INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2026-06-04

EXAMENSARBETE Test-Driven Development as a method for AI-generated code
STUDENTER Emelie Tingberg, Victor Sannicolo
HANDLEDARE Per Runeson (LTH), Ivan Aladjoff (Decerno AB)
EXAMINATOR Björn Regnell (LTH)

Testdriven utveckling i händerna på AI, vad händer när ingen människa har koll?

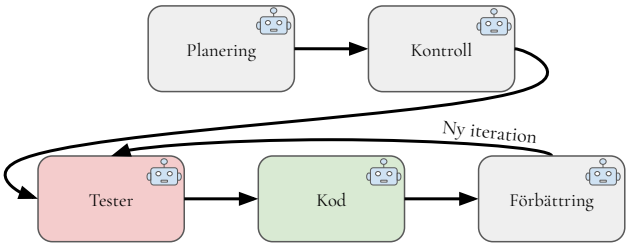
POPULÄRVETENSKAPLIG SAMMANFATTNING **Emelie Tingberg, Victor Sannicolo**

Kan AI skriva kod helt på egen hand, utan att någon kontrollerar vad den gör? I det här examensarbetet byggde vi ett system där AI genererar kod enligt testdriven utveckling (TDD), helt utan mänsklig övervakning. Resultaten är överraskande: TDD förändrar typen av problem i den AI-genererade koden, men löser dem inte helt.

Tänk dig att anställa en programmerare som aldrig frågar om hjälp, aldrig tar rast och jobbar dygnet runt. Det är i princip vad ett automatiserat AI-system för mjukvaruutveckling kan erbjuda. Företag använder redan AI för att generera kod automatiskt, men hur bra är den koden egentligen, och går det att skapa struktur i processen?

En vanlig utmaning med AI-genererad kod är att bibehålla kvaliteten. Koden visar sig ofta vara svår att underhålla, onödigt komplicerad eller rentav felaktig. Testdriven utveckling är en välbeprövad metod inom mjukvaruutveckling där man skriver tester innan koden skrivs. På så sätt fungerar testerna som regler som koden måste följa. Metoden är designad för mänskliga utvecklare, men vi frågade oss vad som händer om AI följer samma process helt på egen hand.

Vi implementerade ett system i fem faser: planering av uppgiften, kontroll av planeringen, generering av tester, generering av produktionskod och förbättring. Till höger visas en överblick av systemet. AI genererar både tester och produktionskod utan mänsklig övervakning. Systemet testades i ett verkligt scenario där ett webbprojekt migrerades, och resultaten jämfördes mot ett system för kodgenerering utan TDD.



Resultaten tyder på att TDD förändrar typen av problem i koden snarare än att eliminera dem. TDD-koden hade lägre komplexitet och mindre duplicerad kod, men totalt fler kvalitetsproblem. Icke-TDD producerade färre problem totalt, men de som uppstod var generellt sett allvarligare. Hur man värderar detta beror på sammanhanget: ibland är det viktigare att undvika de allvarligaste felen, och ibland att minimera antalet fel. En oväntad iakttagelse var att TDD ofta genererade tester som bara kontrollerade att innehåll existerade snarare än att testa faktiskt beteende, trots tydliga instruktioner om motsatsen.

Slutsatsen är att TDD är möjligt att tillämpa i ett autonomt AI-sammanhang, men att metodens strikta principer skapar utmaningar när ingen människa är involverad. Resultaten tyder på att en modifierad och mer flexibel version av TDD kan vara vägen framåt.