Investeren in hergebruik

Ontwikkelaars verzamelen en hergebruiken codefragmenten. Al sinds Fortran kan dat systematisch met library's. C bestaat zelfs voor een groot deel uit een bibliotheek (LibC) met functies als printf(), een complex stuk code van meer dan zevenhonderd regels. Er zijn weinig ontwikkelaars die zelfs maar overwegen om een dergelijke routine zelf te programmeren. Via printf() gebruiken ze deze code telkens opnieuw.

Niemand spreekt echter over hergebruik als ze printf() aanroepen. Het is *normaal* gebruik. Dat geldt ook voor bijna alle andere stukken code die op deze – overigens correcte – manier in meerdere projecten of producten terechtkomen. Zowel bij ingekochte functionaliteit, 'geleende' opensource, als voor bedrijfseigen code geldt: als het beschikbaar is als een library, dan gebruiken we het gewoon.

Hergebruik is een na te streven maar utopisch doel. Ook is het een prima verkoopargument. Talloze dure adviezen zijn aan de man gebracht door in te spelen op hergebruik. Bladen en artikelen over het onderwerp doen het goed. Zelfs slechte softwarepraktijken worden zo 'verkocht': het klonen van code heet dan plots hergebruik.

Maar wat is hergebruik eigenlijk precies? Wikipedia geeft enige duidelijkheid: 'besparen door het meermaals gebruiken van materialen, liefst nadat ze als afval zijn bestempeld'. Hoe moeten we dat vertalen naar software? Het lijkt mij onwaarschijnlijk dat we oude bits gaan inzamelen.

Ook code klonen, het ongecontroleerd knippen en plakken van code, leidt niet tot hergebruik. Het bespaart een beetje tikwerk, maar geeft niet minder afval en bespaart niet echt. Als we bugs als codeafval erkennen, zien we het probleem. We kopiëren namelijk niet alleen code, maar vermenigvuldigen ook de hoeveelheid afval. Al die kopieën moeten we testen en waarschijnlijk moeten we vooral meer bugs verwijderen.

Een belangrijk argument voor hergebruik is kostenbesparing. Het kan goedkoper zijn om materiaal te hergebruiken dan het opnieuw te maken. Zo is het aanroepen van de printf()-functie 'goedkoper' dan haar vanaf scratch herprogrammeren. We moeten ons echter wel goed realiseren wat 'goedkoper' in deze context betekent. Behalve dat het geheugenruimte bespaart (weer een argument om klonen af te raden), kost het vooral minder ontwikkeltijd, zowel voor de programmeur als voor de tester. De kans dat er een fout zit in printf() is zo klein dat we hierop eigenlijk nooit (expliciet) testen. Dat geldt ook voor andere code. Het ontwikkelen van een library kost tijd; het gebruik ervan bespaart tijd, vooral omdat het testen sneller kan.

Albert Mietus noemt zich embedded-R&D-architect en is herbruikbaar op de arbeidsmarkt.

In de praktijk kost het ontwikkelen van herbruikbare code meer tijd dan van gewone code. Bijvoorbeeld omdat we van librarycode (terecht) een hogere kwaliteit eisen en omdat gebruikersdocumentatie essentieel is. Een manager die de keus heeft om nieuwe code specifiek te ontwikkelen voor zijn project of om die generiek te maken, heeft dus een dilemma. Zelfs als hij, gedreven door technische overwegingen, de voorkeur heeft voor een library, zal de economische werkelijkheid hem dwingen richting de goedkoopste oplossing: alleen die code en documentatie die nú belangrijk is. Een populaire aanpak als Scrum en de huidige recessie versterken deze trend.

Wie echt aan hergebruik wil gaan doen, zal over projecten heen moeten kijken. Bijvoorbeeld door projecten te laten betalen voor hergebruikte code, die ze nu 'gratis' krijgen. Alleen dan kunnen projecten er ook voor kiezen om te investeren in hergebruik. De extra kosten verdienen we terug als andere projecten ervoor kiezen die library's te hergebruiken. Iedereen gaat er dan op vooruit.

Behalve een andere projectaansturing vereist dit ook een andere praktijk van ontwerpen. Nu denken we in het ontwerpproces nauwelijks na over hergebruik. Ook hier zullen programmeurs *out of the box* moeten gaan. Naast technisch correct kun je het ontwerpen ook economisch benaderen. Willen we alle onderdelen specifiek ontwikkelen of willen we een ontwerpvariant die vooral bestaande code hergebruikt? Of wellicht wil het project juist investeren en nieuwe library's opleveren? Daarvoor is samenwerking tussen managers en ontwerpers noodzakelijk.

Economisch gezien heeft hergebruik pas zin als software een kostprijs heeft. Helaas is ooit bedacht dat software niets kost, dus hoe zou je daarop kunnen besparen? Nog steeds doen we of software gratis is. Ik ken geen één projectmanager die bereid is om code te kopen van een ander project. Zolang we niet willen betalen voor hergebruikte code, zal niemand daarin willen investeren. En blijft codehergebruik een utopie.