Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

4

Aansluiting hbo en wo op instroomopleidingen Info Support

Gert Jan Timmerman - hoofd Kenniscentrum Info Support

SAMENVATTING

Al twintig jaar werft Info Support (een IT-dienstverlener) starters op de arbeidsmarkt onder mensen die op hbo- of wo-niveau zijn afgestudeerd in Informatica. Ter wederzijdse kennismaking bieden we studenten ook vaak een afstudeerplek aan. Aan het begin van het dienstverband worden deze starters, na een strenge selectie, verder opgeleid in ons eigen Kenniscentrum. In de afgelopen twintig jaar hebben we gemerkt dat de kloof tussen wat men geleerd heeft tijdens de opleiding en wat wij aan kennis en vaardigheden van onze instroom verwachten, steeds groter wordt.
Zijn onze verwachtingen te hoog? Is de mismatch te verklaren door inhoudelijke verschillen van inzicht? Wat zou Info Support hieraan kunnen doen en wat kunnen hbo- en wo-opleidingen doen?

TREFWOORDEN

Starters, instroom, opleidingen, kennis, vaardigheden, vakmanschap

INLEIDING

Info Support is een IT-dienstverlener die maatwerksoftware bouwt voor klanten, zowel met .NET- als met Java-technologie. Voor Info Support is vakmanschap een kernwaarde en dit blijkt onder andere uit het feit dat wij uitsluitend medewerkers aannemen (afgezien van stafmedewerkers) die een afgeronde hbo- of universitaire opleiding Informatica hebben. Ook het feit dat we deze medewerkers voordurend opleiden draagt hiertoe bij.
Starters (nieuwe medewerkers die rechtstreeks van een informaticaopleiding komen) krijgen een (standaard) opleidingstraject voordat ze ingezet worden op projecten. Bij deze initiële opleiding bouwen we, uiteraard, voort op de basis die in hun Informaticastudie gelegd is. Steeds vaker lopen we er echter tegenaan dat deze starters niet de kennis hebben waar wij wel op rekenen. Er is sprake van een steeds groter wordende mismatch tussen wat de starters aan kennis en vaardigheden hebben en wat wij daarvan verwachten. Deze mismatch is niet gekoppeld aan een bepaald opleidingsinstituut, maar we komen die tegen bij alle opleidingen Informatica in Nederland.
Er zijn verschillende verklaringen mogelijk voor een dergelijke situatie: zo kan het zijn dat wij te hoge eisen stellen aan de starters. Het kan ook zijn dat er inhoudelijke verschillen zijn in de manier waarop wij tegen zaken aankijken en hoe daarover gedacht wordt op hogescholen en universiteiten. De bedoeling van de sessie is om daar meer helderheid over te krijgen.

De hiervoor beschreven mismatch tussen de kennis van starters en de verwachtingen die wij daarvan hebben, spitst zich toe op een aantal onderwerpen.

SYSTEEMONTWIKKELING

Allereerst het systeemontwikkelingstraject. Alle systeemontwikkelingsmethodieken zijn uiteindelijk afgeleid van het klassieke Watervalmodel. Dat wil zeggen dat alle systeemontwikkelingsmethodieken op een of andere manier een volgorde aanhouden in het verzamelen en specificeren van requirements, het maken van een ontwerp (functioneel en technisch), realisatie en testen en ten slotte oplevering. In het Watervalmodel worden deze fasen sequentieel doorlopen en wordt in principe pas met een volgende fase gestart als de vorige afgerond is. In de meer Agile-methodieken wordt daarentegen vaak iteratief gewerkt, waarbij steeds stukjes van de functionaliteit ontworpen en gebouwd worden. Omdat het klassieke Watervalmodel de basis is voor al deze verschillende methodieken, is het van bijzonder groot belang dat studenten het Watervalmodel heel goed leren kennen. In onze ervaring hebben starters daarentegen een zeer verwrongen beeld van het Watervalmodel. Zo hebben zij vaak de indruk dat het Watervalmodel alleen geschikt is als 'alles van tevoren bekend is' en dat het onmogelijk is naar een vorige fase terug te keren. Ook denken ze dat het onmogelijk is in de loop van het traject wijzigingen te accepteren. Hierdoor hebben ze vaak de indruk dat het Watervalmodel nooit in de praktijk gewerkt kan hebben of uitsluitend geschikt is voor zeer kleine projecten. Wij zouden veel liever zien dat studenten eerst het Watervalmodel echt goed leren kennen en begrijpen, voordat ze in aanraking komen met meer Agile-methodieken, omdat het lastig is, zonder goede kennis van het klassieke Watervalmodel, andere methodieken goed te begrijpen.
Een andere misvatting van starters gaat over Requirements: veel starters beginnen bij het opstellen van requirements bij de input, terwijl requirements meestal vooral over de output gaan. Uit de output kan vervolgens de benodigde input afgeleid worden. Organisaties laten systemen ontwikkelen vanwege de output: dat is (in het algemeen) het doel van een informatiesysteem. Dat er ook nog gegevens ingevoerd moeten worden, kost alleen maar tijd en geld. Dat is een vorm van investering die men moet doen om ooit gegevens uit het systeem te kunnen halen. Het is dus van belang om, gegeven de output, de input zo beperkt mogelijk te houden. Dat kan alleen als bij het opstellen van requirements vooral naar de output gekeken wordt en de benodigde input hiervan afgeleid wordt.

FUNCTIONEEL EN TECHNISCH ONTWERP

Ook over de onderwerpen Functioneel en Technisch Ontwerp komen we veel misverstanden tegen bij starters: vaak denken zij dat het Technisch Ontwerp een detaillering is van het Functioneel Ontwerp en dat het Functioneel Ontwerp bijgevolg vooral globaal is. Ook zijn ze vaak van mening dat de Requirements het Functioneel Ontwerp vormen.
De werkelijkheid is echter dat het Functioneel en Technisch Ontwerp twee kanten van dezelfde medaille vormen. Beide zijn op hetzelfde detailniveau uitgewerkt. De belangrijkste reden om onderscheid te maken tussen Functioneel en Technisch Ontwerp is dat functionele en technische beslissingen door verschillende mensen (in verschillende rollen) genomen worden. Om te voorkomen dat functionele beslissingen impliciet genomen worden door een persoon die de rol heeft van Technisch Ontwerper en die dus geen goed beeld heeft van de gewenste functionaliteit, is het van belang om functionele beslissingen op te nemen in een Functioneel Ontwerp en Technische beslissingen in een Technisch Ontwerp.

In de praktijk wordt tegenwoordig veel gebruik gemaakt van UML. Ook Info Support gebruikt deze taal om ontwerpen weer te geven. Veel starters denken dat het Functioneel Ontwerp af is als ze een Use Case Model gemaakt hebben. Het Technisch Ontwerp bevat dan een Class Diagram en een aantal Sequence Diagrammen. In het Class Diagram zijn de verschillende classes weergegeven met de cardinaliteiten van de relaties (1-op-1, 1-op-veel, veel-op-veel). Dat zijn functionele beslissingen die op deze manier in een Technisch Ontwerp terechtkomen en niet gedekt worden door een Functioneel Ontwerp dat deze relaties beschrijft. Ook de attributen (properties) van een class zijn functionele beslissingen (welke

attributen horen bij welke class, welke informatie wil ik modelleren?). Ook deze beslissingen komen op die manier terecht in het Technisch Ontwerp. Vaak worden er in het Class Diagram technische datatypen gebruikt bij de verschillende attributen (typen zoals int, double, datetime, string etc.), maar deze datatypen moeten natuurlijk gebaseerd zijn op de bijbehorende functionele domeinen. Als er in het Functioneel Ontwerp echter alleen een Use Case Model is opgenomen, zijn deze functionele domeinen nergens beschreven en is dus niet na te gaan of de technische datatypen de functionele domeinen voldoende afdekken. Naar onze mening zou een Functioneel Ontwerp niet alleen uit een Use Case Model moeten bestaan, maar ook uit een Analysis Model, waarin gedetailleerde Class Diagrams en Sequence Diagrams staan. Alleen op die manier is duidelijk wie (in welke rol) de functionele keuzes gemaakt heeft en wie (in welke rol) de technische keuzes gemaakt heeft. Ook is op die manier altijd vast te stellen of de technische beslissingen de functionaliteit voldoende afdekken en waar dat niet het geval is, kan het verschil inzichtelijk gemaakt worden, zodat hierover een akkoord verkregen kan worden.

SYSTEEMDOCUMENTATIE

Een ander misverstand bij starters is dat men denkt dat Functioneel en Technisch Ontwerp een onderdeel van de systeemdocumentatie is. Het grote verschil tussen Functioneel en Technisch Ontwerp enerzijds en systeemdocumentatie anderzijds, is dat ontwerpen van tevoren gemaakt worden (met als doel het expliciet vastleggen van genomen beslissingen) terwijl documentatie achteraf gemaakt wordt met als doel het systeem inzichtelijk te maken voor toekomstige beheerders of gebruikers. Documentatie wordt altijd gemaakt met een concreet doel voor ogen (bijv. installatiedocumentatie) en een bepaalde doelgroep (bijv. gebruikers of beheerders). De documentatie wordt op maat gemaakt voor het gestelde doel en de betreffende doelgroep. Onderdelen uit het Functioneel Ontwerp en Technisch Ontwerp kunnen hierbij gebruikt worden, maar dat is een kwestie van hergebruik. Het doel van het maken van ontwerpen enerzijds en het maken van documentatie anderzijds is totaal verschillend. Het probleem van dit misverstand (dat veel starters denken dat ontwerpen documentatie zijn) is dat men vaak denkt dat ontwerpen ook heel goed achteraf gemaakt kunnen worden (dus eerst code schrijven en dan ontwerpen maken). Voor documentatie is dit prima, maar voor ontwerpen is dit principieel onmogelijk omdat na het schrijven van de code alle beslissingen (zowel functioneel als technisch) al genomen zijn en het ontwerp dus in feite al gemaakt is. Probleem hierbij is dat alle beslissingen impliciet genomen zijn zonder dat degene die de beslissing nam zich bewust was van de beslissing en zonder dat hij zich heeft afgevraagd of hij wel de rol had waarin hij die beslissing kon nemen. Ook worden impliciete beslissingen vaak genomen zonder dat men zich voldoende bewust is van de consequenties van de gemaakt keuze.

TRANSACTIES

Starters realiseren zich vaak niet hoe essentieel transacties zijn bij alle gegevensverwerkende systemen. Vaak hebben zij de indruk dat Transacties alleen van belang zijn bij het overboeken van geld door banken. In de praktijk is het zo dat alle acties waarbij meer dan één tabel betrokken is (en in een genormaliseerd model is dat al snel het geval) transacties nodig hebben. Het belang van correct werkende transacties voor de integriteit van informatiesystemen is bij veel starters niet voldoende bekend.

Transacties hebben vier kenmerken: Atomicity, Consistency, Isolation en Durability, ook vaak afgekort als ACID-properties. Van deze vier kenmerken, kennen veel starters alleen Atomicity. Vooral Isolation (concreet gemaakt in een aantal IsolationLevels) is bij veel starters onbekend. Het gevolg hiervan is dat men omtrent IsolationLevels geen keuzes maakt in het Technisch Ontwerp en bij de realisatie dus kiest voor de (toevallige) defaultwaarde. Dit leidt tot een impliciete keuze met gevolgen die soms verstrekkend zijn. Wij zijn van mening dat Transacties, inclusief IsolationLevels, tot de essentiële leerstof van Informaticastudenten behoren.

CONCLUSIE

Ervaring leert dat veel starters op de arbeidsmarkt die een Informaticaopleiding afgerond hebben, essentiële kennis en vaardigheden missen. Daarnaast hebben ze vaak een verkeerd beeld van het toepassen van systeemontwikkelingsmethodieken en ontwerpmethoden. Hierdoor ontstaat er een mismatch tussen de kennis en vaardigheden van starters enerzijds en de verwachtingen die bedrijven, zoals Info Support, hiervan hebben.

VERANTWOORDING

Gert Jan Timmerman is hoofd van het Kenniscentrum van Info Support. In deze functie is hij verantwoordelijk voor de vakinhoudelijke selectie van nieuwe medewerkers en voor hun (initiële) opleiding. In de afgelopen 13 jaar heeft hij in deze functie honderden starters aangenomen en opgeleid. Daarnaast is hij lid van Beroepenveldcommissies van verschillende hogescholen en ook als gecommitteerde is hij al meer dan 15 jaar betrokken bij het afstuderen. De inhoud van deze lezing is een weerslag van de ervaringen die hij in de afgelopen jaren heeft opgedaan.
Uiteraard is hierbij sprake van een zekere generalisatie: niet alle starters zijn gelijk en ook de kennis van de ene starter is niet gelijk aan die van een ander. Wat hier beschreven is, geeft weer wat er in de volle breedte van het Informaticaonderwijs in Nederland zichtbaar is en hoe daar vanuit het perspectief van een bedrijf dat traditioneel veel starters aanneemt, tegenaan gekeken wordt.

↑

Een van de vele keynote-sessies die druk werden bezocht