Er ”kopier og lim”-programmering skadelig?

Når en systemutvikler kopierer en del av programvarekoden og limer denne inn et annet sted i samme programvaren kalles dette kloning eller duplisering av programvarekode. Duplisering av kode er en utbredt praksis og studier tyder på at typisk 10-15% og i noen tilfeller helt opp til 50% av programvarekoden i større IT-systemer er generert på denne måten.

Duplisering av kode har lenge vært ansett som en praksis som gir en reduksjon av utviklingstid på kort sikt på bekostning av økt feilhyppighet og høyere vedlikeholdskostnader på lang sikt. En slik oppfatning er forståelig siden systemutviklerne ved duplisering ofte må oppdatere samme logikk i kode to steder i stedet for ett. Glemmer man ett av stedene oppstår det en feil i programvaren. I tillegg kommer at programvarekoden ved duplisering blir større og dermed mindre oversiktlig. I Martin Fowler, en av grunnleggerne til ”smidig” utvikling, sin liste av indikatorer på dårlig kode, såkalt ”code smells”, inngår duplisert kode som ”*number one in the stink parade*”. En vanlig anbefaling er å trekke ut duplisert kode og erstatte denne med felles metoder eller klasser.

Hvorfor fortsetter så systemutviklere med å duplisere kode? Er det fordi de er late? Er det fordi de er presset på tid slik at de langsiktige problemene kommer i andre rekke? Er det fordi de ikke er kompetente nok til å våge å endre den eksisterende koden og i stedet skriver ny? I noen tilfelle er nok latskap, kortsiktighet og inkompetanse hos systemutviklere årsaken til duplisering av kode, med den konsekvensen at programvaren får redusert kvalitet. I det store og hele synes imidlertid dupliseringen å være et mye mindre problem enn mange lenge har trodd. Det er til og med grunn til å tro at duplisering av kode i mange tilfeller er gunstig for kvaliteten til IT-systemene!

Jeg gikk gjennom ti forskningsstudier fra 2011 til 2013 som sammenlignet duplisert programvarekode med annen programvarekode. Nesten uten unntak fant studiene at duplisert kode i gjennomsnitt var like god som eller bedre enn ikke-duplisert kode. Noen smakebiter. I en studie ved Simula Research Laboratory og Universitetet i Oslo fant Yamashita, Anda og Sjøberg at det ikke var noen statistisk signifikant forverring i vedlikeholdbarhet til duplisert kode. Studien til Harder og Göde (”Cloned code: stable code”) rapporterer at duplisert kode var mer stabil, det vil si hadde mindre behov for endringer, enn annen kode. Rahman, Bird og Devanbu (”Clones: What is the smell?”) fant at det var færre og mindre arbeidskrevende feil i duplisert kode.

Tar da Fowler og mange med han feil med hensyn til at duplisering av kode er ”*number one in the stink parade”*? For å svare på dette spørsmålet er nyttig å skille ulike typer dupliseringer fra hverandre, noe Kapser og Godfrey gjør i sin artikkel ””Cloning considered harmful” considered harmful: patterns of cloning in software”. Forfatterne observerer at motivene for å duplisere kode ofte verken er latskap, kortsiktighet eller inkompetanse, men tvert i mot et ønske om å øke kvaliteten på programvaren. En type duplisering (”forking”) er den man gjør når original og duplisert kode antas å utvikle seg ulikt. Dette er annerledes fra typen duplisering (”templating”) der det er samme krav som skal dekkes og man må anta at duplisert og original kode vil måtte utvikle seg likt. Forfatterne finner at mens ”forking” type duplisering stort sett er bra, så er mange varianter av ”templating” mye mer tvilsomme. Totalt sett fant de at 71% av duplisert kode var godartet.

Vi står dermed tilbake med at svaret på om duplisering av programvarekode er bra eller ikke er at ”det kommer an på”. En systemutvikler vil blant annet måtte vurdere hvordan fremtiden for den dupliserte koden vil se ut, om nytten av å ha en mer lesbar men mindre generell kode oppveier ulempen med å måtte huske å oppdatere mer enn ett sted og hvor viktig det er å unngå feil i kjernedeler av programvaren. Forskningsfunnene tyder på at de fleste systemutviklere klarer dette ganske bra. Det finnes likevel mange eksempler på at kode dupliseres av latskap, kortsiktighet eller inkompetanse. Regelmessig gjennomgang og forbedring (refactoring) av programvarekode for å unngå ondartet duplisering er derfor fortsatt viktig. Vi kan imidlertid ikke bruke grad av duplisering i et system som en måling på kvaliteten til systemet og det er langt fra noen automatikk i at duplisert kode bør endres. For å si det med Kapser og Godfrey’s respons til Fowler’s påstand om at duplisering er den verste av alle ”bad code smells”: ”Perhaps we can clone, and breathe easily, at the same time.”