Cor Ligthert

Zijn de .NET-tools de vijfde generatie van developmenttools?

VAN PONSKAART NAAR .NET

Met bovenstaande vraag loop ik al langere tijd rond en dat komt omdat ik .NET zoveel beter vind dan alle andere developmenttools die ik tot nog toe heb gezien. En dat waren er nogal wat. Om deze stelling te verklaren, zal ik de verschillende generaties ontwikkeltools in beeld proberen te brengen in relatie tot de tijd waarin ze werden gebruikt. Ik beschrijf de generaties tools vanuit de ervaring zoals ze op mij zijn overgekomen. Let op, ik heb het over soorten, er zijn er namelijk ontelbare geweest, waarbij de meeste waarschijnlijk nooit door meer dan drie man zijn gebruikt.

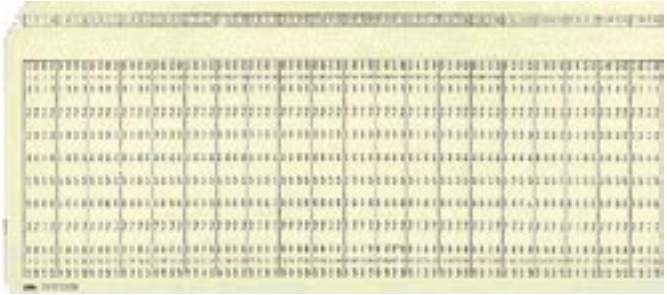
Ik begon met mechanische machines met ponskaarten en papertape. Het programmeren ging met kabels, waarmee relais en/of transistoren en andere zaken werden bestuurd. De machines maakten een cyclus, waarin een ponskaart (zie afbeelding 1) volledig werd verwerkt. Velen zullen de ponskaart niet kennen en toch geloofden de meeste automatiseerders in die tijd dat de ponskaart nooit meer weg te denken zou zijn in de moderne wereld. Bij een ponskaart kun je denken aan een ansichtkaart waarin tachtig kolomen met twaalf rijen waren voorbedrukt. In deze kolomen konden gaatjes worden geponst die via machines met stalen borstels konden worden gelezen. Dit gebeurde door ze met hoge snelheid onder de borstels door te halen (later gebeurde dit optisch tot 100.000 per uur). Vooral in het begin was het niet mogelijk twee gaatjes direct achter elkaar te lezen. Je ziet dit nog steeds terug in het EBCDIC-codestelsel waar veel codes soms een rare sprong maken.

De hardwaretechniek in het begin

De cycli vinden we nog steeds terug in de hedendaagse computer, maar ze zijn vele malen sneller. Er was ook papertape, deze hadden zeven of acht gaatjes naast elkaar. Een leuk aspect van deze papertape was dat je met de machines die deze papertape aanmaakten muziek kon maken doordat het aanmaakprogramma als een trommeltje werkte. Het was dan ook een van de veelbeoefende spelletjes van die tijd. Beeldschermen waren er in het begin nauwelijks en de eerste zaten direct vast aan de centrale computer om deze te besturen. Daarvoor gebeurde dit met een pakje ponskaarten of de typemachine zoals de Friden- of Teletype-schrijfmachine, zie afbeelding 2.

De eerste toepassingen

Wat er geautomatiseerd werd, was buiten wetenschapswerk meestal de debiteuren- en crediteurenadministratie, met soms de facturering en meer van dit soort zaken. Beter gezegd de registratie en het gesorteerd uitprinten hiervan, waarbij tellingen en soms andere berekeningen werden gemaakt. De input bestond meestal uit eindeloze stapels bakken met ponskaarten (de papertapes werden veelal omgezet in ponskaarten om te kunnen sorteren), de output uit stapels papier. Dit laatste is eigenlijk heel lang zo gebleven en in mijn ogen heeft dit (te) lang de automatisering bepaald.



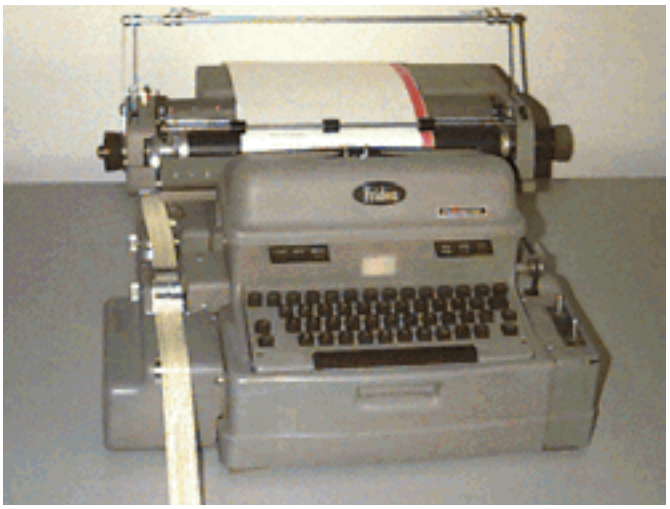
Afbeelding 1. Een ponskaart

De besturingsystemen

Er waren in het begin geen besturingsystemen, de meeste programma's werden met ponskaart of papertape ingelezen en dan begon het proces. Dit was ook een van de redenen dat velen dachten dat de ponskaart nooit zou verdwijnen. Toen er al besturingsystemen waren, kon men zich bijvoorbeeld niet voorstellen hoe je, zonder de bootstrap of de ponskaart, een computer op zou moeten starten als de stroom uitviel. Geen enkele batterij zou genoeg energie kunnen leveren om deze informatie lang genoeg vast te houden in de toenmalig gebruikte fysieke geheugensystemen.

De ontwikkeling van de besturingsystemen

De eerste besturingsystemen deden bijna niets en hadden alles ineen en door elkaar heen zitten. Verandering kwam toen men het besturingssysteem ging onderverdelen in lagen. Dit werd voorna-



Afbeelding 2. De Friden-schrijfmachine

melijk gedaan om de verschillende onderdelen te kunnen scheiden. De kernel rond de processor, een BIOS om de interne hardware te kunnen besturen en IO-gedeelten om de externe hardware te kunnen besturen. Voordeel hiervan was dat men bij vernieuwing van externe hardware niet steeds met een andere besturing te maken kreeg. Er zijn vele namen voor gebruikt. Het begon vooral bij de introductie van de Disk-systemen, die een wat andere benadering vroegen dan de tot dan toe gebruikelijke sequentiële in- en output. Overigens is dit sequentieel denken heel lang mode gebleven en ik heb het idee dat velen dit nog steeds doen. (Vroeger heette het sequentieel, tegenwoordig wordt het meer serieel genoemd.)

De invloed van de microcomputer

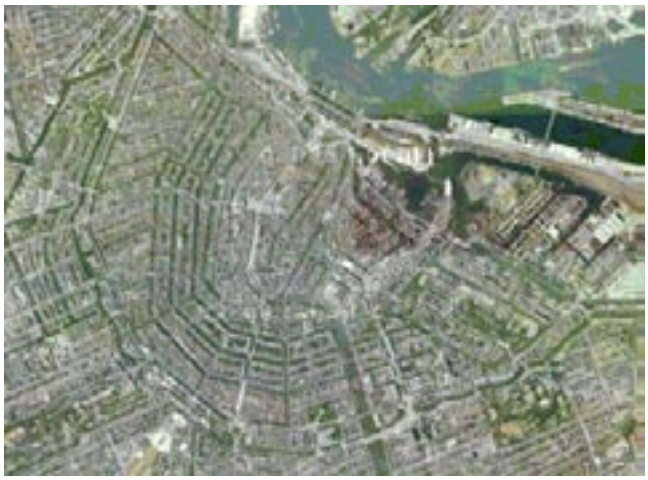
De komst van de microcomputer (een mini was niets meer dan een gestripte mainframe in die tijd) veranderde dit. Ook hier werd eerst wat pionierswerk verricht, maar Bill Gates en consorten waren de eersten die alle troep bij elkaar voegden en dit het besturingsysteem PC-DOS noemden (ook wel MS-DOS als ze het aan anderen dan IBM verkochten). De scheiding tussen gebruikerssoftware en besturingsystemen was compleet onmogelijk gemaakt. Daarom is het misschien niet zo vreemd dat ik het framework nog altijd niets meer dan een extra laag op het besturingsysteem noem.

De start van de ontwikkeltools (de eerste en tweede generatie)

Mijn eerste ervaring met programmeertalen was met RPG en Macro Assembler. De laatste was bijna complete machinetaal met wat macro's om de IO te vereenvoudigen. Bij het debuggen gebruikten we dan ook de sleutels, zoals deze dingen heten (knoppen meestal met zestien posities van 0 tot F), op de computer om machinetaal in te brengen en uit te lezen.

De ontwikkeling van de ontwikkeltools (de derde generatie)

Voor het echte programmeerwerk is de Macro Assembler heel veel gebruikt en niet alleen op IBM-machines. Elke leverancier had zijn eigen versie. De tool was echter omslachtig en stond ver van de materie. Voor veel zaken was het beter om macro's te maken. De veelheid van versies waardoor de uitwisselbaarheid een probleem was, leidde er toe dat men over meer uniforme programmeertalen ging denken. Hierin ontstonden al gauw twee richtingen, de op technische resultaten gerichte talen zoals Algol en Fortran (beide gebaseerd op algebra) en het op zakelijke resultaten gerichte COBOL (een taal die gebaseerd was op het beschrijven van de gebruikte informatie, de processen en de benodigde output). De processen waren in al deze talen niet op de gebruiker gericht, maar nog steeds om de sequentiële in- en uitvoer. Er is geprobeerd deze talen ook voor UI-processing te gebruiken, maar het blijft gebaseerd op serieel, voor echte gebruikerinteractie zijn de talen minder geschikt. Het gebruik van deze symbolische talen werd al snel de derde generatie van het ontwikkelen genoemd.



Afbeelding 3. Het centrum van Amsterdam met de grachtengordel

De analogie van de derde generatie met de ontwikkeling van Amsterdam

Op zich was in mijn ogen de derde generatie de gouden tijd van ontwikkelen. Ik vergelijk het zelf altijd met Amsterdam. Eerst had je de binnenring van de eerste generatie, waarin alles ongeorganiseerd werd gebouwd, dan de tweede generatie, van de Singel tot de Herengracht, het begin van de beginnende handel met heel veel innovatie en dan de derde generatie, de Keizersgrachtbuurt, waar de mensen woonden die net zo rijk waren als keizers, zie afbeelding 3. Hierna kregen we de vierde generatie; voor degenen die Amsterdam een beetje kennen, zo vanaf de Prinsengracht tot de buurten die gebouwd zijn tot ongeveer de Eerste Wereldoorlog. Dat zijn onder andere de Jordaan, de Pijp en de Kinkerbuurt. In deze vierde generatie was snel en goedkoop bouwen het trefwoord, waarbij de kwaliteit en de gebruiker onbelangrijk was. Het moest snel veel geld opleveren. Dat deed het ook voor de bouwers. Het meeste is nu gelukkig gesloopt met uitzondering van de Jordaan, maar kijk dan niet hoeveel geld er door de huidige rijke eigenaren is ingestoken om het kwalitatief bewoonbaar te maken.

De slechte invloed van ontwikkelingen als RPG op de ontwikkeling voor de developer

Het eerder door mij genoemde RPG was iets bijzonders. Het was door IBM-programmeurs ontwikkeld om de vele mechanische apparatuurgebruikers zo snel mogelijk naar computers toe te brengen. Als je het gebruikte, dan beschreef je in ponskaartvorm de input, het rekengedeelte en het outputgedeelte. Daarnaast was er een kort gedeelte om de te gebruiken in- en outputhardware te beschrijven. Eigenlijk hoefde je niet veel meer te weten dan hoe de aan te leveren gegevens er uit zagen, de benodigde berekeningen en hoe de output er moest uitzien. Het maakte programma's op basis van de IBM-apparatuur, die ponskaarten (maar ook voornamelijk sequentieel benaderbare schijffiles) en printers gebruikte. De programmagenerator genereerde een programma waarmee alles was te verwerken. Degenen die hiermee programma's maakten, noemden zich dan ook al snel programmeur.

Het ontstaan van de vierde generatie tools

Het management dat RPG zag, was al snel van mening dat hiermee alles kon. Wat wisten zij van de techniek die er achter zat, wat op zich niet zo vreemd was, omdat zij alleen de output op papier zagen. Dit was de toekomst voor degenen die niet beter wisten. Dit is niet de laatste keer dat een dergelijke misvatting ontstaan is. De managers die op school naast hun echte vak Basic en Pascal hebben geleerd, hebben nog wel meer onzin weten te verkondigen. Zo is mij door personen, die duidelijk niets van programmaontwikkeling afwisten, talloze malen verteld dat het maken van programma's zou verdwijnen. Zij baseerden dit op allerlei slimme tools die ze tegenkwamen en die door ontwikkelaars werden gebouwd om snel resultaat te halen bij simpele problemen.
Ik heb zelf ook dit soort tools gebouwd, ze waren echter vluchtig en gebaseerd op het type machine dat ik gebruikte. Het is bijvoorbeeld eenvoudig om op een scherm te tekenen en er dan een programmabeschrijving uit te laten komen als een scherm maar 1920 posities bevat. Op basis van deze input en output beschrijvende gedachte zijn echter complete ontwikkelsuites gemaakt die al heel snel de vierde generatie tools genoemd. Ze waren/zijn meestal gebaseerd op een type hardware, besturingsysteem en/of database, wat niet zo moeilijk was in een wereld waarin eigenlijk naast de IBM-mainframes voornamelijk Unix en MS-DOS (later Windows) bestond. Zelfs de boekhouder kan hiermee programma's maken. Het resultaat was navenant. Bij elke organisatorische wijziging moet het hulpmiddel vaak weer helemaal opnieuw worden aangepast. Deze tools genereren doorgaans ook geen opnieuw te gebruiken code, maar verwerken vaak nog telkens de gemaakte brongegevens.

Het einde van de vierde generatie door het voortschrijden van de ICT-techniek

Deze tools zijn en waren volledig gericht op registratie en de verwerking hiervan. Omdat de tools meestal op een bepaalde hard-/softwareomgeving gericht zijn, zijn wijzigingen als het gebruik van internet en het gebruik van PDA's naar mijn mening de doodsteek voor deze tools. Er worden wel allemaal kunstgrepen toegepast, maar het past niet meer in de door de boekhouder gecreëerde hulpmiddelen. Je mag het ook niet van hem verwachten. Een boekhouder moet registreren, zelden is hij een goede communicator. Hij weet welke cijfers er ingaan en wat er moet uitkomen. Dat daar ook techniek voor wordt gebruikt, gaat langs hem heen en hoeft hij ook niet te weten. Communicatie in de ICT gaat verder dan het verzenden van een file over internet zoals velen in het begin hebben gedacht. Het omvat alle aspecten van communicatie en wordt ondersteund door hulpmiddelen die in de vorige eeuw nauwelijks bestonden.
Het ontwikkelen is dus weer terug waar het moet zijn: bij de programmaontwikkelaars, de developer, de programmeur of hoe je hem ook wilt noemen, maar in ieder geval een vakman op het gebied van het gebruik van hulpmiddelen op het gebied van communicatie en de registratie hiervan. Deze registratie is niet alleen administratief, maar maakt ook gebruik van andere media zoals beeld en geluid.

De volledigheid van .NET over de aansluiting op de techniek

Microsoft is er in geslaagd deze ommekeer bij de eeuwwisseling in te zien. Ze gaven de tool met .NET echter een onmogelijke naam, die direct alleen maar aan internet deed denken. Het gebruik van de term .NET is nu algemeen en ik gebruik dit dan maar ook. De tools hierin zijn niet alleen voor internet, we kunnen ze gebruiken voor een batchproces, wetenschappelijke toepassingen, toepassingen op PDA's, media in multivormen en dat alles met dezelfde basis aan hulpmiddelen. Deze hulpmiddelen zorgen ervoor dat het hiermee te maken product op een kwalitatief hoog niveau kan staan en onderhoudbaar is voor een vakman op dit gebied, waarbij vooral de uniformiteit een grote pré is.
Programmeren voor internet is niet vreemd meer voor iemand die vroeger alleen maar voor Windows programmeerde. De te gebruiken begrippen en hulpmiddelen zijn hetzelfde. Het zijn niet alleen de ontwikkeltools op zich; bestaande toepassingen als kantoorautomatisering met Office, hulpmiddelen als Sharepoint Portal Server om informatie te verspreiden en BizTalk om informatie samen te brengen zijn ook aan de orde. De uniformiteit maakt dat .NET het steeds eenvoudiger maakt om hulpmiddelen hiervoor te creëren.
Een zijsprongetje. Een in mijn ogen innovatieve vakman op dit gebied is vaak iemand die het leuk vind om ook eens een spelletje te ontwikkelen, net zo als het in het begin gebeurde met de papertape ponser, maar nu natuurlijk veel mooier, met zaken als GDI en DirectX. Met andere woorden, ontwikkelen met .NET is weer leuk.

De brede invulling van .NET-ontwikkeltools

Dit ontwikkelen gebeurt nu in deze nieuwe generatie tools op een wijze, waarin alle voordelen van de voorgaande generaties zijn opgenomen. Ik ben mij bewust dat ik om te chargeren nogal eenzijdig over de vierde generatie software schrijf. Er zijn in die tijd natuurlijk ook veel goede dingen ontstaan. Om er een te noemen, de nadruk op communicatie vooraf tussen alle partijen over het te verkrijgen product en de documentatie hiervan. De goede zaken van de generatie zijn echter behouden in .NET of zelfs verbeterd. Maar ook de eerste scheiding van geesten in de ICT, die er was met Fortran/Algol contra COBOL, is behouden in de vorm van C++/C# en Visual Basic-programmeertalen die de problematiek op verschillende wijze benaderen, maar uiteindelijk het zelfde eindresultaat opleveren.

De vijfde generatie

Sommigen zullen het een nadeel vinden dat ICT niet meer door de boekhouder, de commerciële directeur of elk ander is te doen, De tijd is voorbij dat je zelf even een programmaatje kon maken of het door een neefje als oefening in elkaar kon laten knutselen. De deskundigen moeten weer serieus worden genomen, zodat de organisaties de concurrenten kunnen bijblijven. Als we met de nieuwe generatie resultaten willen bereiken, dan kunnen we er niet omheen om bij deze deskundigen te rade te gaan. Mijn eindconclusie: ik zie .NET als de vijfde generatie software, die hopelijk snel de vierde generatie software in de vergetelheid zal brengen.

Wil je hierover met me discussiëren, dan kan dat. Er is een praktisch ongebruikte nieuwsgroep Microsoft.public.nl.devtools wat volgens mij de ideale plaats is om van gedachten te wisselen. Natuurlijk is deze nieuwsgroep slechts goed bereikbaar met een newsreader. Dit bespaart ons dan meteen een discussie met boekhouders en gelijkgerichten via browsers (overigens niets ten nadele van boekhouders, we kunnen niet zonder ze, maar laat ze in hun eigen vakgebied blijven).

Cor Ligthert (MVP) is waarschijnlijk een van de in Nederland langst actieve personen in de ICT. Dit laatste geldt zowel voor technische als voor organisatorische aspecten. Cor is wars van luchtfietserij. De tijd heeft hem geleerd dat wat vandaag alleen voor de sier wordt toegevoegd, morgen belachelijk is. Samen met zijn 'fellow' MVP Ken Tucker uit Florida onderhoudt Cor een website. www.vb-tips.com, die is gericht op het gebruik van Visual Basic (ADO.NET, ASP.NET). Via email is Cor te bereiken op: cor@computron.nl