

## The Winner takes it all

# OOP 2008: Dynamic Languages Shootout

Christian Albrecht, Stefan Tilkov

Im Rahmen der OOP 2008 wurde zum ersten Mal der „Dynamic Languages Shootout“ durchgeführt, ein Programmierwettbewerb, bei dem ein einfaches Spiel in Pflicht und Kür implementiert werden sollte. Unter den beeindruckenden Einreichungen in verschiedenen mehr oder weniger exotischen Sprachen setzte sich ein Klassiker durch.

► Warum sind dynamische – oder noch viel allgemeiner: alternative – Sprachen in letzter Zeit so populär, insbesondere als Gegenpol zur Mainstream-Sprache Java? Zwei Meinungen dazu haben wir im Angebot, suchen Sie sich die aus, die Ihnen besser gefällt.

Meinung 1: Die enorme Leistungssteigerung der Hardware und die Tatsache, dass Anwendungsperformance heute eher durch Netzwerk- und Datenbankdurchsatz als durch die Programmiersprache der Geschäftslogik begrenzt wird, führen dazu, dass mächtige High-Level-Sprachen, die früher für einen ernsthaften, produktiven Einsatz viel zu langsam gewesen wären, heute in der Praxis nutzbar sind.

Meinung 2: Java kennen wir mittlerweile alle so gut, dass es langweilig wird – also muss ein neues Spielzeug her. Vielleicht liegt die Wahrheit zwischen diesen beiden extremen Positionen, in jedem Fall aber erfreuen sich Programmiersprachen jenseits des Mainstreams eines immer größeren Interesses.

Grund genug für JavaSPEKTRUM, im Rahmen der OOP 2008 den ersten „Dynamic Languages Shootout“ zu organisieren, einen Wettbewerb, in dem die Teilnehmer ein einfaches Spiel in Pflicht und Kür in einer dynamischen Sprache ihrer Wahl implementieren sollten. Und natürlich war die am häufigsten gestellte Frage die nach den „erlaubten“ Programmiersprachen – was genau ist eine dynamische Programmiersprache?

Das wesentliche Merkmal einer dynamischen Sprache ist unserer Meinung nach, dass die Grenze zwischen Entwicklungs- und Laufzeit verschwimmt – was dazu führt, dass sich in einer dynamischen Umgebung Dinge noch ändern können, wenn es in einer statischen Umgebung dafür schon lange zu spät ist. Mit einer dynamischen, häufig interpretierten Sprache assozi-

ieren wir aber auch Aspekte wie eine besonders schnelle Entwicklungszeit oder besonders kompakten Code, während wir eine hohe Ablaufgeschwindigkeit eher einer statisch getypten, kompilierten Sprache zutrauen.

Für den Wettbewerb haben wir es uns einfach gemacht und uns die Freiheit genommen, die Kategorie willkürlich zu erweitern – kein Vorschlag ist abgelehnt worden, weil uns die Programmiersprache nicht dynamisch genug gewesen wäre.

## Das Spiel

Wie es bei einem richtigen Wettbewerb üblich ist, gab es auch beim „Dynamic Languages Shootout“ eine Aufgabe, genauer eine Programmieraufgabe, zu lösen. Damit der Spaß nicht zu kurz kommt, bestand die Aufgabe darin, ein Scrabble™-ähnliches Spiel umzusetzen, ein Wortanlegespiel, das sich kurz so erklären lässt: Aus einem begrenzten Buchstabenvorrat müssen die Spieler kreuzworträtselartig neue, gültige Wörter auf einem Spielbrett bilden. Gültige Wörter konnten anhand des OpenOffice Dictionary bestimmt werden, wobei auch gebeugte Wortvarianten (Affixe) und Eigennamen zulässig waren. Pro Runde sollten zu dem Buchstabenvorrat eines Spielers sieben neue hinzukommen. Gespielt wurden fünf Runden.

Die Wahl ist auf dieses Spiel gefallen, da einige algorithmische Kniffe zur Problemlösung angewandt werden mussten und zu einem Brettspiel – in der Kür – auch eine schöne grafische Oberfläche passte.

## Die Teilnehmer-Sprachen

Perl,	<a href="http://www.perl.org/">http://www.perl.org/</a>
Python,	<a href="http://www.python.org/">http://www.python.org/</a>
Ruby,	<a href="http://www.ruby-lang.org/de/">http://www.ruby-lang.org/de/</a>
Smalltalk,	<a href="http://www.smalltalk.org">http://www.smalltalk.org</a>
Haskell,	<a href="http://www.haskell.org/">http://www.haskell.org/</a>
Scheme,	<a href="http://www.schemers.org/">http://www.schemers.org/</a>
Groovy,	<a href="http://groovy.codehaus.org/">http://groovy.codehaus.org/</a>
Squeak,	<a href="http://www.squeak.org/">http://www.squeak.org/</a>
Scala,	<a href="http://www.scala-lang.org/">http://www.scala-lang.org/</a>
Lua,	<a href="http://www.lua.org/">http://www.lua.org/</a>

## Die Problematik

Die zwei Hauptschwierigkeiten, die es algorithmisch zu lösen galt, waren zum einen die geschickte Ablagestruktur des Wörterbuches bzw. die performante Suche darin und zum anderen die Verteilung oder das Anlegen der Wörter auf dem virtuellen Spielbrett, sodass eine höchstmögliche Punktzahl erzielt werden konnte.

## Die Teilnehmer

Unter allen Einreichungen zum Wettbewerb haben sich insgesamt zehn verschiedene dynamische Sprachen wiedergefunden (s. Kasten „Die Teilnehmer-Sprachen“). Von funktionalen wie zum Beispiel Scheme und Haskell, über gerade sehr moderne wie Ruby und Groovy, bis zu wiederentdeckten wie Smalltalk hatte das Teilnehmerfeld einiges zu bieten.

## Die Algorithmen

Zur Lösung dieser Probleme lieferten die Teilnehmer verschiedene Lösungsansätze. Wurde zur Ablage des Wörterbuches in



Abb. 1: Die Jury unter Leitung von S. Tilkov und C. Albrecht, die Gewinner der ersten drei Preise und weitere Platzierte.



den meisten Fällen ein Graph oder eine Baumstruktur gewählt, unterschiedlich die Herangehensweise zur Suche von gültigen Wörtern im Wörterbuch bei den meisten Lösungen. Nutzung von regulären Ausdrücken, Backtracking, DAWG oder Trie-Algorithmen und die Brute-Force-Methode (das Ausprobieren aller möglichen Buchstabenkombinationen) sind nur einige.

## Die Gewinner

Die drei ersten Plätze erreichten:

- ▼ Thorsten Seitz, Smalltalk und Seaside (s. Abb. 3)
- ▼ Mirko Stocker, Ruby und JRuby/Swing
- ▼ Robert Brandner, Lua

Der Gewinner implementierte eine „Trie“ genannte Datenstruktur (abgeleitet aus reTrieval, s. Abb. 2). Diese kann in einer Baumstruktur mehrere Zeichenketten gleichzeitig speichern. Jede Kante des Baumes entspricht einem zusätzlichen Buchstaben. Jeder Knoten bildet eine Zeichenkette, die der Verkettung der Kantenbuchstaben entspricht.

Die zweitplatzierte Ruby-Lösung verwendet ein Set und durchsucht dieses bei jedem Durchlauf. Bei der Lua-Lösung wird das Wörterbuch in einem doppelt verketteten Graphen abgelegt.

## Die Kür

Die grafische Oberfläche sollte zur Verschönerung der im Pflichtteil abzuliefernden Konsolenanwendung dienen. Hierbei konnte und sollte man aus dem Vollen schöpfen und das Beste aus „seiner“ Programmiersprache herausholen.

Das Framework der Wahl zur Erstellung der grafischen Oberfläche war für Thorsten Seitz „Seaside“, ein Open-Source-Web-Framework für Smalltalk, das ähnliche Features wie populäre Web-Frameworks in Java bietet, aber ausschließlich Smalltalk-Kenntnisse verlangt. So erfolgt zum Beispiel die Erstellung von HTML programmatisch und nicht über Templates; es werden Komponenten unterstützt, die einfach zusammengesetzt werden können, Code-Blöcke können direkt mit HTML-Komponenten assoziiert werden und auch moderne Ajax-Mechanismen fehlen natürlich nicht. Für die Kür der Ruby-Lösung wurde der Ruby-Code mit JRuby in Java-Bytecode übersetzt;

dadurch stand das komplette Swing-GUI-Toolkit von Java zur Verfügung, welches dann für die Kür genutzt wurde.

Der Drittplatzierte, die Lua-Lösung, verzichtet auf eine Kür. Diese Lösung verdankt ihren dritten Platz vor allem ihren ausgezeichneten Spielergebnissen (von allen Teilnehmern die höchste Punktzahl) bei äußerst kompaktem Code: Sie umfasst gerade einmal 364 Zeilen (zum Vergleich: die Ruby-Lösung hat 435 Zeilen, die Smalltalk-Lösung immerhin 1000).

Nicht nur die drei Erstplatzierten haben



Abb. 3: Thorsten Seitz stellt seine Lösung vor

## Der Gewinneralgorithmus

Thorsten Seitz: „In sechs Sätzen zusammengefasst ist die Idee des Algorithmus‘: sowohl das Wörterbuch als auch die Suchmuster jeder Runde (inkl. Platzhalter) jeweils in einen Retrieval-Tree (Trie) einzutragen. Das Nachschlagen geschieht dann durch Bilden der Schnittmenge der beiden Tries, wobei die Platzhalter mit den aktuellen Restbuchstaben expandiert werden. Hierdurch erreicht man quasi das gleichzeitige Nachschlagen aller Suchmuster im Wörterbuch. Dabei können doppelte Zweige im Ergebnis-Trie entstehen (Buchstabe lag schon bzw. Buchstabe wurde neu gelegt – beide Zweige unterscheiden sich durch die Restbuchstaben), weshalb diese rekursiv zusammengefasst werden, um wieder einen regulären Trie zu erhalten. Dieser enthält nun alle prinzipiell möglichen Lösungen, aus denen noch diejenigen aussortiert werden müssen, bei denen ungültige orthogonale Wörter entstehen. Aus den verbliebenen Lösungen wird mittels einer einfachen Strategie (codiert durch eine Closure) die zu verwendende Lösung herausgesucht.“

uns beeindruckt: Äußerst elegante Lösungen gab es z. B. auch in Haskell und Scheme. Leider lieferten sie nicht immer korrekte Ergebnisse.

Dass die Gewinnerlösung nicht in einer aktuell besonderem Hype unterworfenen Sprache, sondern im Klassiker Smalltalk realisiert wurde, finden wir prima – weil es auch die Jury-Mitglieder (zumindest zum Teil) überrascht hat. Und darum ging es uns: Natürlich ist jede der eingesetzten Sprachen, ebenso wie Java, C, C++, Ada, für einen produktiven Einsatz geeignet. Keine dieser Sprachen ist „die Beste“, jede hat Stärken und Schwächen. Wir sind aber davon überzeugt, dass es sich lohnt, über den Tellerrand der Umgebung, die man gewohnt ist, hinwegzusehen und sich mit alternativen Konzepten zu beschäftigen: Der ideale Entwickler der Zukunft ist mehrsprachig (in jeder Beziehung).

Den Teilnehmern am Wettbewerb, die allesamt erstklassige Arbeit geleistet haben, danken wir dafür, uns bei der Verbreitung dieses Gedankens geholfen zu haben. Eine Bedingung für die Einreichung war, dass der Code unter einer freien Lizenz (GPL oder LGPL) stehen muss – und damit sind wir in der Lage, Ihnen sämtliche Lösungen auf der Homepage des „Dynamic Languages Contest“ (s. u.) zum Download zur Verfügung stellen zu können. Viel Spaß!

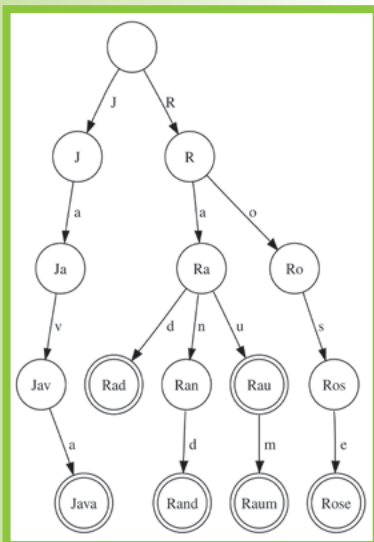


Abb. 2: Trie-Datenstruktur, Quelle: Wikipedia