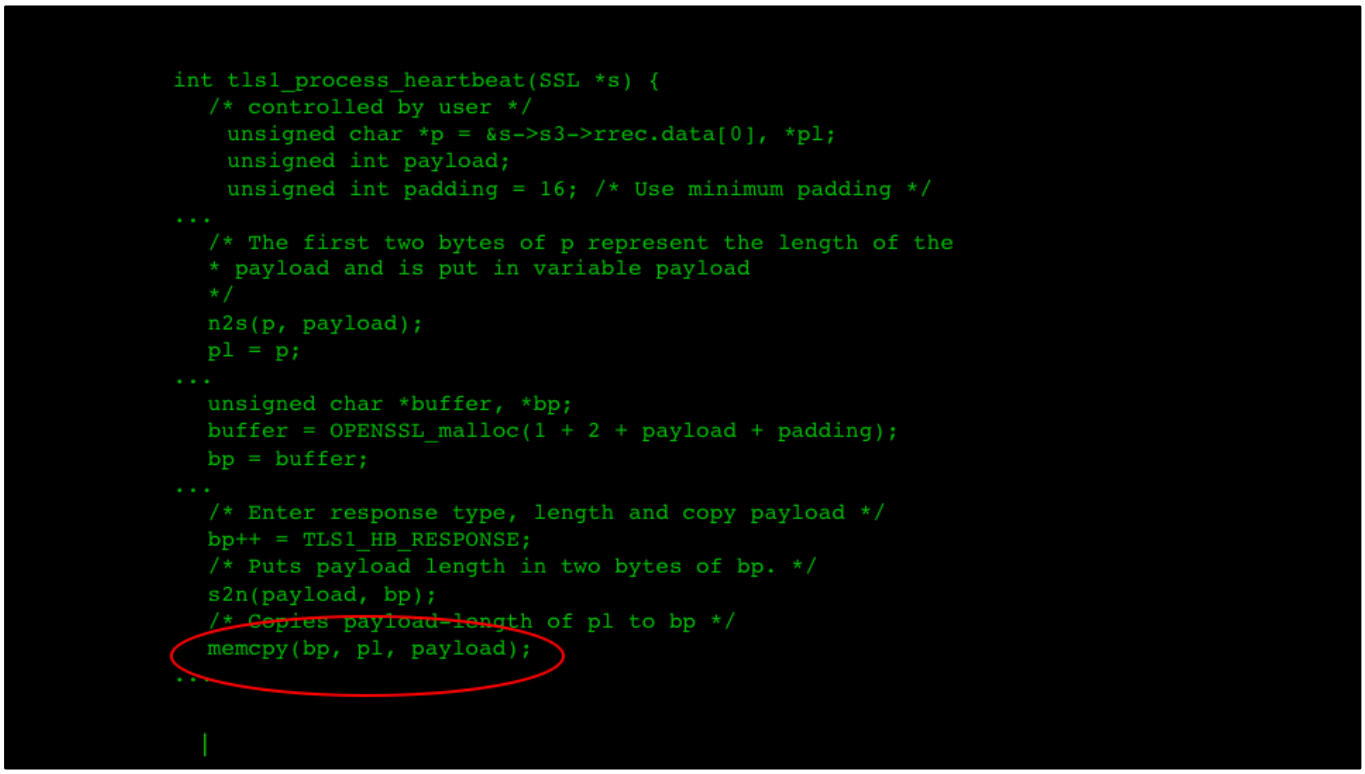
Ik neem u graag mee in een **zoektocht** naar de bron van datalekken, aan de hand van voorbeelden uit de praktijk. Ik **werk namelijk bij SIG** en daar onderzoeken we honderden softwaresystemen per jaar en dan krijg je een aardig beeld van hoe het ervoor staat met security en privacy by design.
Aan **recente voorbeelden** geen gebrek. Zoals bijvoorbeeld vandaag de fout in de broncode van de Stemwijzer, waardoor eenvoudig de stemadviezen tot nu toe konden worden geteld over de politieke partijen. Inmiddels is dat lek gelukkig gedicht.
Dit verhaal vertel ik niet alleen. Ik doe dit met **de ultieme getuige** van de IT: de broncode. De broncode bepaalt namelijk voor een groot deel wat er allemaal gebeurt in onze **digitaliserende** maatschappij met gegevens, dus ook persoonsgegevens. Dus ook in **uw organisatie**. In **75%** van de IT datalekken ligt de oorzaak in fouten in die broncode. En het **neemt toe**. Volgens onderzoek van Akamai neemt het aantal aanvallen op de webapplicatielaag toe met 25% van 2014 op 2016.
Wat is daarvan het **gevolg**: persoonsgegevens liggen op **straat** met bijvoorbeeld identiteitsfraude-problemen. Organisaties lijden **gezichtsverlies**, verliezen **vertrouwen** en verantwoordelijken kunnen hun **positie** kwijtraken. **Boetes** vormen een groot risico en deze worden alleen maar groter met de nieuwe wetgeving. Of: diensten zijn door een hack lange tijd **niet bruikbaar**.
Het doel is om met onze zoektocht te ontdekken hoe we privacy by design voor elkaar kunnen krijgen. Waar hebben we mee te maken? Welnu, bijvoorbeeld hiermee:



```
int tls1_process_heartbeat(SSL *s) {
   /* controlled by user */
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */
...
   /* The first two bytes of p represent the length of the
   * payload and is put in variable payload
   */
   n2s(p, payload);
   pl = p;
...
   unsigned char *buffer, *bp;
   buffer = OPENSSL_malloc(1 + 2 + payload + padding);
   bp = buffer;
...
   /* Enter response type, length and copy payload */
   bp++ = TLS1_HB_RESPONSE;
   /* Puts payload length in two bytes of bp. */
   s2n(payload, bp);
   /* Copies payload-length of pl to bp */
   memcpy(bp, pl, payload);
...
```

Iemand die hem herkent?

Dit is heartlbleed. Weet u het nog. April 2014.

Hiermee lekt het geheugen van een webserver met daarin vertrouwelijke gegevens waaronder bijvoorbeeld wachtwoorden.

Hierdoor was de belangrijkste implementatie van het belangrijkste beveiligingsprotocol op het internet zo lek als een mandje

Duitse professor, **oudejaarsnacht**

Jarenlang daar gezeten. **Niet te detecteren** door tools en niet te vinden met tests.

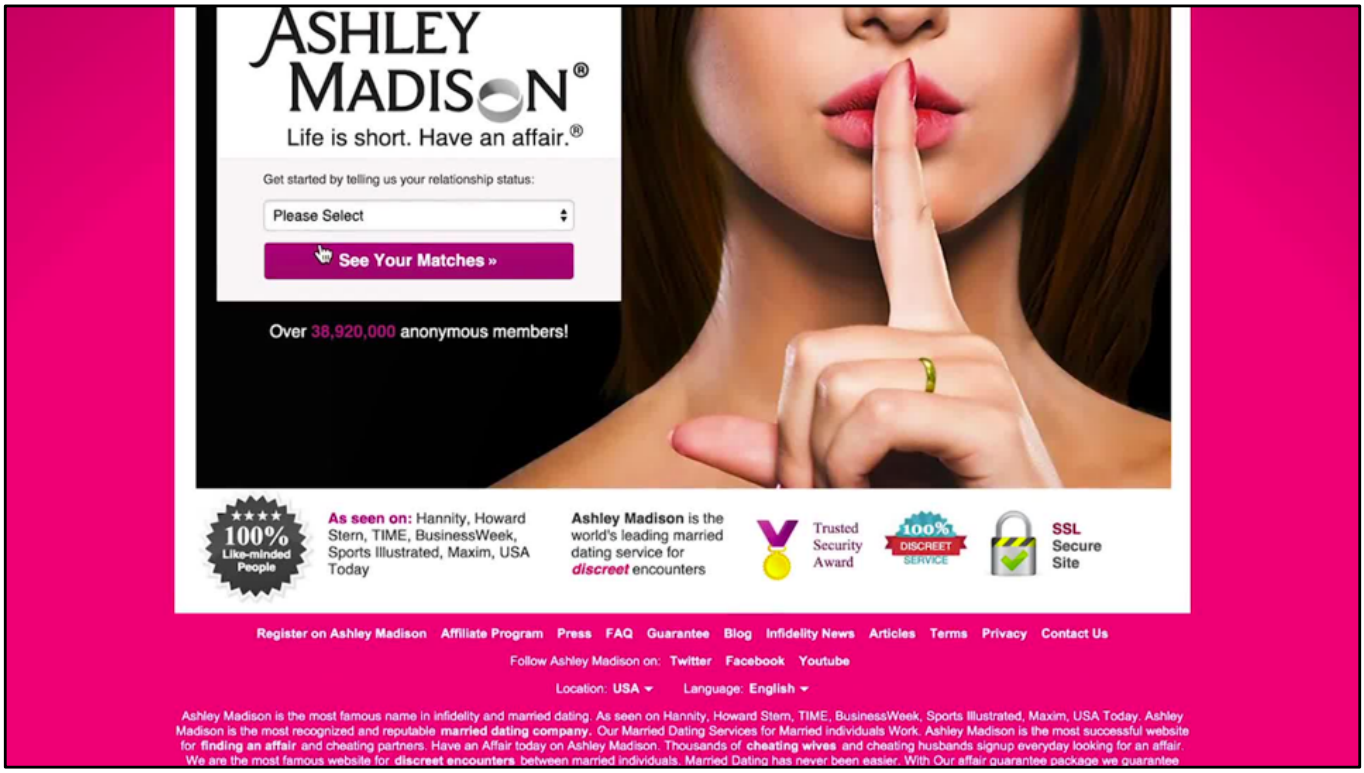
Een maand na bekendmaking was **nog steeds** een ernstig deel van de populaire beveiligde websites kwetsbaar (2%)

Zie hier de **gedaante** van het probleem.

Wat zien wij in de systemen die **SIG onderzoekt**, van set-top box tot verkeerscentrale, van mobiele app tot overheidsregistratie:

Het CPB spreekt in een recent rapport over ‘**Marktfalen**’ van cyber security.

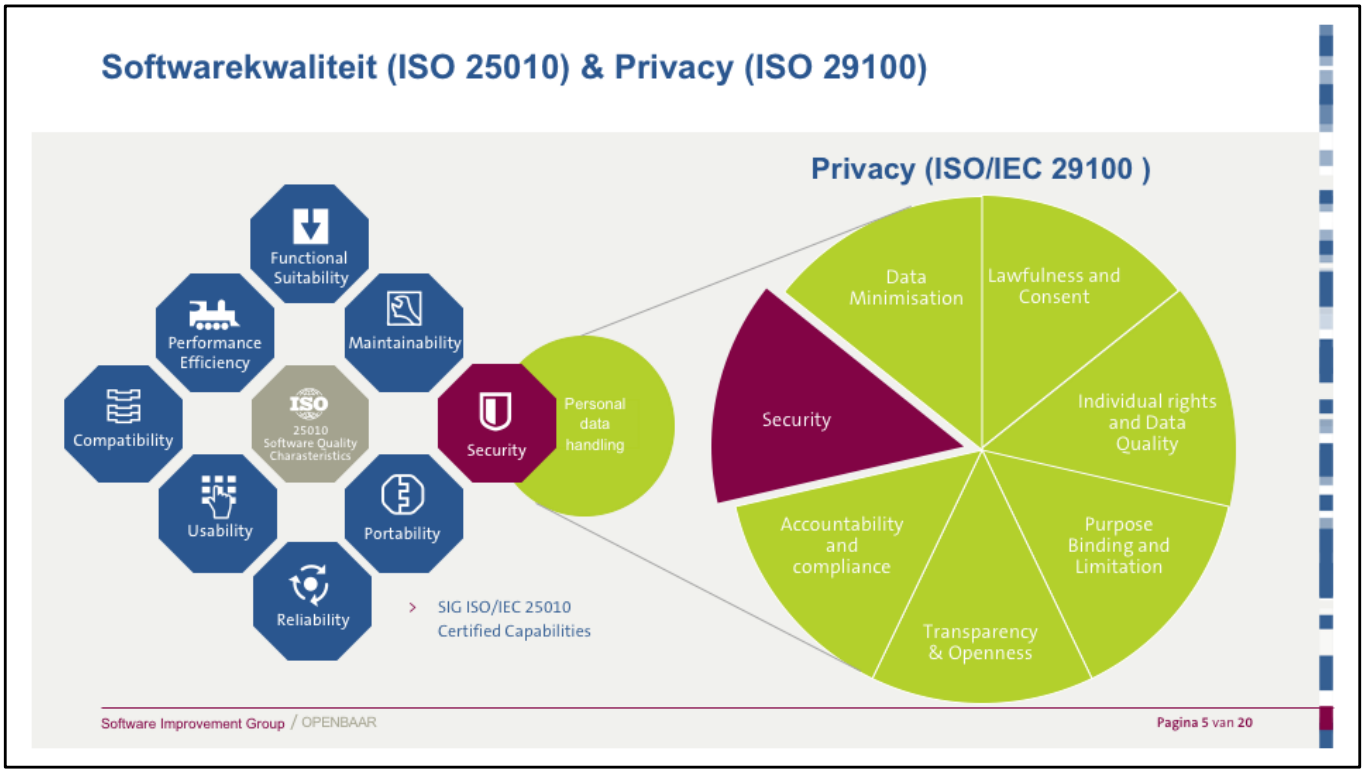
Zoals u ziet dit zijn allemaal securityfouten, die dus een bedreiging zijn voor de privacy, maar er is nog een **andere categorie**
van fouten, die zich het best laat omschrijven met een voorbeeld.



Ashley madison is… was een website om een affaire te regelen voor getrouwde mensen.
Wie was lid? U hoeft het niet te zeggen, we weten het toch allemaal al, want alles ligt op straat
Dating-site Ashley MadisXon (ziet u de keurmerken?) werd gehackt door een te eenvoudig VPN wachtwoord: **Pass1234**, dus security was niet goed, maar daarnaast werd er ook niet goed omgegaan met persoonsgegevens. **Adresgegevens** en **ip-adressen** werden onnodig bewaard. Die werden na de lek allemaal gepubliceerd op het internet. Zo konden **werkgevers** bijvoorbeeld zien wie vanaf werk gebruik had gemaakt van deze site.
Dit is dus een andere categorie van issues. Het is **geen security en het is wel privacy**. **Onnodig veel risico** creëren op het gebied van persoonlijke gegevens of **onwettig** omgang daarmee.

U denkt misschien nu: ja maar **hoe ga ik hier nu grip** op krijgen? Hoe ga ik die spelden in de hooiberg vinden? Nou in elk geval niet door in de documentatie te kijken, want daar staat niet
in dat ip adres wordt opgeslagen of dat betalingsgegevens eigenlijk niet worden weggegooid, of dat autorisaties niet worden gecontroleerd. De waarheid zit in de broncode.

Eerst laat ik u zien hoe wij hier **conceptueel** mee omgaan:



ISO25010 gebruiken als raamwerk voor softwarekwaliteit en security is daar een belangrijk onderdeel in.
ISO29100 is een goed raamwerk voor privacy en daar is security weer een onderdeel van.
Privacy in software is daarmee: security plus personal data handling.
Deze raamwerken gebruiken wij bij SIG om in broncode het niveau van security by design en privacy by design te meten.

En als wij systemen onderzoeken, **wat vinden wij zoal** als het gaat om personal data handling:

Wat opvalt is dat dit niet echt fouten zijn, maar nadelige effecten van het nastreven van een ander doel. Er zijn **goede redenen** voor, zoals software-engineering principes en business intelligence, maar wat we veel zien ontbreken is een goede afweging.

Het gaat te veel ten koste van de privacy, vooral door **onbekendheid** of **desinteresse** hoe je én de voordelen kunt hebben van bijvoorbeeld business intelligence én tegelijk op privacy let. **Win win is vaak mogelijk.**

Dus we hebben nu risico's gezien op het vlak van security en personal data handling. Er is **nog een categorie van risico's in broncode** die een dreiging zijn voor privacy:

We zien ook dit: moeilijk aan te passen software waar het **wachten is op een nieuwe kwetsbaarheid of een datalek**. Daar hoeft dan in principe geen hacker aan te pas te komen: de **developer wordt de dreiging**. **Complexiteit** is misschien wel de grootste bedreiging in informatiebeveiliging en privacybescherming.
De onderhoudbaarheid van software kan dus ook een securityrisico zijn en het komt regelmatig voor dat we **adviseren** een systeemonderdeel op te schonen of zelfs opnieuw te ontwikkelen.

Waarom is het zo matig gesteld met security en privacy by design? Ik onderscheid drie hoofdredenen:

Allereerst: **onmatigheid**. Software is eating the world. We hebben **weinig tijd en weinig aandacht voor security en privacy** omdat software-ontwikkelaars veel te druk zijn met de grote hoeveelheid software van matige kwaliteit.

We maken meer software dan we aankunnen:
12 miljoen software ontwikkelaars
120 miljard regels code in een jaar
15% daarvan moet het jaar daarna worden aangepast
Met 1,8 miljoen nieuwe ontwikkelaars

Gevolg:
Alleen **reactief** onderhoud
Innovatie staat onder druk
Kwaliteit lijdt en dat maakt het probleem alleen maar groter
Systemen lopen **vast** en worden **kwetsbaar** of **lekken** gegevens

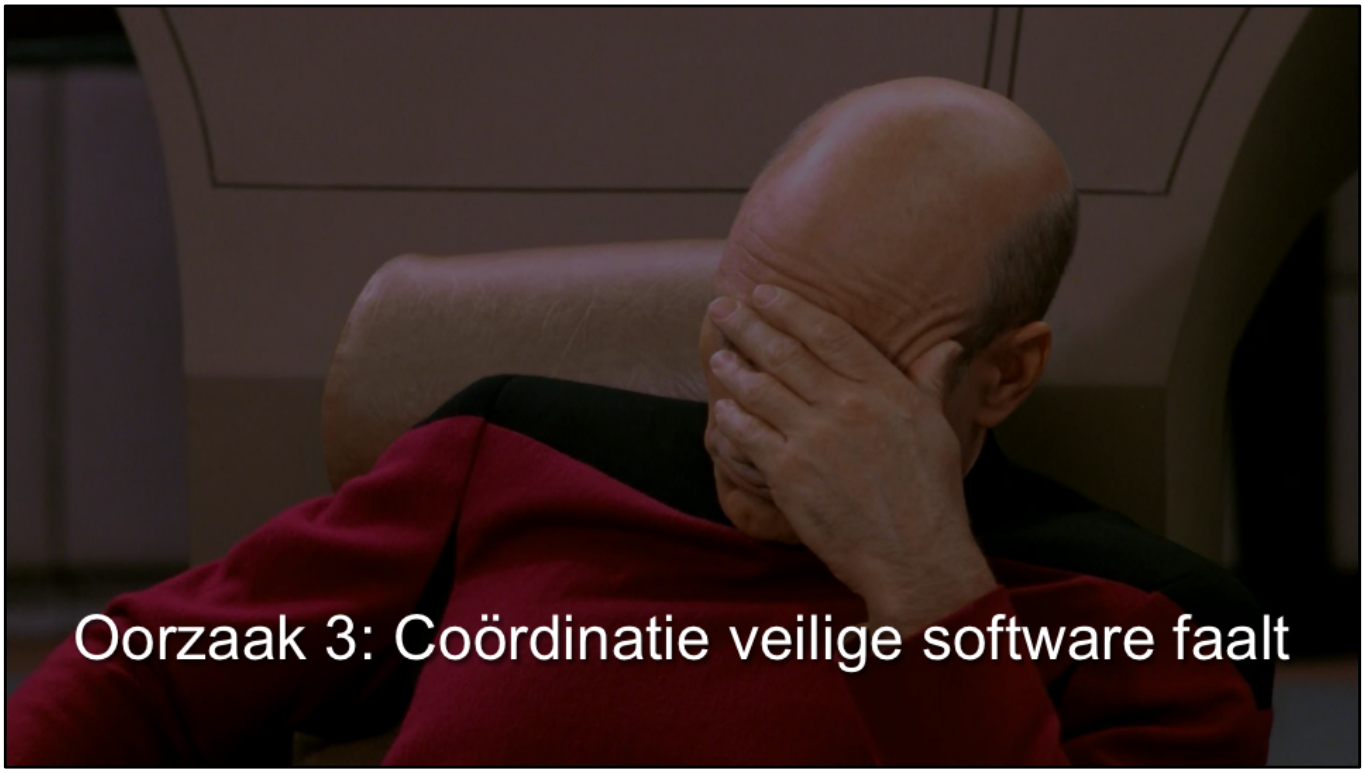
Waar komt dit vandaan. **Build now, worry later**. De vervelende gevolgen, de zogenaamde technische schuld, daarvan ervaar je het grootste deel pas later.

De tweede hoofdreden is **schade-asymmetrie**. Tussen gebruikers en organisatie en tussen organisatie en leverancier.
Tussen gebruikers en organisatie:
Als persoonsgegevens lekken, dan voelen organisaties dat typisch nog onvoldoende, afhankelijk van het type organisatie. Er kan wel sprake zijn van ernstige imagoschade of boetes door nieuwe wetgeving (AVG).

tussen organisatie en leverancier:
Als een organisatie schade lijdt door een softwarefout dan zijn softwareleveranciers maar beperkt aansprakelijk.

De negatieve gevolgen van een security/privacy incident liggen dus typisch minder bij degene die moet zorgen voor veilige software.

Dat helpt niet echt bij het totstandkomen van een **goede afstemming:**



De derde hoofdreden is dat het nog niet zo goed lukt **om privacy by design te coördinere**n: interne en externe aansturing van het ontwikkelproces, klant/leverancier dialoog, toetsing, etc.
Waarom?

1. Opdrachtgevers verwachten kwaliteit, maar ze **weten niet** hoe ze daar specifiek in moeten zijn. "Wij nemen aan dat onze leveranciers veilige software maken". Maar wat vind jij veilig en wat niet? En hoe ga je dat toetsen? Het landschap van standaarden is uiterst gefragmenteerd
2. Leveranciers (intern of extern) **beginnen er typisch niet over**. Het is niet in hun belang. Misschien is het wel heel lastig om aan de eisen te voldoen. Eigenlijk zouden ze moeten zeggen “Sorry, maar ik kan je geen garanties geven over kwaliteit als je niet specifiek bent". Daarmee zijn ze minder interessant voor opdrachtgevers dan bouwers die dat niet zeggen. Dit is een prisoner's dilemma. Zie verder voor een idee om dit vanuit de software-ontwikkelaar toch in gang te zetten.

We zijn aangekomen bij de bron van datalekken

De broncode is niet in control. Men weet niet waar men staat. **Geen risico-inzicht**.

De AVG vereist dat uw processen, dus ook uw code privacy by design zijn. Dat moet u kunnen bewijzen en **volgens de wet moet u daar dus naar gekeken hebben.**

Brenno de Winter interviewde 9 CISO's: die zeggen moeite te hebben om risico's in kaart te brengen en onzeker of voldoende afgedekt. Ze willen allemaal anoniem blijven.

Toetsen: '**you can't control what you don't measure**'.
Er wordt **niet kritisch en niet goed naar de broncode** gekeken. Terwijl, daar gebeurt het.
Vaak wordt pas getest als het klaar is

- Wijzigingen zijn dan 100 keer zo duur
- Geen tijd om goed te fixen
- Je kunt niet alles vaststellen met testen

Volwassenheidsniveau softwarebouwers: Ze worden **ook nauwelijks gedwongen** om te groeien. Eisen ontbreken, controle gebeurt amper.

Bij die samenwerking: kijk goed naar de **afwegingen bij privacy**: BI, forensics, architectuur. Waar ligt de balans.

Digitale bescheidenheid: **weinig software maken en weinig data** verzamelen

Toetsing: niet alleen checklists en doe diagnostiek van blinde vlekken!

Zo toon je aan dat je bezig bent privacy onder controle te krijgen.

Als je als opdrachgever groeit, dan zullen de **bouwers ook groeien**

Maar hoe ga je om met de grote verzameling aan applicaties? Hoe maak je die **inhaalslag**? Wordt nu verwacht dat je die allemaal tot op de laatste regel gaat doorlichten? Niet per sé, maar je **moet wel starten**. Hoe dan?

Stappenplan onder controle krijgen bestaande software

1. Vul privacy-risicofactoren in voor elk maatwerk-softwaresysteem
   - Verwerkt persoonsgegevens
   - Verwerkt bijzondere persoonsgegevens (artikel 16 wbp)
   - Hoge complexiteit van systeem of keten
   - Einde levenscyclus
   - Inschatting ontwikkelteam over privacy by design
   - Incomplete documentatie
   - Risicofactoren op basis van ervaring experts
2. Onderzoek de broncode van de hoogst scorende systemen
3. Breid het risico-model uit op basis van de nieuwe ervaring en vul opnieuw in
4. Herhaal stappen 2 en 3 tot voldoende vertrouwen over de resterende software

Zo toon je aan dat je het op orde aan het krijgen bent of hebt. En zo kan worden voldaan aan de plicht. Niet alleen
de plicht voor de wet, maar ook de plicht naar het individu en de plicht naar onze eigen professionaliteit. Veel succes! Ik dank u wel.


Software
Improvement
Group
Rob van der Veer, MSc
+31 6 20 43 71 87
+31 20 314 09 50
r.vanderveer@sig.eu
www.sig.eu
Amstelplein 1
P.O. Box 94914
1090 GX Amsterdam
The Netherlands