Softwaretechnik Übung: Statische Programmanalyse

Klaus Kusche

1.) Sicherheitsanalyse

Die Hauptwerkzeuge in diesem Bereich sind *Datenfluß-basierte Tools*, von denen wir schon einige in der Übung zu Pointer- und Array-Fehlern kennengelernt haben.

Auf unseren Systemen sind aber auch zwei Tools installiert, die ausschließlich mit *Pattern Matching* arbeiten: **`rats`** und **`flawfinder`**.

Probiere sie kurz mit einigen C-Programmen. Hältst du solche Tools für nützlich?

2.) Code-Metriken

Auf unseren Systemen sind drei Tools installiert, die Code-Metriken berechnen:

- **`cccc`** (siehe HTML-Doku):

  `cccc` analysiert primär größere C++-Projekte.
  Leider wird das Programm seit Jahren nicht mehr gewartet
  und hat Probleme mit der Syntax-Analyse von aktuellem C++-Code.

  Versuche, es auf die Musterlösung von Beispiel 18a des 2. Semesters anzuwenden.
  Lies den generierten HTML-Report.

  - Kannst du die ungefähre Bedeutung der Zahlen erkennen?
  - Wie weit entspricht die Kennzahl "MVG" der "intuitiven" Komplexität der jeweiligen Funktion?

- Ein weiteres Komplexitätsmaß berechnet **`complexity`**. Leider kennt es nur C-Code, kein C++, lass es daher mit unserem **`xref-tree-bal.c`** laufen (Optionen **`-t 1 -s 1`** bzw. **`-h -s 1`**).

  Liefert es eine brauchbare Maßzahl für die Code-Komplexität?
  (Achtung: **`-s 1`** bewirkt, dass die ausgegebenen Komplexitätswerte 20 Mal größer sind als die Referenzwerte in der Doku!)

- **`cloc`** (mit **`--by-file`**) ist viel primitiver. Was liefert es?