

Inhoudsopgave

Introductie	i
Waarom dit boek?	i
Wie zou dit boek moeten lezen?	iii
Hoe dit boek ingedeeld is	iv
Het gebruik van de Engelse taal in dit boek.....	vi
Hoofdstuk Een	1
Wat is Software?	1
Programmeertalen.....	1
Enorme Software Programma's.....	5
Scheiding van Zorgen.....	6
Abstractie	8
Van Specificaties naar Software	9
Ontwerp	11
Het gewenste ontwerp	11
Het bedoelde ontwerp.....	11
Het geïmplementeerde ontwerp.....	12
Bron Code	12
Executiepaden.....	13
Configuratie Management.....	17
Versie Beheer van Bron Code	17
Configuratie als Code	19
Software diversiteit.....	21

Bouwen, Installeren en Inzetten.....	21
Bouwen, installeren, inzetten op een embedded apparaat	25
Bouwen, installeren, inzetten op een personal computer	26
Bouwen, installeren, inzetten in De Cloud	26
De Cloud	27
Infrastructuur als Code	28
IaaS, PaaS, SaaS.....	28
Softwaretypes	30
Systeemsoftware	30
Applicatiesoftware.....	31
Open Source Software.....	31
OSS-licenties	32
Copyleft.....	33
Permissive	34
Risico's bij het gebruik van OSS	34
Snippets en libraries	35
OSS-management	37
Software versus Hardware	38
Software is ontastbaar, hardware is tastbaar.....	38
Wetten van de fysica	38
Software-instanties zijn identiek, hardware-instanties wijken af	39
Het produceren, opslaan en distribueren van software is goedkoop	40
Productie van software is een integraal deel van ontwikkeling	41
Software-ontwikkeling “stopt nooit”	41

Samenvatting	42
Hoofdstuk Twee.....	45
Software Ontwikkel Levenscyclus.....	45
Van Waterval naar Incrementele Ontwikkeling.....	45
Spiraal.....	48
Agile.....	50
Scrum	53
Het Scrum-team.....	54
Product Backlog	55
Definition of Ready	57
Sprint Backlog	58
Definition of Done.....	59
Scrum meetings	60
Het Opschalen van Agile	61
Het Opschalen van Scrum.....	61
SAFe	63
Continuous X.....	67
DevOps.....	69
De Stacey Matrix.....	71
Product Levenscyclus Management.....	74
Innovatie	75
Analyse.....	75
Ontwikkeling	75
Inzet	75
Onderhoud.....	76

End of life	77
Samenvatting.....	78
Hoofdstuk Drie	81
Proces Referentie Modellen	81
CMMI.....	82
Maturity levels	82
Practice Areas	83
Het bepalen van het CMMI Maturity Level	86
ASPICE.....	88
Processen	88
Capability levels	92
Proces Beoordeling.....	93
CMMI, ASPICE en Agile	94
Het Nut van Proces Referentie Modellen	96
De kracht en het gevaar van een procesreferentiemodel.....	96
Samenvatting.....	98
Hoofdstuk Vier.....	101
Wat is Software Kwaliteit?.....	101
Het 1+3 Software Kwaliteitsmodel.....	102
Product Kwaliteit	103
Ontwerp Kwaliteit.....	104
Code Kwaliteit	105
Organisatorische Kwaliteit	105
De iconen in het 1+3 SQM	107

De iconen van organisatorische kwaliteit.....	107
De iconen van productkwaliteit.....	109
De iconen van ontwerp kwaliteit.....	110
De iconen van codekwaliteit.....	111
Transcendente Kwaliteit vs. Gemodelleerde Kwaliteit	112
Transcendente Kwaliteit	112
Gemodelleerde Kwaliteit	113
Beide moeten overwogen worden	114
ISO-25010	115
Functionele Geschiktheid	116
Prestatie Efficiëntie.....	116
Compatibiliteit	117
Bruikbaarheid.....	117
Betrouwbaarheid	118
Security	118
Onderhoudbaarheid	119
Overdraagbaarheid	119
Samenvatting	120
Hoofdstuk Vijf.....	121
Totale Kosten van Kwaliteit.....	121
Kosten van het Implementeren van Kwaliteit	122
Product Kwaliteit.....	122
Ontwerp Kwaliteit.....	122
Code Kwaliteit	123
Organisatorische Kwaliteit.....	123

Kosten van de Gevolgen van Lage Kwaliteit.....	124
Product Kwaliteit	124
Ontwerp Kwaliteit.....	125
Code Kwaliteit.....	125
Organisatorische Kwaliteit.....	125
Totale Kosten van Kwaliteit	126
Product Kwaliteit	129
Ontwerp Kwaliteit & Code Kwaliteit.....	129
Organisatorische Kwaliteit.....	130
Verschillende kwaliteitsniveaus?.....	131
Bepaal je gewenste kwaliteitsniveau.....	131
Samenvatting.....	133
Hoofdstuk Zes.....	135
Kwaliteitsschuld	135
Kosten van Verandering	137
Kwaliteitsschuld - Voorbeelden.....	139
Product Kwaliteit	139
Ontwerp Kwaliteit.....	140
Code Kwaliteit.....	143
Organisatorische Kwaliteit.....	145
Technische Schuld.....	146
Technische Schuld Kwadranten	147
Samenvatting.....	149
Hoofdstuk Zeven.....	151

Beperk Technische Schuld.....	151
Bewustwording van Technische Schuld.....	152
Het belang van Clean Design and Clean Code	152
Snelheid van Software-ontwikkeling	155
Technische Schuld en Ontwikkelaar Moreel.....	158
Oorzaken van Technische Schuld.....	159
Aard van de Business	159
Verandering in Context.....	160
Ontwikkelproces	160
Mensen en Team	161
Software Vakmanschap.....	161
Creëer een Clean Design en Clean Code Mentaliteit.....	164
Software Ontwikkel Praktijken.....	165
Specificaties Ontwikkeling	165
Modelleer Visueel	167
Pas Patronen toe.....	169
Architectuur Risico Analyse	172
Communiceren en documenteren van de architectuur	174
SOLID.....	177
Pas Clean Code toe	180
Peer Reviewing	180
Traceerbaarheid.....	183
Refactoring en Restructuring	188
Registreer en prioriteer Technische Schuld	190
Statische Code Analyse	191

Statische Ontwerp Analyse	197
Strategisch Adoptie Model voor het Volgen van Technische Schuld.....	199
Onwetend	199
Niet gevolgd	199
Ad-hoc.....	200
Systematisch	201
Gemeten	201
Geïstitutionaliseerd	202
Geautomatiseerd	202
Zet Technische Schuld in Context.....	202
Samenvatting.....	204
Hoofdstuk Acht.....	205
Testen en Fouten	205
Fouten	205
Heisenbugs.....	208
Het Beheren van Fouten	210
Debuggen	211
Debuggen – scope is van belang.....	212
De debug-activiteiten	214
Testen.....	219
Verschillende Testscenario's.....	220
Controle gedreven testscenario's.....	221
Datagedreven testscenario's	222
Korte Terugkoppelingen	224

Test Piramide.....	225
Agile Test Kwadranten.....	227
Test Driven Development	230
Behavior Driven Development.....	231
Grafische Gebruikers Interface Testen	233
Chaos testing.....	235
Test Automatisering is Niet Gratis.....	238
Kunstmatige Intelligentie in Test Automatisering	239
Testcase-selectie.....	239
Unit-tests	240
Zelfhelende tests.....	240
Samenvatting	241
Hoofdstuk Negen.....	245
Safety, Security, Wetgeving.....	245
Safety.....	247
Safety engineering	247
ISO-26262.....	249
Security.....	250
Kwetsbaarheden	251
Organisaties die het Uitbuiten van Kwetsbaarheden Bestrijden	256
Security Engineering	258
Wetgeving.....	262
Nieuwe technologieën vereisen aangepaste wetgeving	263
Intellectuele Eigendom Rechten	264

Opgelegde wetgeving	265
Wetgeving in relatie to safety en security	267
Samenvatting.....	268
Hoofdstuk Tien.....	271
Het Meten Van Software Kwaliteit	271
Meten Wat niet Gemeten Kan Worden.....	271
Achterblijvende en Leidende Metingen	274
Business Georiënteerde Metingen	275
Het Meten van Product Kwaliteit	275
Klantenterugkoppeling	277
Gerapporteerde fouten uit het veld	278
Gevonden fouten gedurende ontwikkeling.....	278
Testdekking.....	279
Het Meten van Ontwerp Kwaliteit	285
Modellen en ontwerpdocumentatie	286
Gat tussen bedoeld en geïmplementeerd ontwerp	287
Het Meten van Code Kwaliteit	290
Begrijpelijkheid	291
Statische code-analyse	292
Secure code.....	294
Versiebeheersysteem	294
Het Meten van Organisatorische Kwaliteit.....	295
Moreel.....	297
Opleiding.....	297

Programmeervaardigheden	298
Infrastructuur	299
Processen	299
Velocity	302
Inzet	304
Samenvatting	305
Hoofdstuk Elf	309
Het Zetten van de Juiste Prioriteiten	309
Creërend Vermogen zou de Hoogste Prioriteit moeten hebben	310
Cultuur	310
Intrinsieke Motivatie	316
Professionaliteit	319
Het in dienst nemen van de beste ontwikkelaars	320
Heroverweeg Operationele Prioriteiten	322
Verzeker een Goede Betrouwbare Ontwikkelomgeving	322
Beperk Kwaliteitsschuld	325
Verbeter Constant	326
Ontwikkel Functionaliteit	326
Samenvatting	327
Hoofdstuk Twaalf	329
Beruchte softwarefouten	329
De crash van de Mars Climate Orbiter	330
Stroomstoring Boeing 787 Dreamliner	331

Millenniumbug.....	332
A2LL.....	333
Boeing 737 MAX: een safety-gerelateerd ongeluk.....	335
Onbedoelde acceleratie.....	336
Retrospect	339
Hoofdstuk Dertien.....	343
Stof Tot Nadenken.....	343
Trial and Error-Programmeren	343
Ponskaarten	344
F5	345
Het Beste uit twee Werelden	346
Moeten Softwareontwikkelaars Gecertificeerd worden?	348
Lage drempel om softwareontwikkelaar te worden	348
Software-ontwikkeling is meer dan programmeren alleen.....	349
Test Driven Development nader Bekeken	350
Volg je de Rode of de Groene Lijn?.....	355
Snelheid versus Tempo	357
Software Development.....	359
Proces versus Vaardigheden.....	361
Software Schattingen..., Waarom is het zo Moeilijk?.....	364
Cone of uncertainty	365
Puzzelanalogie	366
Hoofdstuk Veertien.....	371
Afsluiting.....	371

Afkortingen.....	375
Lijst van Concepten	379
Bibliografie	387
Dankbetuiging	391
Over de Auteur	393
index.....	395

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 versturende 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 UML¹ 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.