Produktivität von Programmiersprachen

Tim Jammer

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik, Arbeitsbereich Wissenschaftliches Rechnen
Seminar Softwareentwicklung in der Wissenschaft im SoSe 15

02.06.2015

1. Wie misst man die Produktivität von Programmiersprachen?
   - Performance
     - Speicherbedarf
   - Entwicklungszeit
     - Wartbarkeit / Anpassbarkeit
     - Größe der Community
   - Zusammenfassung
2. Einige Programmiersprachen im Vergleich
   - Die zu vergleichende Problemstellung
   - Die Ergebnisse bezüglich Performance
   - Die Ergebnisse bezüglich Entwicklungszeit
   - Zusammenfassung
3. Fazit

Wie misst man die Produktivität von Programmiersprachen?

Entscheidend sind im Wesentlichen

- Performance
- Entwicklungszeit

[0] vgl. [KK04]

Performance

Unter Performance fallen im Wesentlichen alle Aspekte eines Programms, die *nach* der Entwicklung messbar sind.

- **Laufzeit**
- **Speicherbedarf**
- (Energieeffizienz)
- (Stabilität)
- (Robustheit)

[0] eigene Einteilung von Faktoren aus [KK07]

Laufzeit

- Zeit, die der fertige Code benötigt, um ausgeführt zu werden
- Parallelisierungsmöglichkeiten können Laufzeit verkürzen
- Je Häufiger das Programm ausgeführt werden muss, desto wichtiger ist dieser Faktor im Vergleich zu anderen

[0] vgl. [KK07]

Speicherbedarf

- Speicherbedarf, den das Programm zur Laufzeit hat
- Im Bereich HPC besonders wichtig (Scheduling)

- Speicherplatz, den das Programm selbst einnimmt oft nicht so entscheidend, darf aber nicht vergessen werden

[0] vgl. [ea10]

Entwicklungszeit

Beinhaltet die Zeit, die *während* der Entwicklung verstreicht.
Sie wird u.a. durch folgende Aspekte beeinflusst:

- Erfahrung der Programmierer
- Deren Verständnis vom Problem
- Dem zu lösenden Problem
- Kompilierungszeit
- Benutzen vorhandener Librarys möglich?

[0] vgl. Lehrveranstaltung SWT

Wartbarkeit / Anpassbarkeit

- Zeit, die in die Anpassung der Software gesteckt werden muss
- Darunter fällt auch die Plattformunabhängigkeit

[0] vgl. Lehrveranstaltung SWT

Größe der Community

Durch größere Community gibt es

- mehr erfahrene Programmierer
- mehr Unterstützung bei (technischen) Problemen
- mehr und bessere Tools zur Unterstützung der Implementation
- Standardisierung
- eventuell mehr Librarys

$\Rightarrow$ die Entwicklungszeit kann dardurch verkürzt werden

[0] vgl. Lehrveranstaltung SWT

Zusammenfassung: wie kann man Produktivität messen

Man kann diese Faktoren häufig in Zeiteinheiten messen

$\Rightarrow$ diejenige Sprache ist optimal,

die die geringste Gesamtzeit aufweist(?)

Aber: Gewichtung der Faktoren ändert sich von Problemstellung zu Problemstellung

[0] vgl. [KK07]

Einige Programmiersprachen im Vergleich

- Ergebnisse der Studie von Lutz Prechelt werden vorgestellt[1]
- 80 verschiedene Programme untersucht
- Perl, Phyton, Rexx, Tcl als Skriptsprachen zusammengefasst
  - besondere Rolle: andere Erhebung der Daten

[1] Studie aus [Pre00]

Das betrachtete Problem

Telefonnummern Wörter zuordnen,
um sie sich besser merken zu können

E	JNO	RWX	DSY	FT	AM	CIV	BKU	LOP	GHZ
0	1	2	3	4	5	6	7	8	9

Beispiel: Fort -> 4824

[1] vgl. [Pre00]

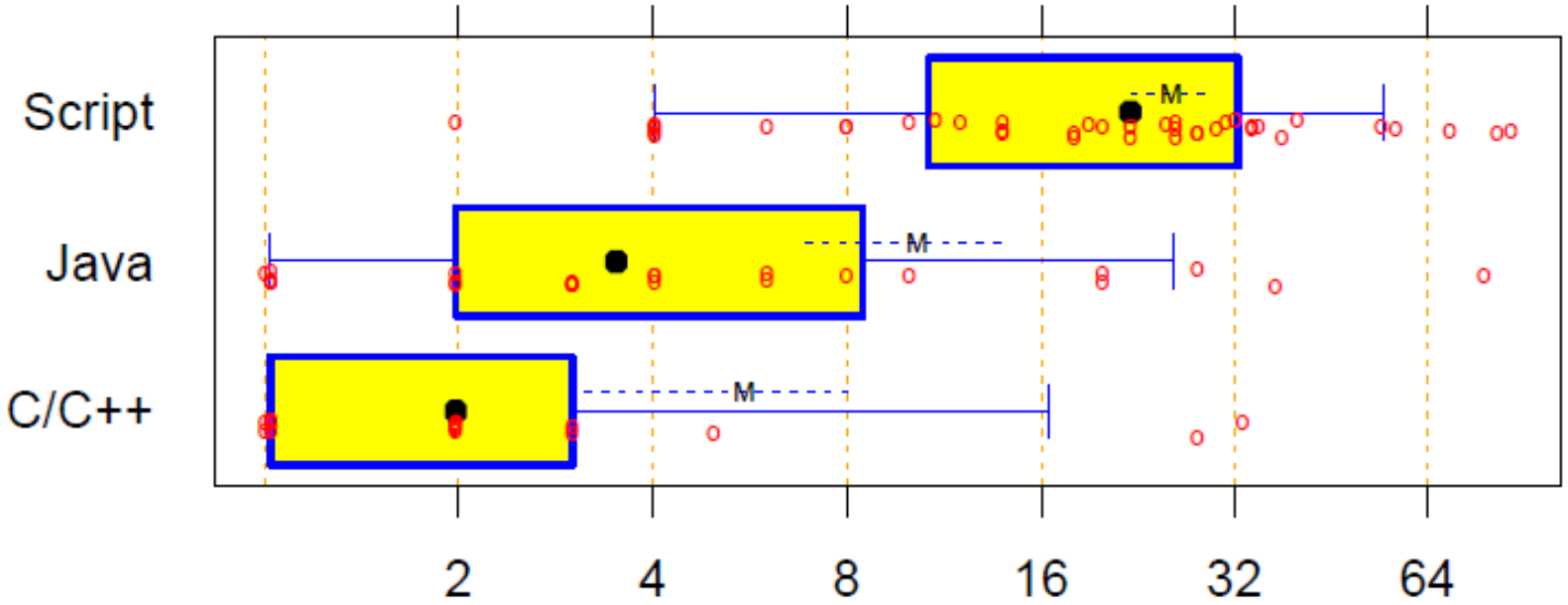

Abbildung: Laufzeit: Laden des Wörterbuches in Sek.

[1] Abb. aus [Pre00]

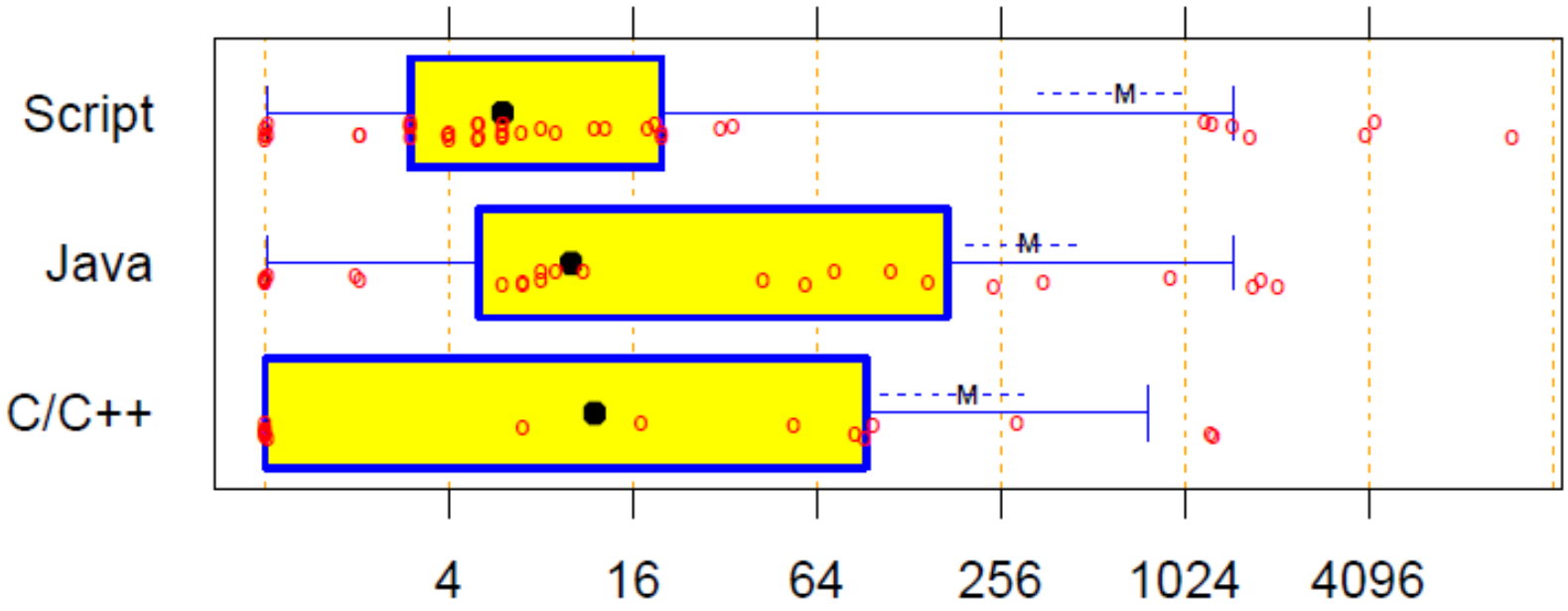

Abbildung: Laufzeit: Suche möglicher Wörter der Telefonnummern in Sek.

[1] Abb. aus [Pre00]

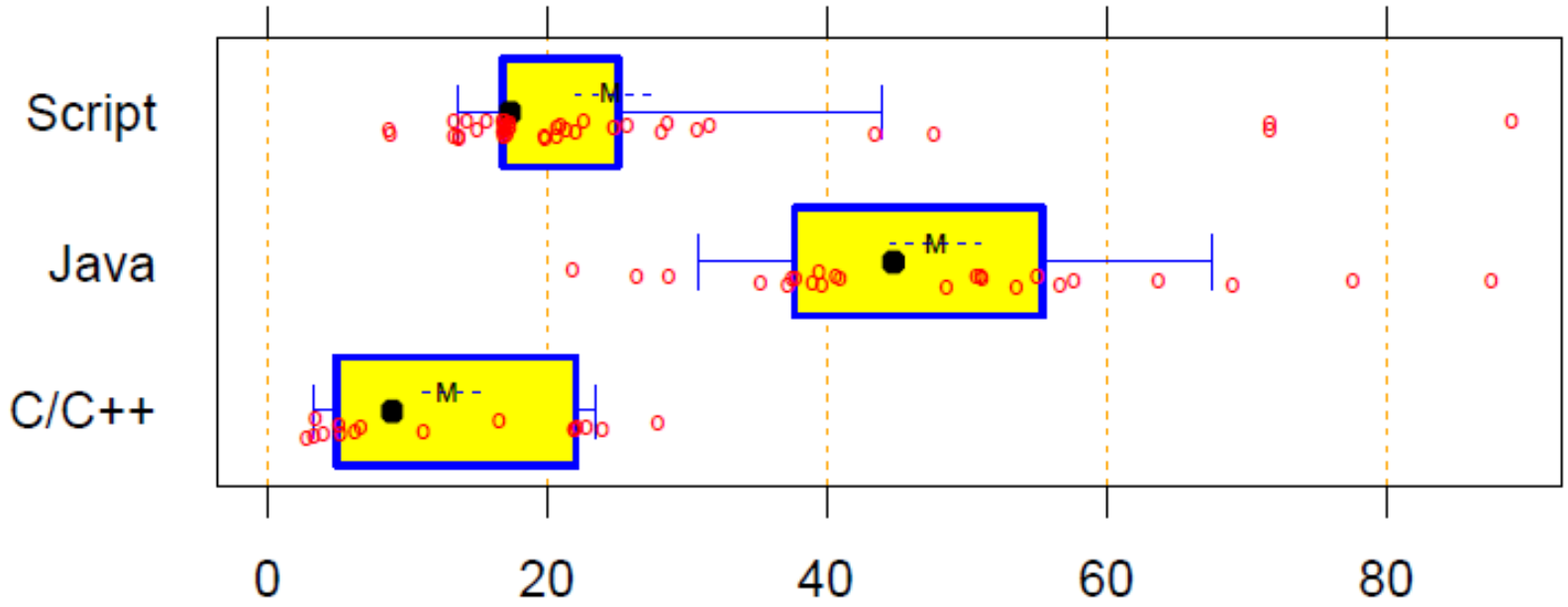

Abbildung: Speicherverbrauch in MB

[1] Abb. aus [Pre00]

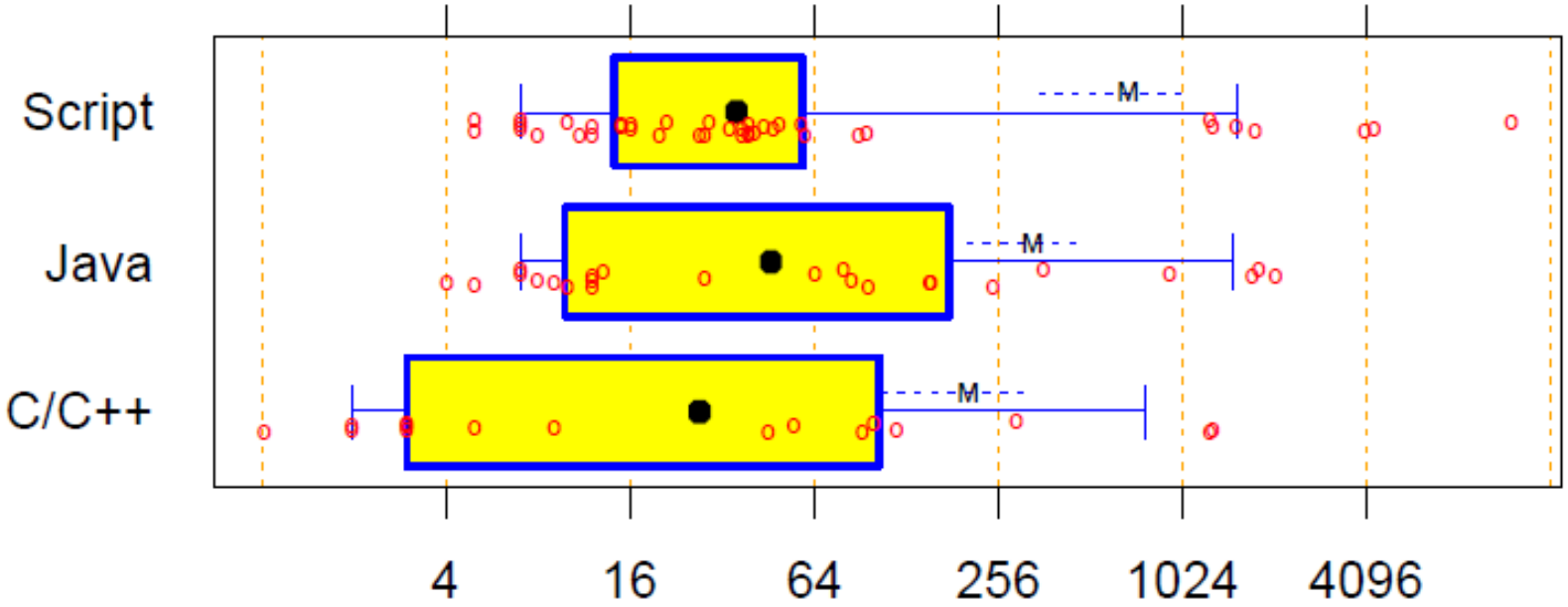

Abbildung: Gesamtlaufzeit in Sek.

[1] Abb. aus [Pre00]

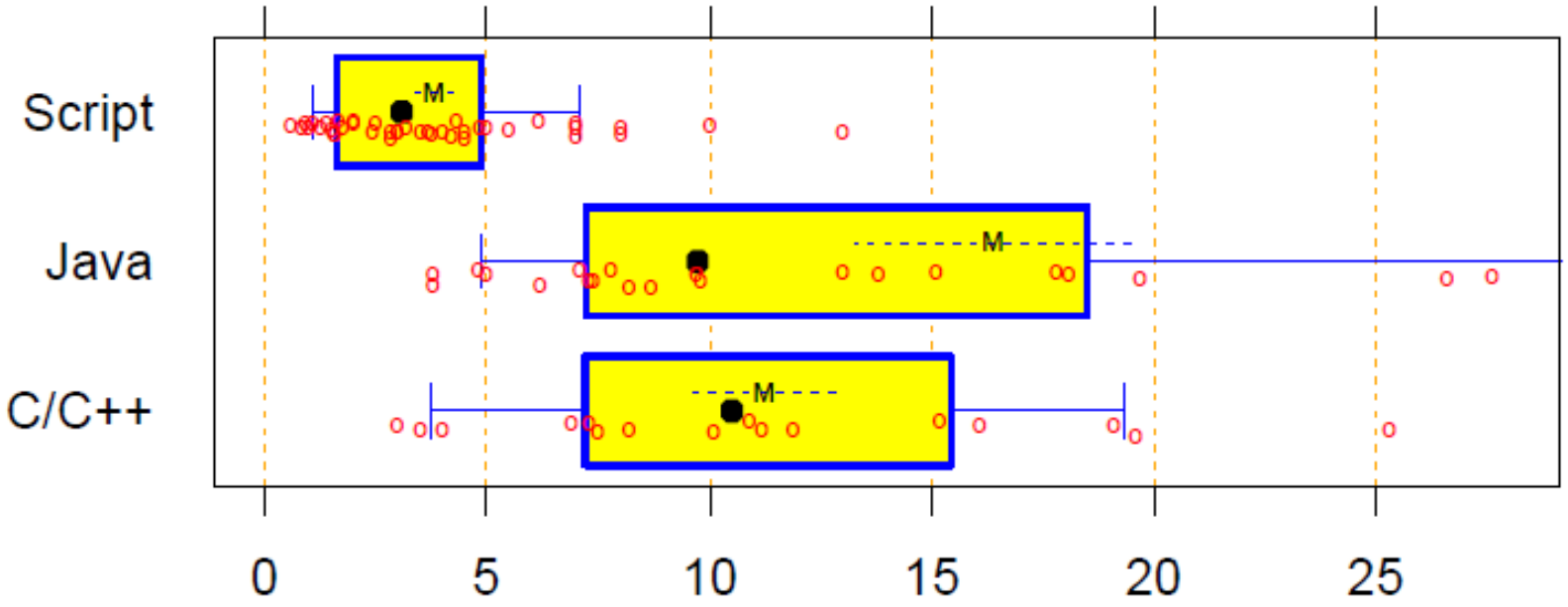

Abbildung: Entwicklungszeit in Stunden

[1] Abb. aus [Pre00]

Zusammenfassung: Ergebnisse der Studie

- Unterschiede zwischen den einzelnen Sprachen:
  - wesentlich weniger stark
- als Unterschiede zwischen
  - den Programmen der gleichen Sprache

- Keine Unterschiede in der Zuverlässigkeit der Programme

[1] vgl. [Pre00]

Fortran

- Fortran als "natural language for expressing sience and engeneering ideas" entstanden
- wurde zu einer DSL weiterentwickelt
  - keine Vererbung
  - Polymorphie

[1] vgl. [VKD07]

Fortran

"Fortran is harder to compete with. It has dedicated following who [...] care little for programming languages or the finer points of computer sience. They simply want to get their work done."
form C++ Designer Bjarne Stroustrup

⇒ Fortran ist (heute) eine DSL

- muss sich daher eher mit Sprachen wie Matlab messen

[1] vgl. [VKD07]

Fazit

- Studierende sind heute "multilingual"

⇒ Herangehensweisen an komplexe Probleme müssen gelehrt werden z.B. design-patterns

- Erlernen einer weiteren Sprache vergleichsweise einfach

Quellen

[ea10] Yoongu Kim et al.
Atlas: A scalable and high-performance scheduling algorithm for multiple memory controllers.
2010.

[KK04] Charles Koelbel Ken Kennedy.
Defining and measuring the productivity of programming languages.
International Journal of High Performance Computing Applications 18, 2004.

[Pre00] Lutz Prechelt.
An empirical comparison of c, c++, java, perl, python, rexx, and tcl for a search/string-processing program.
Technical report, Universität Karlsruhe, Fakultät für Informatik, March 2000.

[VKD07] Henry J Gardner Viktor K. Decyk, Charles D. Nortron.
Why fortran.
Computing in sience & engineering, 2007.

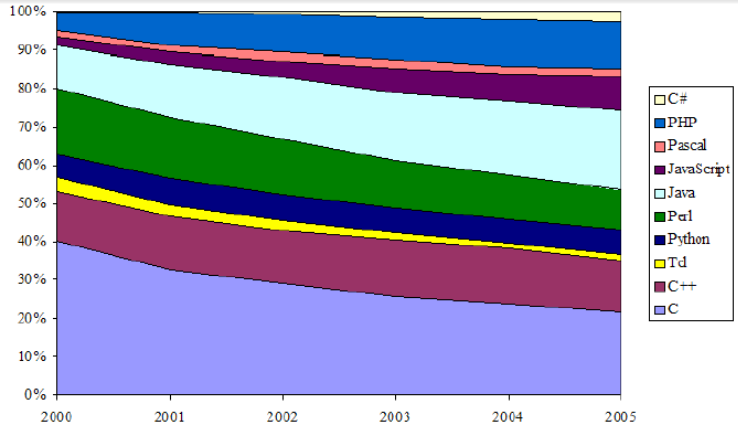

Figure 1: Distribution of projects

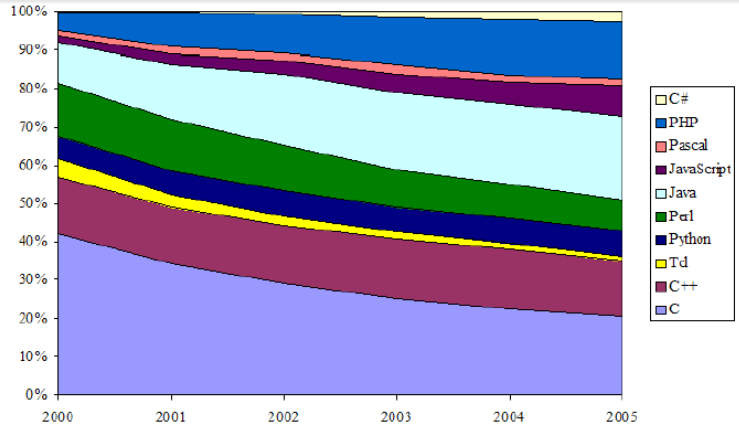

Figure 2: Distribution of authors

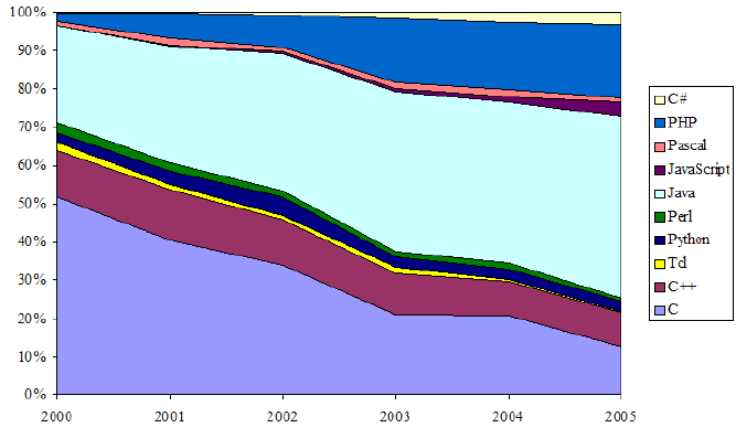

Figure 3: Distribution of files

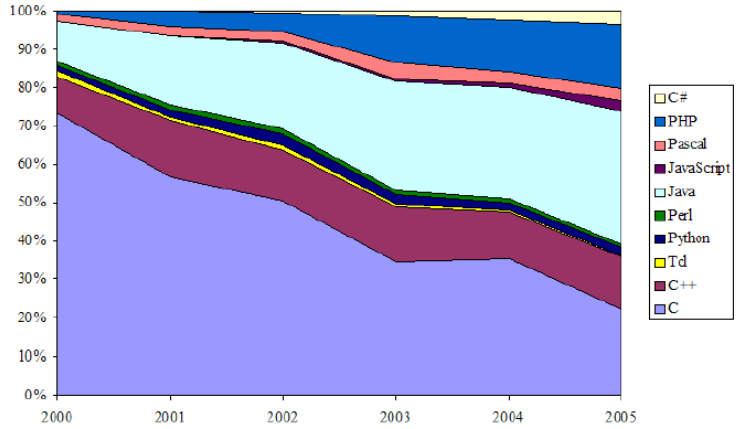

Figure 4: Distribution of lines of code

[1] Abb aus: http://dml.cs.byu.edu/ cgc/pubs/WoPDaSD2007.pdf