On the Impact of Programming Languages on Code Quality

Typ	Tidskriftsartikel
Författare	Emery D Berger
Författare	Celeste Hollenbeck
Författare	Petr Maj
Författare	Olga Vitek
Författare	Jan Vitek
Band/Årgång	1
Nummer	1
Sidor	23
Bibliotekskatalog	Zotero
Språk	en
Tillagd den	2019-09-02 16:26:33
Ändrad den	2019-10-07 12:36:26

Etiketter:

Software quality, Programming languages, Research methodology, Statistics

Bifogade dokument

- Berger m. fl. - On the Impact of Programming Languages on Code Qua.pdf

  En tidigare studie av Ray et al. kom fram till att:

  1. Av 17 undersökta programmeringsspråk hade 11 stycken antingen ett positivt (C, C++, Objective-C, JavaScript, PHP, Python) eller negativt (TypeScript, Clojure, Haskell, Ruby, Scala) samband med antalet buggar.
  2. Program skrivna i funktionella språk har lite färre defekter än procedurella och skript-dito.
  3. Det finns inget generelllt samband mellan tillämpningsområde och bugbenägenhet.
  4. Det finns ett starkt samband mellan bugtyper och språk.

  Ray et al. presenterar detta som korrelationer, men många har senare tolkat det som kausaliteter (språk -> #buggar). Ray et al. kompenserar för projektens ålder, antal commits, antal utvecklare och antal rader committad kod.

  Den här studien försöker repetera experimenten och dessutom reproducera (=inte använda exakt samma metodologi) slutsats nummer 1 ovan.

  Vid repetetionen hittade författarna inte samma mängd projekt, kodrader o.s.v. som orginaltudien, trots att de använde samma metodik. När det gäller orginalstudiens resultat observerade den här studien följande:

  1. kunde repeteras kvalitativt (signifikans för samma språk) men inte exakt (samma värden och signifikansnivåer)
  2. Författarna var inte överens med orginalstudien när det gäller kategorisering av språk, men när de använde denna kategorisering lyckades de repetera resultaten. (De menar dock att dessa är meningslösa eftersom kategoriseringen är intetsägande.)
  3. Inte heller dessa författare hittar något samband mellan faktorerna. (Till skillnad från ovan har de inte haft tillgång till orginalstudiens kod här, utan backwards-engineerat.)
  4. Här får de inte ens samma resultat när det gäller antalet buggar av respektive typ.

  Vid reproduceringen tog de bort duplicerade commits och uteslöt ett språk som visade sig inte ha särskilt många commits. (Många filer var felklassificerade.)

  Vidare konstaterar de att orginalstudiens metod för att klassificera commits som bugfixar är bristfällig. De uppskattar att den ger 36% false positives och 11% false negatives. Den här studien använder dock samma klassificeringsstrategi (i brist på annan praktiskt genomförbar dito), men kompenserar för det i den statistiska analysen genom att bootstrappa (=göra ett permutationstest).

  De gör även annat annorlunda i den statistiska analysen. Bland annat minskar de family-wise error rate med Bonferroni (-> mycket lägre signifikanströsklar för respektive test) och en annan liknande procedur.

De hittar bara signifikanta språk-bug-samband för C++ (positivt) samt Clojure och Haskell (negativa).

De räknar inte bara ut konfidensintervall, utan också "prediction intervals", som förutsäger den praktiska effekten av språk på framtida bugmängder. Dessa överlappar för C++ och Clojure, språken med störst positivt respektive negativt samband med antalet buggar. De drar slutsatsen att den praktiska skillnaden mellan olika språk är minimal.

De listar också ett antal anledningar till att man inte kan dra några kausala slutsatser av resultaten:

- 16% av filerna är testfiler. Dessa kan ha andra typer av buggar än "produktionskod".
- Deras korrigering för felklassificerade buggar förutsätter att klassificeringsfelen är uniformt fördelade mellan språken, vilket de inte verkar vara.
- Urvalet av projekt är sannolikt inte helt representativt.
- Studien täcker inga kommersiella projekt.
- Tillämpningsområde påverkar troligtvis bugfrekvensen, och det finns troligtvis andra "confounding factors".

De rekommenderar följande som best practices för andra forskare:

1. Automatisera, dokumentera och dela det du gör!
2. Involvera domänexperter för att kontrollera dina antaganden.
3. Kontrollera ditt data, verkar det korrekt och rimligt?
4. Stirra dig inte blind på p-värden. Dessa säger inte allt, framförallt inte om vikten av dina resultat.

Min kommentar: Det är viktigt att notera att det faktum att författarna inte hittar signifikanta samband för vissa språk inte nödvändigtvis innebär att sådana inte finns. Styrkan är troligen ganska låg, inte minst med tanke på deras FWER-strategi.