

Programmierparadigmen

Kapitel 1 Einführung



Was ist ein Paradigma?

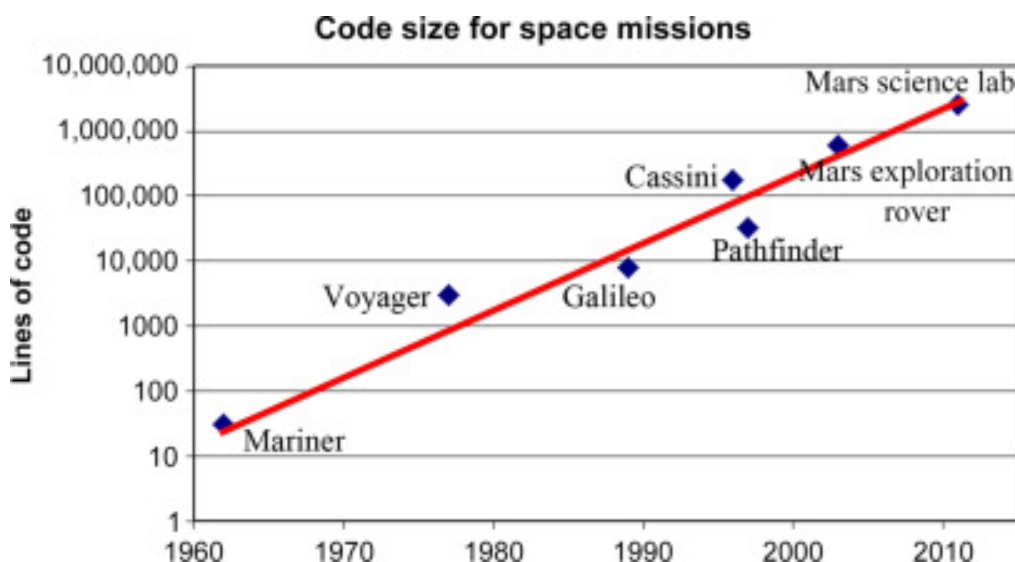
- Paradigma – aus dem Altgriechischen *Beispiel, Muster; Erzählung mit beispielhaftem Charakter* (laut Duden)
- Programmierparadigmen beschreiben grundsätzliche Arten wie Computer-Programme formuliert werden können
- Programmiersprachen können einzelne oder viele Konzepte aufgreifen
 - Keine verbreitete Sprache greift alle behandelten Konzepte auf
→ Betrachtung unterschiedlicher Sprachen
- Ziel der Veranstaltung: Weiten der in der Vorlesung „Programmierung und Algorithmen“ (PuA) eingeführten Sichten hin zu einem Werkzeugkoffer zur Lösung realer Probleme...



“The major cause [of software crisis] is... that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming had become an equally gigantic problem.”

Edsger W. Dijkstra
1972

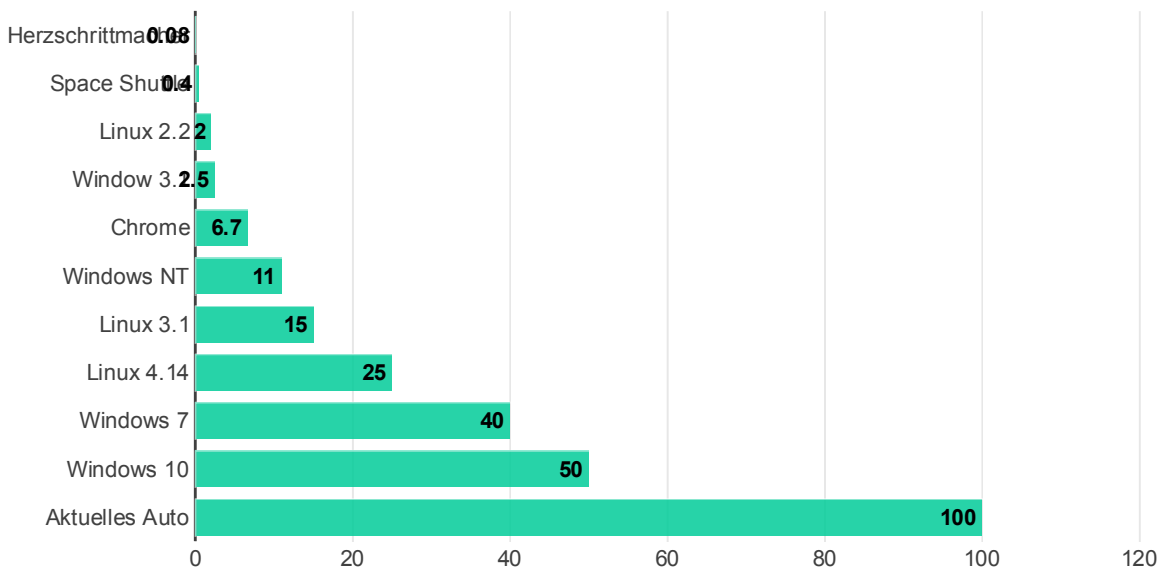
- Komplexität von Software schlecht beherrschbar...



From: Abella, Cazorla. *Harsh computing in the space domain*.



Anzahl von Code-Zeilen



<https://informationisbeautiful.net/visualizations/million-lines-of-code/>



Was bedeutet das?

- Programmierer schreiben, testen und dokumentieren zwischen 325 und 750 Codezeilen pro Monat
→ maximal 360.000 Zeilen in 40 Jahren!
- Komplexität muss verborgen werden, z.B. durch
 - Kapselung
 - Spezifische Sprachkonstrukte, Domain Specific Languages
 - Ausdrucksstärkere Sprachen
- Entwicklung neuer Programmierparadigmen hilft Grenzen (ein wenig) zu verschieben
- Theoretische Rahmenbedingungen (Turing-Mächtigkeit, Satz von Rice) behalten Gültigkeit!



- Aus Vorlesung PuA:
 - Imperative Algorithmen
 - Applikative Algorithmen
 - Deduktive Algorithmen
- Aber Vielzahl weiterer Formen
 - Teilweise ergänzend, unterschiedliche Kategorisierung möglich
 - Beispiele: prozedural, deklarativ, objektorientiert, datenstromorientiert, parallele & verteilte Programmierung...
- Teilweise unterschiedliche Bezeichnungen
 - Applikativ bzw. Funktional
 - Deduktiv bzw. Logisch
- Aktueller Trend: Multiparadigmen-Sprachen
 - Umsetzung unterschiedlichster Paradigmen in einer Sprache
 - Beispiele: Scala, neuere C++-Standards, ...



- Vorlesung orientiert sich an den folgenden Fragen:
- Wie kann die Komplexität von Software beherrscht werden?
- Wie kann komplexere Software nachweisbar korrekt konstruiert werden?
- Wie können viele Aufgaben/Daten verarbeitet werden (ohne dass Softwarekomplexität zu stark steigt)?
- Wie können Aufgaben über Systemgrenzen hinweg skalieren (ohne dass Software zu komplex wird)?

→ Leitfragen die sich durch die Vorlesung ziehen werden...



- Grundkonzepte zu
 - Objekt-orientierter Programmierung
 - Funktionaler Programmierung
 - Paralleler Programmierung
 - Verteilter Programmierung
- Am Beispiel von Java, C++ & Erlang
- Hier nicht behandelt:
 - Logische Programmierung → eigene Veranstaltung „Logik und Logikprogrammierung“
 - Hardware-Beschreibungssprachen → „Entwicklung integrierter HW/SW Systeme“

