus und stoppt sobald ein Fehler auftritt, wodurch der Grad der Frustration bei nichtfunktionierendem Code geringer ist, als bei Programmiersprachen, die auf einen Compiler setzen, da dieser nach dem Kompilieren sofort alle Fehler ausgibt, was teilweise etwas erschlagend wirken könnte.

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

3.3.4 Dokumentation

Eine aktuelle und fein säuberlich 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“* 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 Lernprozess hier nicht durch

Verständnisprobleme wegen mangelnder Erläuterung der Methoden unterbrochen wird. Hierbei finden sich in der *„Ruby-Doc“* Anwendungsbeispiele, die zur Unterstützung des Verständnisses 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. 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.

3.3.5 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 zu lernen, die keine reelle Perspektive für die Zukunft bietet. 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 oben genannten 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“ ([RM 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 ([PC 01]). Auf der anderen Seite wird die Sprache aber in relativ regelmäßigen Abständen seit 2002 weiterentwickelt ([RL 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.

3.3.6 Anforderung an Lehranstalt

Selbstverständlich sollten die Lehrmittel beziehungsweise die Beschaffung nötiger Lizenzen im Optimalfall keine Kosten verursachen, damit auch Schulen mit geringerem Budget diese Programmiersprache und nötige weitere Programme erwerben können. Ideal wäre ein/e Lehrbeauftragte/r die bereits grundlegende Kenntnisse in der Programmiersprache beherrscht. Somit könnten Ausbildungsmaßnahmen für die/den 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 werden keine Drittanbieterprogramme benötigt, wofür weitere Fachkenntnisse erforderlich wären. Somit sind die Kosten und der Aufwand für die Benutzung von *„Ruby“* niedrig, da kein weiteres Equipment außer dem Computer erforderlich ist. Da *„Ruby“* wie bereits in 3.1 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. Eine permanente Internetverbindung wäre von Vorteil, da es auf einfachem Wege nicht möglich ist, externe Pakete von der Ruby Gems Webseite lokal über ein Wechselmedium oder ähnlichem auf allen Rechnern zu installieren.

Auch wenn man den Punkt mit der Internetverbindung durchaus bemängeln kann, ist dies kein grundlegendes Problem, welches die Lernzielerfolge beeinträchtigen würde, da gerade bei Anfängern/-innen teilweise oder auch komplett auf das Paketverwaltungssystem verzichtet werden kann, um grundlegende Programmiertechniken zu lehren. Dementsprechend trifft das gewählte Kriterium *Anforderung an Lehranstalt* trotz des kleines Mangels voll zu.

4 Bewertung

In diesem Abschnitt wird eine abschließende Bewertung zu der Frage, ob sich *„Ruby“* als erste Programmiersprache für den Einsatz im Unterricht eignet, zusammenfassed dargestellt. Die jeweiligen Bewertungen der einzelnen Kriterien führen zu einer Gesamtbewertung der Sprache von drei mal „trifft voll zu“ (Einstiegsfreundlichkeit, Verständlichkeit, Anforderung an Lehranstalt) und drei mal „trifft teilweise zu“ (Skalierbarkeit, Dokumentation, Verbreitung der Sprache). Im Durchschnitt schneidet *„Ruby“* also eher positiv ab. Trotzdem ist dieses Ergebnis mit Vorsicht zu genießen, da es eigentlich notwendig ist, die Sprache an dieser Stelle mit anderen zu vergleichen, damit diese Bewertungen eine nachvollziehbare Aussagekraft bekommen.

5 Schlussbemerkungen

Bei der eigentlichen Forschungsfrage hatten die Forscher, in diesem Fall die Studierenden, freie Wahl bei der Auswahl der Kriterien, und diese waren in der Realität auch öfter sehr unterschiedlich. In manchen Gruppierungen hatte zum Beispiel das Kriterium *Kosten* eine relativ starke Gewichtung, während es in anderen Präsentationen gar keine Erwähnung fand. So konnte zum einen ein großes Spektrum an Kriterien gesammelt werden, zum anderen machte dies den objektiven Vergleich der einzelnen Sprachen aber schwieriger. Zusätzlich konnten auch die „Bewertungsnoten“ an sich frei gewählt werden, wodurch einige Gruppen ihre eigenen Kriterien mit Noten von 1-6 und andere mit „trifft zu“ oder „trifft nicht zu“ bewertet haben. All dies führte dazu, dass es am Ende im Allgemeinen schwer war, die eigene gewählte Sprache mit den anderen zu vergleichen und zu bewerten, welche Programmiersprache sich jetzt wirklich oder gar am „besten“, als erste Programmiersprache eignet.

6 Literaturverzeichnis

[WL01] Wirtschaftslexikon Gabler
https://wirtschaftslexikon.gabler.de/definition/objektorientierung-43907

[PC01] 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)

[RM01] Beliebtheit von Programmiersprachen
https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/

[RL01] Ruby Releases
https://www.ruby-lang.org/en/downloads/releases

[CD01] Debugging
https://www.collinsdictionary.com/de/worterbuch/englisch/debugging

[DI01] Compiler vs Interpreter https://www.dev-insider.de/der-unterschied-von-compiler-und-interpreter-a-742282/

[RM01] IRB http://ruby-for-beginners.rubymonstas.org/your_tools/irb.html

[WP01] Terminal
https://de.wikipedia.org/wiki/Terminal_(Computer)