Inhoudsopgave

INTRODUCTIE

Waarom dit boek?

De eerste versie van dit boek is geschreven in het Engels, tijdens de COVID-19-crisis in 2020, een pandemie veroorzaakt door het SARS-CoV-2-virus. Het was noodzakelijk om besmetting zo veel mogelijk te voorkomen door het houden van onderlinge afstand en het handhaven van hygiëne. Naast het advies om geen handen meer te schudden werd men verzocht om zoveel mogelijk thuis te werken, hetgeen sterk afhankelijk was van IT, gerealiseerd door software.

Ikzelf was werkzaam bij Bosch als groepsleider van een softwareontwikkelafdeling waarvan plotseling 90% van de ingenieurs vanuit huis ging werken. Computers, beeldschermen en apparatuur werden mee naar huis genomen, en gebaseerd op een betrouwbare netwerkinfrastructuur kon het werk gelukkig doorgang vinden. Enkele van de ontwikkelaars of testers waren nodig op de zaak omdat zij werkten met de grote testsystemen bestaande uit honderden apparaten, maar de meeste ontwikkelaars konden probleemloos vanuit huis werken. Codedatabases, build-systemen en testsystemen werden remote benaderd en bestuurd, terwijl de communicatie en samenwerking plaatsvonden met *Teams* of *Skype*.

Dit alles was gedreven door software. Eens te meer realiseerde ik me gedurende deze periode hoe afhankelijk onze samenleving is van software.

En nog steeds worden de rol en de afhankelijkheid van software groter. Denk er eens over na waar en in welke apparatuur software

een rol speelt in je dagelijkse leven, en je zult tot de conclusie komen dat software de wereld regeert.

Software controleert je smartphone, je computer, je televisie, je auto, je geluidsinstallatie, je oven…. Kun je je een apparaat voorstellen waarin geen software draait?

Daarnaast heeft software een kritische rol binnen onze samenleving. Financiële systemen zijn geïmplementeerd door software, ons openbaar vervoer wordt beheerst door software, maar ook onze luchthavens worden beheerd door software. Laten we eerlijk zijn, software beheerst de wereld. Stel je eens een wereld voor zonder software.

Op 20 augustus 2011 publiceerde The Wall Street Journal een artikel van Marc Andreessen, met de titel *Why software is eating the world*. In dit artikel beschreef Andreessen het toenemende belang van software in onze wereld met onder andere de verstorende invloed op het zakenleven en de vernietiging van complete industrieën. De invloed van software op ons zakenleven en onze industrieën is, zacht uitgedrukt, enorm!

Als gevolg van dit toenemende belang van software realiseren bedrijven zich meer en meer dat digitalisering niet alleen een feit is, maar ook een noodzakelijkheid. Bedrijven dienen te transformeren naar softwaregedreven bedrijven en moeten daardoor meer en meer investeren in software. En daardoor is de rol van softwarekwaliteit evident.

Regelmatig krijg ik de indruk dat softwarekwaliteit synoniem is met softwaretesten, zeker als ik boeken of artikelen over softwarekwaliteit lees. Echter, mijn bescheiden mening is dat

softwarekwaliteit zo veel meer is dan alleen testen. Testen is een noodzaak, omdat we niet in staat zijn foutloze software te ontwikkelen.

Dit boek geeft een holistische kijk op softwarekwaliteit. Behalve dat ik in het boek testen behandel, zal ik aandacht besteden aan de interne kwaliteit van software en zelfs aan wát softwarekwaliteit mogelijk maakt. Dit boek geeft inzicht in wat echt belangrijk is in software-ontwikkeling zonder in details te treden, welke veel beter beschreven zijn in de boeken in de bibliografie.

Wie zou dit boek moeten lezen?

Dit boek zou gelezen moeten worden door managers van softwareteams, zodanig dat ze begrijpen waarom softwareontwikkelaars bepaalde keuzes maken.

Meer specifiek zou het boek gelezen moeten worden door kwaliteitsmanagers, zodat ze weten welke vragen ze moeten stellen om een idee te krijgen over het kwaliteitsniveau van de software.

Dit boek kan ook als hulpmiddel dienen voor softwareontwikkelaars die bepaalde aspecten van software-ontwikkeling willen uitleggen aan hun collega's of management die geen achtergrond in software-ontwikkeling hebben.

Het boek zou gelezen kunnen worden door recruiters die competente softwareontwikkelaars dienen te selecteren, ondanks het gebrek aan kennis over software-ontwikkeling.

Het kan gebruikt worden in de meer algemene technische opleidingen, zoals Automobiel- of Gezondheidsprogramma's, waarin

mensen worden opgeleid tot generalist in meerdere technische disciplines in plaats van tot specialist in een specifieke discipline.

Tot slot kan dit boek gelezen worden door iedereen die geïnteresseerd is in softwarekwaliteit en overtuigd wil worden dat softwarekwaliteit meer is dan alleen testen.

Hoe dit boek ingedeeld is

In de eerste twee hoofdstukken van het boek worden een basisuitleg en een overzicht gegeven van software en software-ontwikkelcycli. In het geval je als lezer bekend bent met software-ontwikkeling kunnen deze hoofdstukken worden overgeslagen. Echter, het kan zo zijn dat bij het lezen van deze hoofdstukken impliciete kennis expliciet wordt gemaakt. De informatie uit deze hoofdstukken is essentieel voor de rest van het boek.

Kwaliteitshandhaving binnen software is niet evident, juist omdat veel kwaliteitsaspecten van software onzichtbaar zijn. De software-industrie valt hierbij terug op het proces en de procesbeheersing, om via processen de gewenste kwaliteit te bereiken. In hoofdstuk 3 wordt dieper ingegaan op de veel gebruikte procesreferentiemodellen CMMI en ASPICE.

In hoofdstuk 4 wordt een model voor softwarekwaliteit geïntroduceerd. Het is een abstract model dat zowel externe kwaliteit, zoals ervaren door de gebruikers van de software, als interne kwaliteit, zoals gerealiseerd door de ontwikkelaars, bevat. Bovendien bevat het model creërende kwaliteit; het vermogen van een organisatie om software te ontwikkelen. Dit model wordt in de

navolgende hoofdstukken gebruikt om de verschillende aspecten van softwarekwaliteit uit te leggen.

De balans tussen het investeren in softwarekwaliteit en het corrigeren als gevolg van een lage softwarekwaliteit en geassocieerde kosten worden in hoofdstuk 5 behandeld. Deze balans is geen zwart-witverhaal; het is een balans, gedreven door de business-overwegingen.

In hoofdstuk 6 wordt het begrip kwaliteitsschuld geïntroduceerd, met als belangrijk onderdeel Technische Schuld.

Hoofdstuk 7 en 8 bevatten praktische suggesties voor het behalen en het onderhouden van kwaliteit. Hoofdstuk 7 behandelt het verzachten van Technische Schuld terwijl hoofdstuk 8 fouten en testen behandelt.

Aan *safety, security* en *wetgeving* voor software is een apart hoofdstuk gewijd, omdat je er niet omheen kan. Een lage kwaliteit in de context van deze aspecten kan enorm grote schade aanrichten. Het is belangrijk genoeg om expliciet aandacht aan te schenken in een apart hoofdstuk.

Hoofdstuk 10 gaat over het meten van verschillende aspecten van softwarekwaliteit; hoe we een idee krijgen van iets wat als zodanig niet gemeten kan worden.

Met al deze informatie in gedachten zou men in staat moeten zijn om te besluiten welke prioriteiten in software-ontwikkeling gesteld dienen te worden, hetgeen behandeld wordt in hoofdstuk 11.

In hoofdstuk 12 worden enkele beruchte softwarefouten belicht. Om een nog beter inzicht te krijgen in mogelijke fouten en mogelijke consequenties van deze fouten.

Na kennis te hebben genomen van al deze informatie over softwarekwaliteit en wat echt belangrijk is in software-ontwikkeling, wordt stof tot nadenken gepresenteerd in hoofdstuk 13.

Ik realiseer me dat het lastig kan zijn om dit boek te lezen en te begrijpen indien je niet bekend bent met software-ontwikkeling. Onbewust, maar zeker ook bewust, gebruik ik vakjargon. Aan het einde van het boek is een Lijst van Concepten toegevoegd, die wellicht kan helpen het gebruikte jargon te begrijpen.

Het gebruik van de Engelse taal in dit boek

In sommige gevallen zullen Engelse termen, benamingen en afkortingen gebruikt worden. Deze zullen niet altijd naar het Nederlands vertaald worden, omdat deze Engelse benamingen nu eenmaal zeer gebruikelijk zijn in de IT. Een voorbeeld zijn de benamingen van de diagrammen die in UML[1] gebruikt worden, of de woorden 'safety' en 'security', waarvan ik vind dat de betekenis minder duidelijk wordt indien deze vertaald worden naar 'veiligheid'.

Een ander voorbeeld betreft de afkorting GUI, de afkorting van Graphical User Interface. Dit dient vertaald te worden naar Grafische Gebruikers Interface. Echter, de afkorting GUI is een zo ingeburgerde term dat deze in het boek gebruikt wordt in plaats van de eventueel

[1] Unified Modeling Language

te gebruiken Nederlandse afkorting GGI. Hetzelfde kan gesteld worden voor de afkorting AI (Artificial Intelligence), die gebruikt wordt voor Kunstmatige Intelligentie, of CPU, die gebruikt wordt voor Centrale Verwerkings Eenheid.

Het boek bevat quotes die oorspronkelijk uit Engelstalige boeken of artikelen komen. De quotes in dit boek zijn vrij vertaald naar het Nederlands.