toolview

Albert Mietus'

toolview

Elke ingenieur gebruikt tools en iedereen heeft zijn eigen voorkeur; vaak gebaseerd op wat er toevallig als eerste beschikbaar was. Zo ook de ICT-ingenieur. Ook hij gebruikt tools, software-tools wel te verstaan. Daarvan zijn er oneindig veel, waardoor kiezen erg moeilijk is. Veel handige tools worden niet gebruikt, eenvoudigweg omdat niet bekend was dat er een tool is, specifiek voor dat probleem.

Het leren gebruiken van een tool kost tijd. Maar als je de naam en het doel van het tool kent, loont het gebruik al vaak de moeite. Het inleren en inzetten kost vaak minder tijd dan aanmodderen zonder goed tool.

Deze rubriek stelt dergelijke handige tools voor; soms modern, soms uit de oude doos. Maar altijd bruikbaar!

ALbert dot Mietus at PTS dot nl
http://albert.mietus.nl/read.IT

door & more

De laatst bekende versie van Metre, geschreven door Paul Long, is versie 2.3. Die versie stamt uit 1995. Die werkt op bijna alle systemen, van Unix tot VMS. En natuurlijk de PC, toen nog met DOS.

De licentie is een beetje krom; je mag het gratis gebruiken, zolang je er maar niet echt aan verdiend. Je mag het verspreiden en je mag het veranderen. Maar veranderde versies mogen klaarblijkelijk niet verspreid worden.

Om toch de 2.3X versie beschikbaar te maken, met een hack voor XML output (niet door mij!), heb ik die als patch op mijn website gezet

De code, de Makefiles en de patch zijn niet goed gedocumenteerd. Maar bruikbaar! De patch is free en open; verbeter en verspreid naar believen.

Kortom, voor versie 2.3 en 2.3X, kom naar http://albert.mietus.nl

Metre

Software schrijven wordt vaak als complex gezien. Waarmee bedoeld wordt dat het moeilijk is. Maar geldt dat voor de doe-het-zelver of voor de professionele ontwikkelaar? Complexiteit is relatief, maar ook betrekkelijk. Zo heeft elk algoritme een inherente complexiteit; dat zegt iets over de runtime-efficiëntie. Maar ook elk stukje code, elke functie, kan zowel eenvoudig als ingewikkeld opgeschreven zijn. Die cyclomatische complexiteit is objectief en kan met Metre gemeten worden.

Hello World.c

Dit standaard voorbeeld is een zeer eenvoudig programma; de cyclomatische complexiteit is één. Deze laagst mogelijke score komt omdat er geen if-statements, geen lussen, etcetera in de code zitten. Het programma is rechtlijnig, er zitten geen beslissingen in. Dus is de complexiteit laag. Voegen we bijvoorbeeld een for-statement toe, dan wordt de complexiteit twee. Dat gebeurt dan ook typisch in deel twee van de cursus C.

Beslissen

Een lang programma is niet persé complexer dan een kort. Zo zal het aantal benodigde testen niet toenemen als we een eenvoudig statement toevoegen. Maar bij elke if-then-else moet het programma beslissen of het linksom of rechtsom gaat. Daarom zijn er minimaal twee testen nodig. De complexiteit neemt toe, omdat een beslismoment opgenomen is. Ook voor een switch-case geldt dat. En voor elke lus.

Hoe meer beslismomenten, hoe complexer de code. Als die complexiteit, uitgedrukt in een getal, boven de 25 komt, wordt de code vaak gezien als te complex. In veel code standaards is daarom besloten dat de cyclomatische complexiteit onder de 20, of hooguit 30 moet blijven.

Bugs fiksen of maken

Van elk stuk code is het eenvoudig om de complexiteit te bepalen. Even pre-processen en dat door Metre laten analyseren. Naast de complexiteit worden nog veel meer kentallen berekend. Ook die kun je gebruiken om je eigen code objectief te beoordelen. Maar ook voor code die je moet onderhouden.

Zo heeft een studie van McCabe aangetoond dat de kans dat een verandering een bug introduceert gerelateerd is aan de complexiteit. Een complexiteit van minder dan 10 is geen probleem (kans: 5%). Maar bij een complexiteit van 50 is de kans op een fout al 40%! Bij een complexiteit van meer dan 100, is de kans op een fout groter dan de kans op een verbetering!

McCabe, Myers en Halstead

Bij het berekenen van de complexiteit moeten een aantal keuzes gemaakt worden. Zoals hoe belendende cases geteld worden. Of hoe een ingewikkelde if-expressies meetelt; McCabe heeft daar andere ideeën over dan Myers. Het is daarom prettig dat Metre ook Myers' extended complexity berekend.

Daarnaast wordt de omvang in regels code & commentaar, aantal functies & identifiers, etcetera weergegeven. Altijd handig. Maar ook Halstead's lengte, volume en zelfs een taxatie van de programmeer-inspanning en -tijd kunnen worden berekend. Dat laatste klinkt futuristisch en complex. Maar de theorie is er en de complexiteit daarvan weer van een geheel andere orde.