Opstel over Software Engineering

In 1950 werden de eerste programmeertalen (o.a. Fortran, Cobol en ALGOL) ontwikkeld om financiële en wetenschappelijke problemen op automatische wijzen op te lossen. Naarmate de programma's groter werden, was er meer behoefte aan het structureren van deze oplossingen. Het *NATO Science Committee* heeft in 1968 en in 1969 een conferentie geörganiseerd over Software Engineering. Door velen wordt dit als het startpunt van het vak Software Engineering gezien, doordat beide conferenties enkel gefocust waren op dat vakgebied.

In de jaren '70 vindt er een software crisis plaats, die wordt veroorzaakt door de afwezigheid van software engineering, tijdens de ontwikkeling van software. De Amerikaan Fred Brooks schreef een essay, met als titel “The Mythical Man-Month”, over het management achter software ontwikkeling. Een van zijn meest bekende conclusies was:

“Adding manpower to a late software project makes it later.”

Software ontwikkeling vereiste een andere manier van werken en Fred Brooks merkte o.a. de volgende klachten op gedurende de ontwikkeling van het besturingssysteem OS/360:

1. Software ontwikkeling wordt gedaan door amateurs en heeft daarom “onderhoud” nodig, wat in feite wijst op de slechte opbouw van het programma. Dit onderhoud zou niet nodig zijn geweest, als het programma vanaf het begin van de ontwikkeling was voorzien van een duidelijke structuur.
2. Bestaande software is rommelig. Dit weerhoudt anderen om de software te verbeteren of te gebruiken om verder op te bouwen. En als het wel mogelijk is om te verbeteren of te gebruiken, dan zouden de aanpassing te veel tijd en geld kosten.
3. Op massale schaal worden projecten duurder dan het beschikbare budget en kosten deze meer tijd dan gepland. Daarnaast zouden de verwachtingen niet worden waargemaakt, doordat de ontwikkeling meer tijd kost dan gepland en vervolgens staakt.

In “The Mythical Man-Month” wordt ook een voorstel gedaan voor het opstellen van een productief team van programmeurs. Het voorstel komt van Harlan Mills en is afgeleid van het team voor een chirurg. Een chirurg focust zich op de operatie, terwijl zijn collega's zorgen voor de benodigde apparatuur en middelen. Harlan Mills' team bestaat figuurlijk uit:

- **De chirurg**. Dit is het hoofd van het team en heeft vele jaren ervaring. Zijn taak is om het project te leiden en bepaalt in grote lijnen hoe de software wordt ontwikkeld. Hij neemt ook het schrijven van de documentatie voor zijn rekening.
- **De copiloot.** De rechterhand van de chirurg, maar heeft veel minder ervaring dan de chirurg zelf. In feite is de copiloot in opleiding voor chirurg.
- **De administrator.** De chirurg heeft altijd het laatste woord over het personeel, goederen en de ontwikkelruimte, maar de administrator regelt deze benodigdheden.

- **De redacteur.** Hoewel de chirurg een groot deel van de documentatie schrijft, zorgt de redacteur voor het voltooien van zijn werk. Een van zijn taken is het bekritiseren, het verder uitwerken van de documentatie en het aanbrengen van verwijzingen naar andere bronnen.
- **Twee secretaressen.** In veel gevallen is de taak van de redacteur en de administrator te groot om alleen te voltooien. Hierdoor krijgt elk assistentie van een secretares om het werk te verlichten.
- **De smid.** De chirurg kan niet goed werken zonder goede apparatuur en de smid is verantwoordelijk voor de beschikbaarheid en het onderhoud van dit materiaal. Hij is ook de persoon om gespecialiseerd gereedschap te maken.
- **Het proefkonijn.** Het testen van het programma wordt geregeld door de chirurg. Om dit proces te versnellen is een proefkonijn (ook wel 'tester') aangesteld, die de chirurg helpt bij het opstellen en uitvoeren van deze testen.
- **De methodiekspecialist.** Voor sommige operaties is het handig als de chirurg overlegt met een specialist van een bepaalde operatiemethode, voordat de operatie wordt uitgevoerd. Deze specialist is eigenlijk geen onderdeel van het team, maar heeft in de begin van de ontwikkeling een belangrijke adviserende rol.

Dit team functioneert goed, omdat iedereen een eigen taak heeft en zich daar op kan richten. Hierdoor kunnen de taken worden verdeeld over de daarvoor bestemde specialisten. Deze verdeling zorgt ook voor een betere communicatie tussen de teamleden, ze weten immers naar wie ze moeten toestappen voor een bepaald onderwerp. Hierdoor kan de chirurg zich blijven focussen op het project, terwijl de andere leden hun functie kunnen blijven doen.

Uit bovenstaande taakverdeling kan worden afgeleid waarom het niet goed gaat, als het aantal personen wordt verhoogt en het project vertraging heeft opgelopen. De vertraging kan pas worden verholpen, als de aanleiding duidelijk naar voren is gekomen. Bij Mills' taakverdeling is goed in te zien wie waar verantwoordelijk voor is. De chirurg kan dan actie ondernemen om bijvoorbeeld meer c.q. andere mensen aan een taak toe te voegen.

Als de taakverdeling van Mills niet zou worden gebruikt en er zouden nieuwe mensen worden aangenomen, dan verspreidt het probleem zich over meer mensen. De communicatie wordt chaotischer en dat zorgt er voor dat de kern van het probleem niet kan worden gezien en vervolgens worden opgelost.

Fred Brooks is een belangrijk persoon geweest voor de wereld van Software Engineering. Zijn essays hadden grote invloed op de ontwikkeling van Software Engineering en in 1999 ontving hij daarom, naast vele andere prijzen, de Turing Award voor zijn bedrage.