Bericht uit Java

We kennen allemaal de verhalen van de programmeur bij de bank die het restant van afrondingen overmaakt naar zijn eigen rekening. Of dit echt ooit heeft plaatsgevonden weet ik niet. Het laat in ieder geval zien dat de programmeur een heleboel macht heeft. De programmeur kan die macht gebruiken om er zelf beter van te worden, maar meestal zitten programmeurs niet zo in elkaar. De meeste programmeurs gebruiken hun macht ten behoeve van hun baas of opdrachtgever. Programmeurs zijn vriendelijke en welwillende mensen.

De alles-of-niets regel

Mijn stelling is dat programmeurs met één regel code een bedrijf kunnen maken of breken. Voor de meeste regels code gaat dat niet op, maar één op de zoveel regels code drukt net dát verschil uit tussen winst maken en verliezen. Het verschil tussen een veilig systeem en een systeem dat open staat voor elke kwaadwillende hacker, het verschil tussen een systeem dat twintig jaar goed te beheren is en een systeem dat na twee maanden al uitgekotst wordt door de beheerders, het verschil tussen een niet te integreren systeem en een systeem dat zich als een vis in het water tussen andere systemen mengt, het verschil tussen bug en feature. Dit is de alles-of-niets regel, de make-or-break code.

Die ene regel code is zelden te voorspellen. Wanneer die regel geschreven wordt kijkt niet ineens de hele organisatie over je schouder mee. Die regel schrijf je gewoon, in je eentje (of als XP'er met z'n tweeën) tussen neus en lippen door, net als alle andere regels. Misschien weet je het zelf niet eens wanneer je weer zo'n alles-of-niets regel schrijft. Ook aan testers, hoe minutieus ze ook te werk gaan, ontgaan deze regels meestal. Testers kijken alleen naar de functionele buitenkant van het systeem. De alles-of-niets regels uiten zich vaak pas wanneer het systeem buiten de oorspronkelijke functionele eisen treedt, wanneer er net wat meer van het systeem wordt verlangd. De alles-of-niets regel is een onvoorspelbaar ding en ligt volledig in handen van de programmeur.

Ontwerpers houden zich bezig met een abstractieniveau boven de alles-of-niets regels: "Dit component is verantwoordelijk voor klantgegevens, en het heeft een relatie met het adrescomponent". De ontwerper heeft maar een globaal idee waar de alles-of-niets regels ongeveer moeten komen. Als de ontwerper verantwoordelijk moet zijn voor de alles-of-niets regels, dan is de ontwerper in feite al aan het programmeren, want het is ten slotte niet te voorspellen welke regels alles-of-niets regels zullen zijn. Ontwerpers hebben geen macht, ze moeten met veel smeken en bidden hopen op welwillende en verstandige programmeurs. Om aan de macht te komen, hebben ontwerpers MDA (Model Driven Architecture) bedacht. Ze verwachten dat ze het systeem kunnen programmeren in hun favoriete modelleertaal (UML) en zo de afhankelijkheid van de programmeur kunnen doorbreken.

Managers hebben helemaal geen clou van alles-of-niets regels. Ze proberen de boel te overbluffen door te doen alsof programmeurs alleen maar precies hoeven te doen wat ze gezegd wordt: De programmeur als domme lopendebandmedewerker. Al jaren zien we hoe dat nogal slecht werkt. Managers hebben ook een manier bedacht om de macht van programmeurs te breken: ze laten systemen in een ver land programmeren. Daar hebben de programmeurs geen enkel idee wanneer ze wel of niet een alles-of-niets regel programmeren. Ze snappen namelijk niets van het systeem dat ze bouwen. Dat zie je ook af aan de code die je terug krijgt uit die verre landen. Alle regels code zijn ineens alles-of-niets regels geworden. Een meestal zijn het dan niets-regels.

Programmeurs hebben meer macht dan ze denken. Het wordt tijd dat we ons daarvan bewust zijn.

Daan Kalmeijer is docent consultant bij CIBIT adviseurs | opleiders (e-mail: daan@cibit.nl).