Steeds vaker kom ik in opdrachten waar de ontwikkelaars continuous integration / delivery / deployment gebruiken. Niet iedereen bedoelt hetzelfde met de termen en het is niet altijd duidelijk wat er nu precies gebeurt. Stap één is dan altijd uitzoeken wat de gebruikte term in de huidige context betekent. De ene keer sluit deze beter aan bij de definitie dan de andere keer. Wanneer de term minder goed aansluit zijn er twee opties voor de tweede stap:

1. Ermee werken zoals het daar bekend is.

Dit is de weg van de minste weerstand. Je kunt lekker instappen op de rijdende trein en kijken waar het eindigt. Op een kort traject is dit zeker het overwegen waard. Waarom proberen een werkend geheel te wijzigen? Wanneer er snel iets af moet en het gehanteerde proces geen lange levensduur heeft, zit er niets ergs in deze oplossing.

2. Proberen de juiste terminologie in te voeren

Hierbij moet je een groep mensen aanleren om andere terminologie te gebruiken dan ze gewend zijn. De slagingskans hangt vooral af van hoe flexibel de mensen zijn die de termen hanteren. Als het proces al gedurende langere tijd in gebruik is, is het natuurlijk moeilijker om de terminologie aan te passen dan bij een net ingevoerd proces. Natuurlijk lukt dit alleen maar als je zelf zeker bent van de terminologie en weet dat er verschil zit tussen continuous integration, continuous delivery en continuous deployment.

> **Continuous integration**
> The practice of merging all developers' working copies to a shared mainline several times a day.

> **Continuous delivery**
> A software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

> **Continuous deployment**
> Every change is automatically deployed to production.

De derde stap is om -nadat de terminologie helder is (aansluitend bij standaarden of niet)- het proces goed ingericht te krijgen. Je moet stukje bij beetje het proces zien aan te passen, zodat de trein beter gaat rijden. Stappen in het proces moeten misschien gesplitst worden om beter inzicht te geven, of er moeten stappen toegevoegd worden. Dit resulteert in wijzigingen in het ontwikkelproces. Helaas roepen deze wijzigingen ook vaak weerstand op.

Dan helpt het als je een goede onderbouwing kunt geven over het waarom. In deze continuous werkwijze zal veel geautomatiseerd moeten plaatsvinden.

Denk hier bijvoorbeeld aan het automatisch meten van en sturen op:

- Code statistieken (bijv. aantal methoden in een klasse, aantal regels in een methode)
- Code kwaliteit (bijv. code duplicatie, code complexiteit)
- Dekking van testgevallen (unit, integratie)

Daarnaast zullen ook testgevallen geautomatiseerd uitgevoerd moeten worden. Het blijft dus niet bij snappen wat er kan, maar je moet dit ook zelf kunnen regelen.

De training Introductie CI/CD gaat in op de betekenis van de termen **Continuous Integration, Continuous Delivery en Continuous Deployment**. In de training gaan cursisten zelf aan de slag met Jenkins om zo ook te ervaren wat er gebeurt bij het opleveren van software. Niet alleen de impact op de testrol in een team wordt behandeld, maar ook hoe en waar de tester het ontwikkelproces kan beïnvloeden. De training gaat in op de verschillende keuzes die gemaakt kunnen worden bij CI/CD en wat daaraan ten grondslag ligt. Hierdoor zul je als tester een meerwaarde bieden in het project.

Jeroen Mengerink, testarchitect bij Polteq