EXAMENSARBETE Hur påverkar refaktorisering prestanda i Java?
STUDENT Adam Ohlsson
HANDLEDARE Christoph Reichenbach (LTH)
EXAMINATOR Niklas Fors (LTH)

Refaktorisering: Kvalitet och prestanda

POPULÄRVETENSKAPLIG SAMMANFATTNING **Adam Ohlsson**

Vi har begränsad förståelse för hur refaktorisering, en teknik inom mjukvara, påverkar prestanda. Därför utvärderar vi prestandan hos 12 vanliga refaktoriseringstyper.

Refaktorisering är en teknik som används inom mjukvaruutveckling för att upprätthålla mjukvarans kvalitet över tid. Det handlar alltså om att omorganisera programkoden för att vinna fördelar, till exempel för att göra den enklare att förstå, och därmed påskynda vidareutveckling, eller för att den ska bli effektivare, till exempel bära mer last, bli snabbare eller ta mindre plats.

I syfte att utöka förståelsen för hur refaktorisering påverkar prestanda bidrar vi med en prestandaundersökning där vi utvärderar effekten av isolerade refaktoriseringar på körtid. Vi utvärderar 12 vanliga refaktoriseringstyper och ett urval av dess parametrar på 12 representativa prestandatester. Vår studie visar att refaktorisering inte påverkar snittprestanda men att det finns refaktoriseringar som kan ha en betydande effekt.

Tyvärr så är det svårt både att ta kodmått och att bedöma kodens kvalitet utifrån mätetal vilket gör att programmerare i praktiken förlitar sig mer på erfarenhet, konventioner, inlärda mönster och vanor än på mätdata i beslutsfattandet om kodens utformning.

Eftersom refaktorisering är en teknik med bred tillämpning inom industrin och eftersom utövaren behöver erfarenhet och avancerad kunskap för att förstå de avvägningar som kan behöva göras för att ta rätt design-beslut finns det en viss oro för att man suboptimerar kodens prestanda genom att, till exempel, optimera kodens läsbarhet, något som är förhållandevis svårt att mäta. Samtidigt binds man att tillämpa projektets antagna kodkonventioner, regler som man använder för att göra koden mer homogen, vid till exempel namngivning av kodelement eller för att främja särskilda strukturer och syntaktiska konstruktioner som anses vara snabbare, säkrare, mer stilenligt korrekta eller mer flexibla och programmatiskt användbara än alternativen. Det snabbaste alternativet är inte nödvändigtvis det som är mest flexibelt eller lättast att läsa.

Java är ett komplext programmeringsspråk där man tillämpar avancerade teknologier för att optimera koden medan programmet körs, vilket gör det extra svårt i praktiken att mäta prestanda. Man måste tillämpa en robust metodologi med många parametrar för att få ut mätdata av hög kvalitet. Dessutom har vi begränsad förståelse för kodattribut, de kodmått som representerar dem och hur de påverkar varandra.

Idag finns det begränsat vetenskapligt underlag för de teorier som har etablerats kring refaktorisering, kvalitet och prestanda. Vi tror att resultat på det här området kan hjälpa både språkutvecklare, programmerare, och kodägare att få en bättre förståelse för språket, dess styrkor och svagheter.