Softwareproductie en autonomie

Een exploratief onderzoek naar de relatie tussen softwareproductie en autonomie van de gebruiker.

Auteur:	Willem Bongers
Studentnummer:	270828
Datum:	15-04-2010
Begeleider:	Dr. H.D. Pruijt
Tweede begeleider:	Drs. L.F.J. Jetten
Faculteit:	Sociologie
Instelling:	Erasmus Universiteit Rotterdam

Inhoudsopgave

1 Inleiding

Arbeid is van oudsher een belangrijk onderwerp binnen de sociologie. Vergeleken met andere wetenschappen die zich bezighouden met dit onderwerp heeft de sociologie relatief veel aandacht voor de werknemer. Vaak wordt de vorm van organisatie van arbeid geanalyseerd naar de effecten op de werknemer.

Dit onderwerp is zowel voor de wetenschap als voor de maatschappij van belang. Door begrip van de interactie tussen organisatie en werknemer kan niet alleen het bedrijf beter en efficiënter werken maar kan ook de positie van de werknemer verbeterd worden.

Het onderzoek op dit gebied zal echter nooit afgerond zijn. Een van de factoren die verantwoordelijk is voor constante veranderingen op de werkvloer is technologie. Niet alleen worden er steeds nieuwe producten geleverd, maar dezelfde producten worden steeds op andere wijzen geproduceerd. Dit betekent dat steeds weer de vraag gesteld moet worden wat dit betekent voor de organisatie van arbeid en de positie van de werknemer.

Ook de IT kan gezien worden als een dergelijke nieuwe technologie. Software is niet alleen een product dat door steeds meer bedrijven geleverd wordt, maar is ook een middel dat gebruikt wordt om producten te leveren. Software bepaald dus mede hoe arbeid er uit ziet, het is dus een nieuwe vorm van arbeidsorganisatie met wellicht nieuwe kenmerken wat betreft de effecten op de werknemers. Deze nieuwe vraag kan echter alleen beantwoord worden als we deze bekijken met de juiste visie.

Een belangrijk deel van de sociologie is het gebruik van theorieën over de vorm van de werkelijkheid. Door deze theorieën te toetsen aan deze werkelijkheid kunnen uiteindelijk uitspraken hierover gedaan worden. Hieruit volgt dat een nieuwe werkelijkheid wellicht

ook een nieuwe theorie vereist om onderzoek hiervan mogelijk te maken.

Dit geldt ook voor de informatietechnologie als middel van productie. Bij dit nieuwe gebied moet de vraag gesteld worden of een andere visie het mogelijk maakt om deze nieuwe vorm van arbeid beter te begrijpen. Deze scriptie behandelt een deel van deze zoektocht.

Als we software zien als een middel van productie dan zal dit middel veel invloed hebben op de arbeid en de werknemer. Ondanks dat er veel sociologisch onderzoek is naar de inzet van software als productiemiddel is er relatief weinig bekend over de productie van dit middel zelf. Hier worden onherroepelijk keuzes gemaakt die uiteindelijk van invloed zijn op de werknemer. Zowel maatschappelijk als wetenschappelijk is het daarom van belang om ook dit gebied te onderzoeken. In dit onderzoek wordt daarom een theorie geformuleerd welke meer inzicht kan geven in de relatie tussen de productie van software en de werknemer. Vervolgens wordt deze theorie getoetst aan de werkelijkheid. Niet om uitspraken te doen over de werkelijkheid maar om uitspraken te doen over de mogelijkheden die deze theorie biedt om de werkelijkheid te begrijpen.

2 Onderzoeksvraag

Het doel van dit hoofdstuk is het zodanig versmallen van het onderwerp uit de inleiding dat er een werkbare theorie geformuleerd kan worden. De inleiding noemt het begrip computerprogramma's enerzijds en de effecten op de arbeider anderzijds. Wat betreft dit tweede begrip zal vanaf dit punt gesproken worden over autonomie. Dit hoofdstuk zal dan ook beginnen met een uitdieping van dit begrip.

2.1 Autonomie

Een manier om naar de positie van werknemers te kijken is kwaliteit van arbeid. Dit begrip kent een rijke historie binnen de sociologie. Het gaat te ver om hier uitvoerig op in te gaan, maar het is van groot belang een beeld te vormen van wat precies onderzocht moet worden. Vandaar dat we hier kort ingaan op dit begrip. Kwaliteit van arbeid kent vele aspecten zoals lawaai en veiligheid. Deze factoren raken de werknemer op een bepaalde wijze en het veranderen van deze aspecten beïnvloedt de kwaliteit van het werk. Minder geluid zorgt bijvoorbeeld voor een rustigere werkplek en minder gezondheidsklachten. Omdat het hier gaat om computerprogramma's zullen we echter minder op dergelijke aspecten ingaan. Wel zullen onze blik werpen op een kant die meer te maken heeft met de mens als psychologisch individu.

Kwaliteit van arbeid kwam voor het eerst ter sprake als reactie op het Taylorisme, met een kleine tussenstap kwam de human relations benadering met de term zelfontplooiing als een belangrijk aspect van de menselijke behoeften. Ook op de arbeidsplek zou het bieden van deze mogelijkheid de werknemer gelukkiger maken en wellicht zelfs productiever (van Ruysseveldt et al., 1998: 54,58).

Vanaf deze benadering is het een kleine stap naar de moderne sociotechniek, aanhangers van de sociotechniek stellen dat een

goede kwaliteit van arbeid bestaat uit het beschikken over beslissingsruimte of autonomie binnen het eigen arbeidsproces. Macht moet zo veel mogelijk naar de onderkant van de organisatie, de werknemer moet als het ware een opdracht krijgen en mag vervolgens zelf beslissen hoe hij deze (met een team) uitvoert (van Ruysseveldt et al., 1998: 58).

Het construct autonomie biedt een goed startpunt voor een onderzoek. Als deelaspect van kwaliteit van arbeid past het goed in de maatschappelijke en wetenschappelijke discussies rond de positie van werknemers terwijl het voldoende afbakening biedt om een zinnig onderzoek op te zetten. Een bijkomend voordeel is (zoals later zal blijken) dat dit begrip goed te vertalen is naar eigenschappen van computerprogramma's. Vandaar dat we dit begrip als onderzoeksobject kiezen.

2.2 Software

Het begrip software verwijst naar alles wat op de computer is opgeslagen en niet gezien kan worden als pure informatie. Software is een manier om volgens bepaalde regels met informatie om te gaan. We zijn echter geïnteresseerd in een klein deel hiervan, het deel dat een daadwerkelijke rol speelt bij het proces wat we arbeid noemen. Het is om die reden dat het onderzoek zich enkel zal richten op software waarmee werknemers van organisaties hun werk uitvoeren. In de rest van dit onderzoek zal de term software dus gebruikt worden om te verwijzen naar computerprogramma's die gebruikt worden door werknemers om hun werk uit te voeren.

Er is al veel onderzoek gedaan naar de impact van software op de autonomie, er is echter een gebrek aan onderzoek dat autonomie relateert aan de ontwikkeling van software. Als we echter het autonomie van arbeid in alle volledigheid willen begrijpen, moeten we ook zien te achterhalen hoe de software die er invloed op uitoefent tot stand komt.

2.3 De onderzoeksvraag

Nu uiteengezet is om welke begrippen dit onderzoek draait kan de onderzoeksvraag als volgt geformuleerd worden:

“Welke inzichten ontstaan uit het gebruik van de structuratietheorie bij het onderzoeken van de ontworpen autonomie en het productieproces in softwareontwikkeling?”

Kort samengevat wordt in dit onderzoek in wezen onderzocht hoe het maken van technologie, bedoeld of onbedoeld, bepaalde gevolgen kan hebben voor de werknemers. Omdat deze vraag echter dusdanig abstract is dat waarschijnlijk alleen de filosofie hier een antwoord op kan geven, is deze versmald naar het zoeken van een theorie die in staat is de gevolgen van softwareontwikkeling op autonomie van gebruikers te verklaren. In deze onderzoeksvraag worden de structuratietheorie, ontworpen autonomie en het productieproces genoemd. Deze begrippen stammen uit de voor dit onderzoek gekozen analytische visie. Later zal duidelijk worden wat ze precies betekenen, op dit punt is het genoeg om te vermelden dat het constructen zijn om de relatie tussen softwareproductie en autonomie beter te kunnen bevatten.
In het volgende hoofdstuk wordt verkend in hoeverre al onderzoek en theorieën bestaan omtrent dit onderwerp. Al snel zal blijken dat deze verkenning, inclusief de formulering van de gebruikte theorie, in een breder kader is geplaatst. Voor deze verbreding zijn twee redenen. Ten eerste plaatst de brede visie het onderzoek in de maatschappelijke en wetenschappelijke discussie rond autonomie en kwaliteit van arbeid. Dit vergroot de waarde van dit onderzoek als bijdrage aan een grotere discussie. Ten tweede is specifiek onderzoek op dit gebied nog niet aanwezig maar is soortgelijk

onderzoek wel reeds verricht. Deze onderzoeken geven mogelijkheden aan om een theorie te formuleren.

3 Conceptueel kader

We kunnen software zien als een product. De productie van software begint zodra een klant met bepaalde wensen met een softwareontwikkelaar in contact treedt. De vorm is echter op dat moment nog een vaag concept. Dit concept verandert met elk contact, elke communicatie en elke programmeerstap van vorm. Elke actor die betrokken is bij het proces van ontwikkeling is betrokken bij deze constante verandering. Het doel van dit onderzoek is het doorgronden van dit proces wat betreft de factor autonomie. Op welke punten in de ontwikkeling van software krijgt autonomie welke vorm en welke processen liggen hieraan ten grondslag?

Om dit doel te bereiken zal bekeken worden wat er al over dit onderwerp bedacht en onderzocht is. Voor het vinden van een conceptueel kader kunnen een aantal bronnen aangeboord worden. Maxwell wijst op de volgende vier: ervaringskennis, bestaande theorieën en onderzoek, verkennend onderzoek en gedachte-experimenten (Maxwell 1996: 27). Het subdoel van dit hoofdstuk is dan ook om al deze bronnen zo goed mogelijk aan te spreken om een conceptueel kader te vormen.

Voor het beantwoorden van de onderzoeksvraag moeten we meer helderheid krijgen in de wisselwerking tussen een tweetal concepten. Ten eerste moet duidelijk worden hoe software en autonomie in het werk op elkaar van invloed zijn. Wat zijn de kenmerken van een softwaresysteem dat werknemers veel autonomie geeft? Ten tweede moet gezocht worden naar een theorie met goede mogelijkheden om de ontwikkeling van software in kaart te brengen. Deze theorie moet ook de mogelijkheid bezitten om de uiteindelijke kenmerken die van belang zijn voor autonomie, mee te nemen in de analyse.

3.1 Automatisering en autonomie

Het startpunt van deze analyse is het werk van Zuboff. Het is verleidelijk om verder terug te gaan in de tijd naar werken die aandacht hebben voor het invoeren van machines tijdens de industriële revolutie, maar wat Zuboff bijzonder maakt, is het feit dat zij als eerste het unieke karakter van computers onderkende. In tegenstelling tot een automatiserende functie kunnen computers ook aangewend worden als een informatiserend middel. Zuboff beschrijft dan ook hoe het werk bij een aantal verschillende Amerikaanse bedrijven sterk veranderd is door de invoering van computers. Werknemers zijn veel meer bezig met informatie dan met operationele taken (Zuboff, 1988).

Het doel van de analyse is een beeld geven van de gevolgen van de informatisering voor de werknemers. Haar conclusie is vooral gebaseerd is op veranderende machtsrelaties binnen een organisatie. Zij ziet kennis en leren als de nieuwe zaken die bepalen welke positie een werknemer binnen een bedrijf vertegenwoordigt. De regel dat een persoon die veel fysiek werk verricht minder macht heeft binnen de organisatie gaat niet meer op. Een persoon met weinig mogelijkheden om informatie in te zien en te manipuleren heeft weinig macht in de nieuwe organisatie. De mate waarin een werknemer toegang heeft tot informatie bepaalt haar hiërarchische positie binnen de organisatie (Zuboff, 1988).

Het bovenstaande is helaas geen eenduidige relatie tussen software design en autonomie maar er zijn een aantal zaken die Zuboff noemt die wel in dit kader passen. Ten eerste wijst Zuboff op taakverbreding als een belangrijk kenmerk van een goed ingevoerd informatiesysteem. In het verlengde hiervan ligt het verdwijnen van de klassieke machtsbalans. Doordat iedereen in bepaalde mate toegang heeft tot de informatie binnen het bedrijf krijgt elke werknemer meer macht. Hier wordt echter gewezen op een lastig

vraagstuk, zowel werknemers als managers zijn niet per definitie blij met een dergelijke ontwikkeling. Managers verliezen macht door de toenemende informatievaardigheden van de werknemers (dit was vroeger exclusief hun domein). Aan de andere kant is het ook maar de vraag of de werknemers tevreden zijn met het verdwijnen van de klassieke hiërarchische relatie. Hun taak wordt er een van betrokkenheid bij de organisatie die niet meer zo duidelijk omlijnd is als het traditionele werk. (Zuboff, 1988).
Ten tweede ziet Zuboff in dat programmeurs van belang zijn in het ontwikkelen van software. Zij wijst er op dat deze groep deels kan bepalen hoe toegang tot informatie verdeeld is en hoe informatie gepresenteerd wordt voor verschillende gebruikers (Zuboff, 1988).

Een werk wat qua methode betreft lijkt op het werk van Zuboff is het boek “The Electronic Sweatshop” van Barbara Garsons. Het boek kan het best gezien worden als een verslag van een groot aantal interviews met als doel het ontdekken van de gevolgen van automatisering door middel van computers. Het beeld dat in haar werk naar voren komt is ronduit negatief. In haar visie zijn mensen creatieve personen die het fijn vinden om zich te ontplooien in hun werk. Wat ze echter ziet is een maatschappij waarin de computer steeds vaker wordt ingezet om macht van werknemers af te nemen. Een gebrek aan vertrouwen in mensen in het algemeen doet managers computers zo inzetten dat alle beslissingsmogelijkheden van werknemers wordt ingedamd. Haar ideologische visies daargelaten zijn er in het boek inderdaad voorbeelden van dit fenomeen te vinden (Garsons, 1984).
De waarde van dit onderzoek ligt in het concept autonomie. Werknemers krijgen volgens Garsons niet de kans zich te ontplooien door bepaalde keuzes in de opzet en implementatie van het computersysteem. Dit is in de visie van Garsons een daadwerkelijke keuze van het management.

Wat we kunnen leren van deze twee studies is dat de inzet van informatietechnologie van invloed kan zijn op de werknemer. Er wordt geen directe relatie gelegd tussen autonomie en informatietechnologie maar er is een verandering van arbeid aanwezig. Met de taak van arbeiders veranderen ook hun verantwoordelijkheden. Belangrijk is hier vooral de maakbaarheid van de organisatie. Beide auteurs stellen dat technologie zo ingezet kan worden dat de positie van de werknemer verandert, de technologie is dus een determinerende factor voor werkervaring.

In plaats van het onderzoeken van software met kenmerken van een controlemechanisme onderzoekt Pruijt een aantal initiatieven die juist gericht zijn op decentralisatie van verantwoordelijkheden. Dit kan gezien worden als een vergroting van de autonomie van werknemers aan de onderkant van de organisatie. De conclusie van dit onderzoek luidt dat de software niet de werkelijke structuur van arbeid verandert, de taken van werknemers die de software gebruiken veranderen niet wezenlijk. De software mag dan een bepaalde ideologie bevatten, maar als dit niet aansluit op de ideologie van de gebruikersorganisatie dan zal deze weinig effectief zijn (Pruijt, 1996: 68-69).

Batenburg, Benders en Schepers werpen een nieuw licht op het bovenstaande door naar organisaties te kijken die juist een organisatieverandering doormaken. Zij hebben onderzocht hoe de invoering van ERP systemen[1] samengaat met het invoeren van een algehele organisatieverandering.
Hun conclusie is dat de nieuwe informatiesystemen in potentie alle mogelijkheden bieden voor het succesvol vormen van een dergelijk

[1] ERP staat voor Electronic Resource Planning, een dergelijk systeem beheert de stroom fysieke objecten door een organisatie.

systeem naar de beeltenis van de gewenste organisatiestructuur. In principe kan een werkgever dus kiezen voor een structuur (bijvoorbeeld de sociotechniek) die ruimte biedt voor een zekere autonomie. De realiteit is echter gecompliceerder.
De ERP systemen die geleverd worden zijn in de basis hetzelfde, er wordt dus niet voor elke klant een nieuw systeem ontworpen. Dit systeem kan echter aangepast worden aan de wensen van de werknemersorganisatie maar dit kost tijd en geld. Vaak wordt door gebrek aan beide gekozen om het informatiesysteem niet wezenlijk te veranderen. De gebruikersorganisatie bevindt zich toch al in een reorganisatie en dus wordt niet het eigen reorganisatieplan uitgevoerd maar wordt dit plan aangepast aan de basisvorm van het softwarepakket (Batenburg, Benders en Schepers, 2002).

Uit deze resultaten blijkt enige controverse. Zuboff en Garsons stellen vast dat een informatiesysteem inderdaad van invloed kan zijn op een organisatie en de autonomie van werknemers. Pruijt stelt daarentegen vast dat het vergroten van de autonomie door aanpassingen van het softwaresysteem op niets uitlopen omdat de werkelijke werkwijzen niet mee zullen veranderen.
De studie van Batenburg et al. werpt weliswaar geen licht op de autonomiekwestie maar toont wel aan dat een software producerende organisatie wellicht meer macht heeft dan men aanvankelijk zou denken. Dit betekent dat goed bekeken moet worden hoe de interactie tussen de klant en producent verloopt.

De tot nu toe genoemde onderzoeken en resultaten geven weliswaar stof tot nadenken maar we hebben nog geen antwoord gekregen op de vraag hoe software van invloed is op de autonomie van werknemers. Er is mogelijk een relatie maar deze is niet zo simpel dat de kenmerken van software de autonomie van werknemers bepalen. Het probleem en wellicht ook de oplossing ligt

in de confrontatie van de software met de organisatie. De software wordt kennelijk anders gebruikt dan voorspeld kan worden puur op basis van kenmerken van de software. We moeten dus op zoek naar een theorie die het mogelijk maakt om de wisselwerking tussen organisatie, werknemers en software te onderzoeken.
Wellicht kan een dergelijke theorie de onduidelijke relatie tussen software en autonomie verklaren. Een theorie die mogelijk deze functie kan vervullen is de structuratietheorie van Giddens. Deze theorie zal eerst in het algemeen besproken worden. Vervolgens zal bekeken worden hoe de theorie toegepast kan worden op autonomie.

3.2 Structuratietheorie

Uit de rest van dit hoofdstuk zal blijken dat de structuratietheorie op een bepaalde manier wordt gebruikt om de mogelijke relatie tussen softwareproductie en autonomie te verklaren. Om recht te doen aan de complexiteit van de werkelijkheid en van de theorie kan deze toepassing niet simpelweg in een paar zinnen omschreven worden. Om het lezen iets makkelijker te maken wordt de kern van de theorie hier echter kort omschreven.
We gaan ervan uit dat software nooit exact gebruikt wordt zoals deze bedoeld is. We nemen echter wel aan dat software meer gebruikt zal worden zoals bedoeld als deze beter is aangepast op de organisatie waarin deze gebruikt wordt. Nu is de tweede aanname dat software een bepaalde mate van autonomie oplegt aan de gebruiker.
Stel nu dat een werknemer kan kiezen hoe hij of zij gebruik maakt van bepaalde software. In een computerprogramma zitten meestal al bepaalde werkwijzen maar dit betekent niet dat werknemer deze hoeft te gebruiken. Door een computerprogramma op verschillende manieren te gebruiken ontstaan er verschillen in de autonomie van

gebruikers. Gebruikers die zich aan de regels van het computerprogramma houden zullen de autonomie hebben die de regels van de software aan ze oplegt. De autonomie van gebruikers die zich niet volgens het de eisen van de software gedragen zal minder lijken op de autonomie die in de software besloten ligt.
Een gebruiker die de software exact zo gebruikt als bedoeld zal dan ook de autonomie hebben die de software aan deze persoon geeft. Een gebruiker die de software niet zo gebruikt als bedoeld zal minder gehinderd worden door de werkwijzen die in de software besloten liggen. Hierdoor zal ook de autonomie van deze gebruiker minder beïnvloed worden door de software. De werkelijke autonomie van een persoon zal dus afhangen van de software zelf maar ook van de manier waarop hij of zij deze gebruikt.
Het volgende voorbeeld illustreert het bovenstaande. Stel een organisatie maakt een computerprogramma dat werknemers verplicht de naam en het adres van elke klant volledig in te vullen. Dit betekent dat iemand die de software goed gebruikt, minder autonomie in het werk zal hebben, men kan niet kiezen de informatie half te vullen. Stel nu dat een bedrijf de software aanschaft maar dat na gebruik blijkt dat de werknemers het programma alleen gebruiken als er genoeg tijd is om alles in te vullen. Dit betekent dat de werknemers minder autonomie verliezen dan men zou verwachten door alleen naar de software te kijken. Hoe goed de software bij de organisatie past speelt ook een rol. Kort samengevat is er een oorzaak en gevolg relatie tussen software en autonomie. Deze relatie wordt echter verstoord door hoe goed de software bij de organisatie die deze gaat gebruiken past. Dit is de basis van de theorie die in de volgende paragrafen uitvoeriger wordt besproken.

De basis van de structuratietheorie ligt in een poging om het Weberiaanse subjectieve denken te combineren met het objectieve

denken in de stijl van Durkheim. Structuur bestaat door gedragingen van individuen, door deze gedragingen wordt structuur ondersteund en veranderd. Anderzijds oefent deze structuur haar invloed uit op elke gedraging. Elke actie wordt beïnvloed door de reeds bestaande structuur. Deze tweezijdige interactie met structuur vindt plaats op drie wijzen: interpretatieve schema's, middelen en normen. Interpretatieve schema's zorgen enerzijds voor betekenisvolle activiteiten en beïnvloeden anderzijds de structuur van betekenis. Middelen geven individuen macht om bepaalde acties te ondernemen en hebben invloed op de machtsstructuur. Ten slotte zijn normen van invloed op de sanctionering van individuen en beïnvloeden ze de legitimiteitsstructuur (Orlikowski en Robey 1991). Deze drie wijzen van interactie worden later nog uitvoeriger besproken in het kader van softwareontwikkeling.

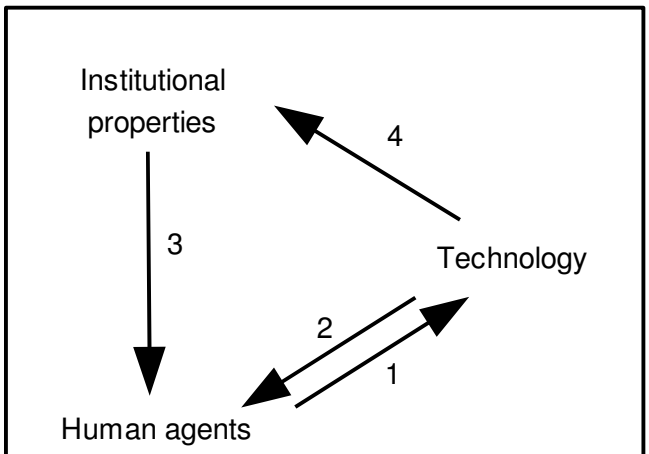
3.2.1 De structuratietheorie en technologie

De structuratietheorie is zeer breed en kan nagenoeg overal worden ingezet waar sprake is van sociale interactie. We zijn echter op zoek naar een theorie die kan verklaren hoe software van invloed is op de autonomie. Een eerste stap in deze richting wordt gezet door Orlikowski en Robey.

Orlikowski en Robey doen een poging de structuratietheorie van Giddens te combineren met (informatie)technologie. Hun aanname is dat technologie in de basis een sociaal product is en dat dit construct via de structuratietheorie begrepen kan worden. Het is een combinatie van de drie wijzen van interactie tussen structuur en individuen (1991).

Deze interactie vindt zowel plaats in het ontwikkelen van de software als in het gebruik hiervan. Door de interactie verandert het product constant, zelfs na de oplevering. In het model van

Orlikowski zijn er vier interacties te onderscheiden zoals weergegeven in figuur 3.1. Ten eerste beïnvloeden individuen de vorm van het systeem, niet alleen door het maken van dit systeem maar ook door het gebruik hiervan (pijl 1). Anderzijds beïnvloedt de software de individuen die het gebruiken. Dit gebruik kan de gebruiker zowel beperken en helpen in haar acties (pijl 2). Ten derde werken individuen met informatie technologie binnen een bepaalde context. Ook deze context is van invloed op hun gedragingen (pijl 3). Ten slotte wordt de institutionele context beïnvloed door het gebruik van informatie technologie (pijl 4). Elk gebruik van computers kan de institutionele context veranderen maar met name ook bevestigen (Orlikowski en Robey 1991).



Figuur 3.1 (Bron: Orlikowski en Robey: 1991)

Het bovenstaande model kan enigszins verhelderd worden door het onderzoek van Brooks als voorbeeld te gebruiken. Deze onderzoeker heeft de introductie van zogenaamde CAD[2] software geanalyseerd met behulp van de structuratietheorie. Uit dit onderzoek blijkt dat in geen enkel geval de gebruikers direct het systeem gebruiken voor alle werkzaamheden waarvoor dit mogelijk is. Hun werk wordt echter wel beïnvloed door het systeem (pijl 2 in

[2] CAD staat voor Computer Aided Design. Een systeem om via software alle stappen van een ontwerproces te faciliteren.

figuur 3.1). Ze kunnen bijvoorbeeld sneller verschillende oplossingen vergelijken. Het systeem wordt echter niet gezien als de enige werkwijze. Het wordt eerder gezien als een extra hulpmiddel bij het werk. De werknemers bepalen dus wat de technologie is en oefen er via die weg invloed op uit (pijl 1 in figuur 3.1). Het feit dat het systeem op deze manier wordt gebruikt betekende volgens Brooks dat het niet geheel geïnstitutionaliseerd is. Het is binnen de institutionele context van het bedrijf nog niet geaccepteerd als de algemene werkwijze (pijl 4 in figuur 3.1). Dit betekent niet dat de context van het bedrijf geen invloed heeft op de acties van de werknemers. Door het plaatsen van computers op bepaalde posities worden gebruikers geforceerd (door de nabijheid van de manager) het systeem te gebruiken (pijl 3 in figuur 3.1) (Brooks, 1997).

Het bovenstaande voorbeeld is maar een korte schets van een gevonden situatie maar de resultaten laten duidelijk zien dat de methode potentieel heeft. Door technologie te zien als een interveniërende factor in de interactie tussen context en acties kan goed begrepen worden wat de invloed van de technologie is. Het feit dat in dit geval technologie meer gezien wordt als hulpmiddel dan als werkwijze kan veel verklaren over het gebruik van de software.

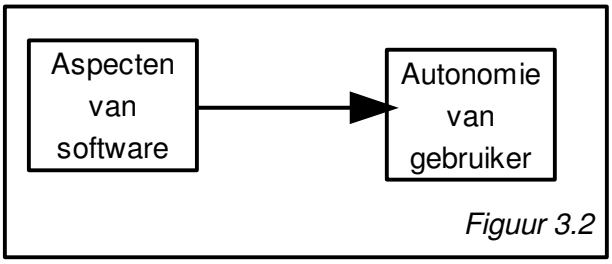
In deze zin is informatietechnologie niet enkel een vastliggend object maar is het een medium voor interactie tussen individuen en de institutionele context. Dit medium kan de individuen helpen bij het uitvoeren van hun taken, maar kan dit ook beperken. Orlikowski en Robey (1991) wijzen in dit verband op de duidelijke keuze die het individu hierin maakt. De manier waarop het informatiesysteem in elkaar steekt is niet determinerend voor de uiteindelijke interactie met dit systeem en de gevolgen voor de institutionele context. Wel zal het systeem altijd van bepaalde invloed zijn. Het zal altijd een rol

spelen in de interactie tussen individuen en de institutionele context.

3.2.2 De structuratietheorie en autonomie

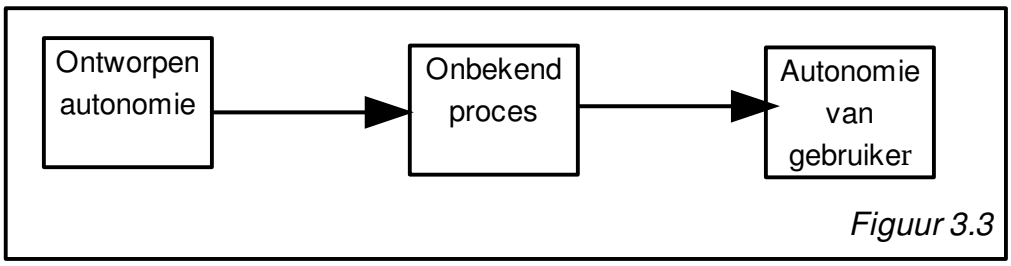
In hoofdstuk twee werd al gewezen op het feit dat het werkelijke onderwerp van dit onderzoek niet de relatie is tussen software en autonomie maar eerder de ontwikkeling van software en de gevolgen voor autonomie. De volgende paragrafen staan in een net wat breder kader dan softwareontwikkeling op zich. De aandacht gaat uit naar de rol van softwareontwikkeling in relatie tot autonomie om zo een maatschappelijk en wetenschappelijk waardevolle theorie op te bouwen.

Alvorens de structuratietheorie toe te passen op autonomie beginnen we met een basisaanname. In de vorige paragrafen werd al gewezen op het feit dat technologie kan beperken of helpen in het uitvoeren van acties. Dit is niet zeer verschillend van het begrip autonomie, vandaar dat we het volgende kunnen stellen: software bezit bepaalde aspecten die de autonomie van gebruikers kan beïnvloeden (figuur 3.2).



Figuur 3.2

Het onderzoek dat reeds is besproken laat zien dat er twijfel is of software van invloed kan zijn op autonomie (Pruijt, 1996: 68-69). Dit kan betekenen dat deze relatie niet bestaat of dat het onderzoek niet juist is uitgevoerd. Het is echter aannemelijker dat de realiteit complexer is dan de bovenstaande figuur en dat een betere theorie nodig is om de waarheid te kunnen verklaren. Voor het gemak breiden we figuur 3.2 uit met een extra variabele.



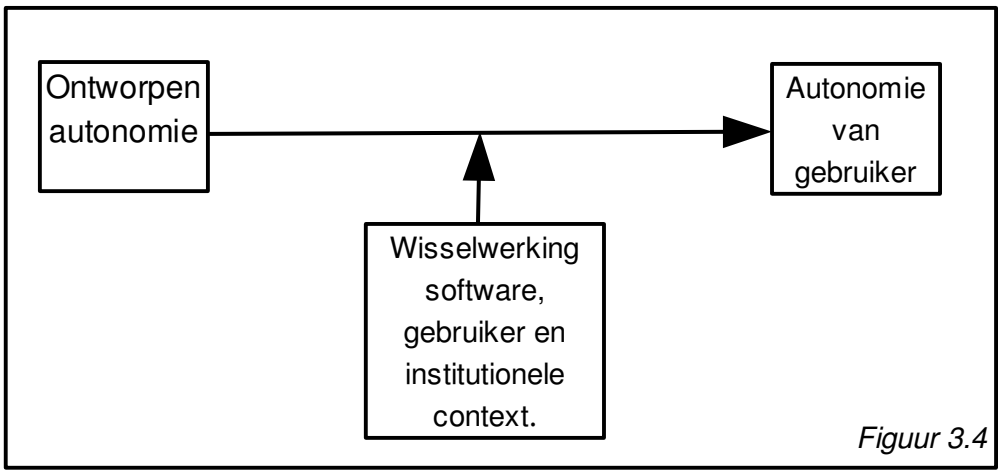
Figuur 3.3

Naast het toevoegen van een onbekende variabele in figuur 3.3 is de term “aspecten van software” veranderd in “*ontworpen autonomie*”. Deze term duidt op een construct dat duidt op de aspecten van software die de autonomie beïnvloeden. In het vervolg van dit onderzoek wordt deze term gebruikt om te verwijzen naar deze aspecten. De term *autonomie van gebruikers* wordt gebruikt om te verwijzen naar de werkelijke autonomie van de gebruikers van de software. Het is nu de uitdaging om het onbekende proces uit figuur 3.3 te ontdekken. Aan deze zoektocht zijn de volgende paragrafen gewijd.

In de bovenstaande bespreking van de structuratietheorie bleek reeds dat technologie een organisatie niet direct verandert maar dat er een wisselwerking is tussen technologie, werknemers en institutionele context.
Het begrip wisselwerking duidt op het feit dat software niet alleen de werknemers en de institutionele context zal beïnvloeden maar dat de institutionele context (indirect) en de werknemers op hun beurt de software beïnvloeden. Deze opmerking lijkt niet mogelijk, software is net als een machine of gebouw een onveranderbaar object. In de traditie van de structuratietheorie worden dergelijke zaken echter alleen gezien vanuit de manier waarop ze invloed hebben op de sociale interactie. In dit geval gaat het dus niet om wat het ding is maar om hoe het gezien wordt. Een computer wordt door sommige wellicht gezien als een hulpmiddel bij het maken van

presentaties, terwijl anderen het zien als een bron van vermaak. Dit verandert niet alleen wat een computer is maar ook hoe de acties omtrent een computer gevormd worden.

Voor autonomie kunnen we een soortelijke redenering formuleren. Software is van invloed op de autonomie van werknemers maar de werknemers kunnen kiezen hoe ze software gebruiken en op die manier bepalen in hoeverre de ontworpen autonomie hen zal beïnvloeden. De structuratietheorie laat echter niet toe de relatie zo simpel te schetsen. Uit figuur 3.1 kunnen we opmaken dat de institutionele context ook nog een rol speelt (via de werknemers). Software die niet past bij de institutionele context van de organisatie zal minder vaak gebruikt worden zoals bedoeld. Voorlopig zullen we ons beperken tot het stellen dat de wisselwerking van invloed is op de mate waarin ontworpen autonomie van invloed is op de autonomie van de gebruiker. We kunnen figuur 3.3 nu als volgt aanvullen:



Figuur 3.4

Om de werking van de in figuur 3.4 weergegeven relatie te illustreren kunnen we wederom het eerder genoemde voorbeeld gebruiken. Hieruit bleek dat de CAD software slechts gebruikt werd als hulpmiddel. Als we ons nu voorstellen dat de genoemde CAD software de autonomie sterk beperkt dan zal de autonomie van de

gebruiker hiervan minder sterk dalen. Hij gebruikt de software immers niet constant. De wisselwerking tussen software, gebruiker en institutionele context heeft als effect dat de software een hulpmiddel wordt gezien. Dit betekent niet alleen dat de gebruiker nog deels de oude werkwijzen gebruikt maar betekent ook dat de nieuwe software niet gebruikt wordt volgens de ontworpen werkwijzen. Het gevolg van dit proces is dat de autonomie van de gebruiker anders is dan de ontworpen autonomie.

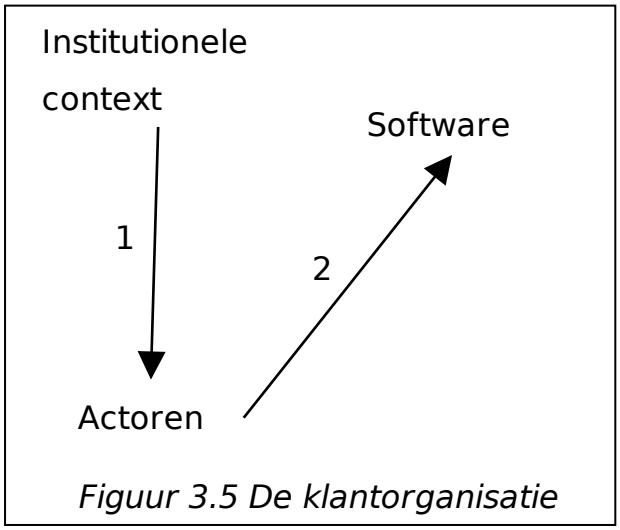
De autonomie die software aan gebruikers geeft, is dus een factor die onderzocht moet worden. Een onderzoek naar autonomie en software moet echter ook duidelijk maken in hoeverre software aansluit op de institutionele context en de actoren binnen de klantorganisatie. Zou de CAD software beter binnen de organisatie passen en zouden individuen ervoor kiezen deze intensiever te gebruiken, dan is het mogelijk dat de autonomie die in de CAD software besloten ligt van grotere invloed zal zijn op de uiteindelijke autonomie van de gebruikers.

In het vervolg van dit hoofdstuk wordt het bovenstaande gebruikt als uitgangspositie voor het verder verkennen van de relatie tussen softwareproductie en autonomie. We gaan ervan uit dat de ontworpen autonomie, de autonomie van de gebruiker beïnvloedt. Deze relatie wordt echter beïnvloedt door de wisselwerking van software, werknemers en institutionele context als interveniërende variabele. De aanname is dat naarmate de software beter past bij de organisatie die deze gebruikt, de ontworpen autonomie meer zal lijken op de autonomie van de gebruiker.

3.2.3 softwareproductie

Alvorens in te gaan op deze productie van software moeten twee begrippen ingevoerd worden om verwarring te voorkomen. In het softwareproductieproces gaan we uit van twee partijen. De partij die

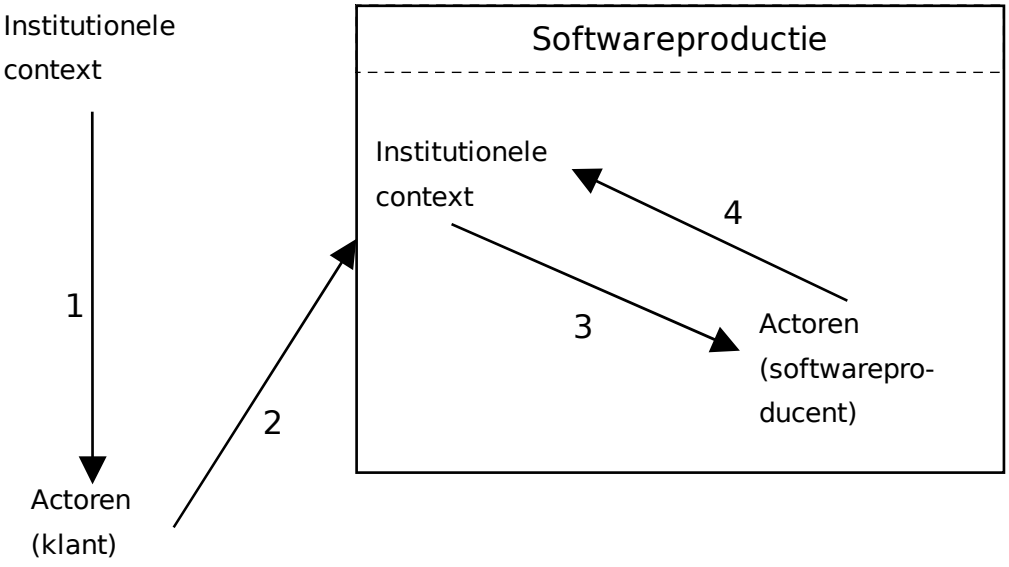
de software afneemt en gaat gebruiken wordt vanaf nu de *klantorganisatie* genoemd. Om de andere partij aan te duiden wordt de term *softwareorganisatie* gebruikt. Deze partij maakt de software en levert deze aan de klantorganisatie. We beginnen met het softwareproductieproces bij de klantorganisatie.



Figuur 3.5 De klantorganisatie

In figuur 3.5 is het proces van softwareontwikkeling binnen de klantorganisatie zichtbaar gemaakt. Het gaat hier niet om de uiteindelijke effecten van de software maar om de ontwikkeling van de software. De institutionele context is van invloed op de vorm van de software maar alleen via actoren. Deze zijn immers degenen die in contact treden met de ontwikkelingsorganisatie. Het belang van dit schema ligt dan ook in het feit dat de aspecten van de institutionele context alleen in de software naar voren komen door de ogen van actoren binnen de klantorganisatie die in contact staan met de ontwikkelaar (pijl 1 in figuur 3.5). Tevens zullen de wensen van gebruikers alleen meegenomen worden als deze contactpersonen dit wenselijk achten en zullen het wederom hun interpretaties van die meningen zijn (pijl 2 in figuur 3.5).

Dit is echter pas het eerste deel van de softwareontwikkeling, de wensen van de klant zijn de eerste input in het proces van softwareontwikkeling. Binnen de software producerende organisatie spelen ook actoren en institutionele context een rol, maar is technologie het product in plaats van een factor binnen de interactie tussen actoren en institutionele context.



Figuur 3.6 De klantorganisatie en de softwareproducerende organisatie

Zoals blijkt uit deze figuur wordt de structuratietheorie zo ingezet dat duidelijk is welke factoren van belang zijn voor dit onderzoek. Niet alleen willen we weten welke inputs een softwareorganisatie binnenkrijgt, we willen ook weten wat er in de softwareorganisatie gebeurt. Software wordt binnen dit proces gezien als een sociaal product wat tot stand komt door de interacties binnen deze organisatie.

Als we nog een keer naar figuur 3.4 kijken dan zien we als eerste variabele de ontworpen autonomie. Zoals eerder werd aangegeven duidt deze term op aspecten van software die autonomie van de gebruiker mogelijk beïnvloeden. Omdat het hier gaat om aspecten van software moeten deze zaken onderzocht kunnen worden aan de hand van het model van figuur 3.6.

De tweede variabele in figuur 3.4 is de wisselwerking tussen software, institutionele context en gebruikers. Al eerder werd gewezen op het feit dat software die beter past bij de klantorganisatie meer invloed zal hebben op de autonomie. Hoe goed software bij een organisatie past is een aspect van software en

ook deze variabele is dus onderhevig aan het model zoals dat geschetst is in figuur 3.6. Sterker nog, de hele linkerhelft van figuur 3.6 toont de mogelijkheid voor klantorganisaties om software aan te laten sluiten op hun institutionele context en werknemers.
Kort samengevat zijn twee aspecten van software van invloed op autonomie van de gebruiker: ontworpen autonomie en de match tussen software en klantorganisatie. Omdat beide aspecten van software zijn krijgen ze in eerste instantie vorm tijdens het softwareproductieproces en kan voor beide onderzocht worden hoe ze tijdens dit proces vorm krijgen.
We hebben nu bijna de theorie waar we naar op zoek zijn. Het doel van dit onderzoek is meer begrip krijgen in de relatie tussen softwareproductie en autonomie door analyse van de productie van software. We zijn op zoek gegaan naar factoren die van belang zijn bij een dergelijk onderzoek. De structuratietheorie leert ons dat hier twee factoren van belang zijn. Omdat software een deel vormt van de interactie tussen context en individuen is niet alleen de autonomie die in de software besloten ligt van belang maar ook is van belang in hoeverre de software aansluit op de institutionele context en de werknemers van de klant. De laatste stap in de theorie is de aanname dat ook de ontwikkeling van software zelf begrepen kan worden als een interactie tussen context en individuen bij de softwareproducent. Het is deze interactie waarin we geïnteresseerd zijn, door deze interactie te onderzoeken moet duidelijk worden hoe softwareproductie de autonomie van werknemers beïnvloedt.

3.2.4 De drie modaliteiten van de structuratietheorie

Dat in de bovenstaande paragrafen besloten is om de softwareproductie te analyseren door te kijken naar de interactie tussen individuen en context betekent nog niet dat deze direct

onderzocht kan worden. Om tot een goed resultaat te komen is nog meer structuur nodig binnen het proces van softwareproductie. Het doel van het onderzoek is duidelijk, maar de conceptuele handvatten om het bedrijfsproces binnen de ontwikkelorganisatie te onderzoeken missen nog.

De structuratietheorie biedt echter een aantal concepten die ons hierbij kunnen helpen. Giddens stelt dat alle interacties tussen institutionele context en individuen plaatsvinden binnen drie deelgebieden: de zogenaamde modaliteiten van de structuratietheorie. Giddens wijst op het feit dat deze modaliteiten alleen analytisch los van elkaar gezien kunnen worden. De eerste modaliteit die Giddens noemt zijn interpretatieve schema's. Individuen handelen op een bepaalde wijze door de manier waarop zij de werkelijkheid zien, handelen is een proces van zingeving. Door dit handelen bevestigen zij weer het schema van interpretatie wat ze aanhangen. Acties kunnen echter ook legitiem en niet legitiem zijn. De modaliteit die hierbij hoort zijn normen. Uit de context halen individuen informatie over welke acties wel mogen en welke niet, door naar deze normen te handelen wordt de norm versterkt. Ten slotte volgt de modaliteit die Giddens facility noemt. Personen oefenen door bepaalde acties uit te voeren macht uit. De context biedt bepaalde vormen van dominantie en door het aanwenden van macht wordt de dominantie structuur herbevestigd (Giddens, 1986).

Het doel van dit onderzoek is het onderzoeken van de ontwikkeling van software, we gaan daarom niet in op de volledige theorie van Giddens. Het is van belang dat we ons hier realiseren dat de interactie van context en individuen analytisch opgedeeld kan worden naar interpretatie, macht en legitimatie.

In dit kader stellen Orlikowski en Robey voor om onderzoek naar softwareproductie op een wijze te structureren die deze modaliteiten centraal stelt. Zij wijken echter af van de modaliteiten zoals ze hierboven beschreven zijn. Zij breiden de factor macht

enigszins uit naar iets wat waarschijnlijk beter verwoord kan worden als mogelijkheden (1991). Individuen worden in hun acties niet alleen beperkt door macht van personen maar ook door de beperkte mogelijkheden om acties uit te voeren. Gedacht kan worden aan geld, tijd, gereedschap en dergelijke. Deze uitbreiding kan nuttig zijn voor organisaties om ook acties te verklaren die ondernomen worden door de aanwezigheid van middelen om deze acties uit te voeren.

Het voorstel van Orlikowski en Robey is weergegeven in tabel 3.1. Deze tabel moet begrepen worden als een blauwdruk voor onderzoek. Voor de drie modaliteiten kan zowel beschreven worden hoe de institutionele context de acties van individuen beïnvloedt (de rij ‘Realm of Social Structure’ in tabel 3.1) als de invloed die de acties hebben op de institutionele context (oftewel de Realm of Human Action). De zes cellen in figuur 3.8 representeren processen van interactie binnen de organisatie en zo blijft het doel van de structuratietheorie behouden. Door op zoek te gaan naar interacties kunnen ook bedrijfsprocessen begrepen worden zonder te focussen op enkel individuen of structurele kenmerken.

Het schema is in dit geval al ingevuld door Orlikowski en Robey met de mogelijke bevindingen voor een onderzoek. Hier moet opgemerkt worden dat deze voorbeelden alleen gericht zijn op programmeurs maar er is geen reden om aan te nemen dat dit niet kan gelden voor een gehele softwareorganisatie.

Modalities	Interpretive Schemes	Power	Norms
Realm of Social Structure	System Developers are informed by system development methodologies and knowledge about their organization to build information systems	System Developers work within the constraints of time, budget, hardware, software, and authority to build information systems	System Developers draw on the values and conventions of their organization, occupation, and training to build information systems
Realm of Human action	System Developers create meaning by programming assumptions and knowledge into the information systems	System Developers build information systems through the organizational capabilities or power they wield in their organizational roles	System Developers create sanctions by designing and programming legitimate options and conventions into information systems

Tabel 3.1. Structuratie in systeemontwikkeling (Bron: Orlikowsi en Robey: 1991: 23)

Het nut van dit model blijkt uit een onderzoek van Fredriksen en Rose. Zij onderzochten volgens de bovenstaande methode een software bedrijf in Denemarken. Uit hun resultaten blijkt dat er binnen het bedrijf een scheiding is in onderhoud en ontwikkeling (oftewel projecten). Deze interpretatie bevindt zich in de institutionele context en wordt steeds herbevestigd door bepaalde programmeurs aan projecten te laten werken terwijl een andere groep enkel onderhoud doet (2003).

Het belang van dit onderscheid wordt duidelijk als de normen bekeken worden. Projecten blijken als veel belangrijker gezien te worden dan onderhoud terwijl onderhoud vaak meer geld oplevert. Tevens is het halen van deadlines een belangrijker norm dan het leveren van kwaliteit. Deze structuur wordt constant voorgezet (in de ‘realm of human action’) door het handelen naar deze structuur; kwaliteitsdoelen worden constant opzij gezet om deadlines te halen (2003).

Wat betreft de machtsverdeling stellen Frendirksen en Rose vast dat personen die hogerop komen in de organisatie uitsluitend afkomstig zijn uit de groep werknemers die zich met projecten bezig houdt. Zo wordt door acties van machtsverdeling de norm dat projecten belangrijker zijn steeds weer bevestigd (2003).

Het bovenstaande laat goed zien hoe de structuratietheorie nuttig kan zijn bij de analyse van softwareontwikkeling. Niet alleen wordt duidelijk wat de structuur is in een bepaald bedrijf maar ook hoe deze wordt gecontinueerd door de acties van individuen. Tevens wijzen Fredriksen en Rose op de gevolgen van de structuur voor het bedrijf en het product. De nadruk op deadlines betekent bijvoorbeeld minder aandacht voor kwaliteit en vermindert de macht van de eindgebruiker (2003).

Voor dit onderzoek betekent het gebruik van de structuratietheorie het volgende. Het proces van softwareontwikkeling is een samenspel van verschillende actoren en de institutionele context. Het samenspel tussen acties van de verschillende actoren en de institutionele context kan beschreven worden naar de drie modaliteiten van de structuratietheorie. Het softwareproduct is een resultaat van dit samenspel. Een conventionele theorie zou de software kunnen zien als een resultaat van de organisatiestructuur of de acties van verschillende actoren of zelfs het gevolg van een bepaalde bedrijfscultuur. Dit gezichtspunt laat de ruimte om al deze zaken mee te nemen in het onderzoek. De software krijgt op die

manier een centrale rol tussen individuen en structuur. Tevens laat het onderzoek zien dat de analytische scheiding van de modaliteiten veel oplevert maar dat juist de interactie tussen domeinen van belang is. Projecten worden qua zingeving gezien als verschillend van onderhoud maar het onderscheid wordt pas echt van belang als duidelijk wordt dat ze via normen ook als belangrijker gezien worden.

3.3 De theorie

De basis van het conceptuele kader van dit onderzoek ligt in het feit dat software gezien wordt als een sociaal product opgebouwd door de interactie tussen individuen en institutionele context. Door deze interactie te onderzoeken kan verklaard worden hoe bepaalde aspecten van de software tot stand komen.
Voor het voorspellen van de autonomie van gebruikers moeten twee van deze aspecten onderzocht worden. De ontworpen autonomie is van invloed op de autonomie van de gebruiker, hoe meer software de autonomie beperkt hoe kleiner de autonomie van gebruikers zal zijn.
Deze relatie wordt echter beïnvloedt door de confrontatie van software en klantorganisatie. Het resultaat van deze confrontatie is moeilijk te voorspellen maar is deels afhankelijk van de pogingen die tijdens de softwareproductie zijn ondernomen om software aan te passen op de klantorganisatie. Ook dit is een aspect van software, de vorm van dit aspect wordt deels bepaald tijdens de softwareproductie.
Door te onderzoeken hoe de ontworpen autonomie vorm krijgt en vervolgens te onderzoeken hoe de software op de klantorganisatie wordt aangepast wordt moet het mogelijk zijn meer inzicht te krijgen in de invloed van softwareproductie op autonomie van de eindgebruiker. Door dit onderzoek te verrichten met de interactie

tussen de drie modaliteiten als leidraad is het mogelijk om zowel de structurele en culturele kenmerken van het softwareproductieproces te doorgronden.

De volgende hoofdstukken zijn gewijd aan het toetsen van deze theorie aan de werkelijkheid. In het volgende hoofdstuk gaan we in op de methode die gebruikt is om dit te bereiken.

4 Methode

Het vorige hoofdstuk heeft een visie op de werkelijkheid opgeleverd, deze visie kan via onderzoek getoetst worden aan de werkelijkheid. Het is dan ook het doel van dit hoofdstuk om de theorie zodanig te vertalen dat deze onderzocht kan worden.

Een belangrijk aspect van dit onderzoek is gebaseerd op het analyseren van de interacties binnen de organisatie. Er moet duidelijk blootgelegd worden hoe de beslissingen over de vorm van de software ontstaan uit de interactie tussen institutionele context en acties van verschillende stakeholders. Dit begrip van interactie is moeilijk te bereiken door het gebruik van een onderzoeksmethode die vooraf het bestaan van bepaalde variabelen aanneemt of uitsluit. Het zijn juist de onbekende factoren in het beslissingsproces die het onderzoek waarde geven. Dit sluit in principe elke vorm van kwantitatief onderzoek uit.

Er zijn een aantal kwalitatieve onderzoeksmethodes die het mogelijk maken om organisatieprocessen te onderzoeken. Binnen de structuratietheorie spelen individuele interactieprocessen een grote rol. De structuren die hierbij van invloed zijn hoeven in principe niet objectief blootgelegd te worden. Het belang ligt juist in de manier waarop de structuur van invloed is op de acties van individuen, het belang van de structuur is hoe individuen er op reageren. Dit feit maakt de keuze om individuen en hun kijk op de werkelijkheid centraal te stellen in de onderzoeksmethode verantwoord. De stap naar semigestructureerde interviews als onderzoeksmethode is vervolgens niet meer dan logisch. Er zijn immers wel een aantal centrale vragen die beantwoord dienen te worden maar er moet genoeg ruimte zijn om onverwachte variabelen te benoemen.

4.1 Onderzoeksobject

Tot nu toe is steeds duidelijk geweest dat softwareproductie het onderwerp is van dit onderzoek. Het is echter nog niet duidelijk gemaakt wat precies onderzocht wordt. De theorie uit het vorige hoofdstuk wijst op het belang van het softwarebedrijf in de ontwikkeling van software maar legt ook een belangrijke rol bij de klantorganisatie voor het communiceren van de insitutionele context. In dit onderzoek is echter gekozen om deze laatste groep niet mee te nemen in dit onderzoek.
Een keuze voor het onderzoeken van klant en leverancier zou het onderzoek duwen in de richting van een case study. Deze methode kan natuurlijk zeer waardevolle resultaten opleveren maar in dit onderzoek is gekozen voor het onderzoeken van meerdere softwareorganisaties om de waarde van de structuratietheorie in verschillende scenario's te kunnen testen. De verwachting is dat dit uiteindelijk meer informatie zal geven over de geldigheid van de structuratietheorie voor dergelijk onderzoek.

4.2 Interviews

In de theorie is een duidelijke scheiding aangebracht tussen twee factoren die gezien kunnen worden als aspecten van software. Deze factoren zijn de ontworpen autonomie zelf enerzijds en de aanpassing van de software op de klantorganisatie anderzijds. Deze analytische scheiding zal ook gebruikt worden bij het onderzoeken van de theorie zelf.
De bovenstaande vaststelling betekent dat het onderzoek ook uit twee delen moet bestaan. In het eerste deel is onderzocht hoe de inputs van de klant invloed hebben op de vorm van de software. Het gaat hier om de software in het algemeen en nog niet om autonomie. Het doel van dit deel is namelijk een antwoord vinden op de vraag hoe tijdens de softwareproductie de interactie tussen

context en individuen de software aan doet sluiten op de klantorganisatie. Of simpeler gesteld:

- Wat gebeurt er tijdens softwareproductie om de software goed te laten passen bij de klant?

Deze vraag is van belang om te begrijpen waarom software niet altijd de verwachte effecten heeft op de autonomie van werknemers.

Het tweede deel van de interviews gaat om de ontworpen autonomie. Factoren die de gebruiker beperken of vrijlaten in het uitvoeren van taken die de software mogelijk maakt. Ook hier is aangenomen dat deze factoren tot stand komen door een proces van interactie tussen context en individuen. Oftewel:

- Wat gebeurt er tijdens softwareproductie om de gebruiker in zijn of haar werkzaamheden te beperken of vrij te laten.

Beide vragen kunnen direct gesteld worden maar het is beter voor de kwaliteit van de antwoorden om iets meer structuur aan te brengen in de vragen, zonder hierdoor de kans op onverwachte antwoorden uit te sluiten. Gelukkig heeft Giddens ons al voorzien van een kader wat mogelijk de structuur van dit interview kan bieden. In het vorige hoofdstuk zijn steeds drie modaliteiten genoemd die verbonden zijn aan twee concepten. Deze zijn weergegeven in de tabel 4.1

	Interpretatieve schema’s	Macht en middelen	Normen
Institutionele context	1	3	5
Acties individuen	2	4	6

Tabel 4.1

In deze tabel zijn horizontaal de drie dimensies van de structuratietheorie terug te vinden. Tevens is een onderscheid gemaakt via de verticale categorieën. De niet gearceerde cellen bepalen wat uiteindelijk onderzocht gaat worden. Ze worden hier niet als zodanig geformuleerd maar het zijn eigenlijk de zes vragen die gesteld zijn tijdens het onderzoek. De vragen zijn niet direct gesteld om een open interview te behouden en mogelijke antwoorden of opmerkingen van de respondent niet bij voorbaat uit te sluiten. Om duidelijk te maken hoe dit is bewerkstelligd gaan we eerst wat dieper op het schema in.

Het doel is hier een beter begrip krijgen in de processen die tot een bepaald resultaat leiden. De scheiding tussen interpretatieve schema's, macht en normen geeft aan welke drie gebieden, of modaliteiten, onderzocht zijn. De scheiding is aangebracht om recht te doen aan de verschillende interacties. De drie interacties hebben elk een eigen context en eigen individuele acties. Zo kan in elk domein de vraag gesteld worden hoe de context de individuele acties beïnvloedt (de cellen 1,3 en 5) in tabel 4.1 en hoe de acties de institutionele context bevestigen of juist afbraak doen aan deze (cellen 2,4 en 6). Via deze methode is de kern van de structuratietheorie omgezet in een onderzoekbaar geheel. Individuen worden in hun gedragingen beïnvloed door de institutionele context maar dragen door hun acties ook weer bij aan deze context.

In dit onderzoek is het proces van softwareproductie onderzocht via de twee deelgebieden die in de theorie geformuleerd zijn. Het bovenstaande schema is voor dit onderzoek dan ook tweemaal gebruikt. Eenmaal om antwoord te krijgen op de vraag hoe kenmerken van de klantorganisatie meegenomen worden in het ontwerp van de software. Een tweede maal is het schema gebruikt om te onderzoeken hoe autonomie in de software tot stand komt.

De opmerking dat dit schema gebruikt wordt verdient echter wat meer uitleg. Het doel is steeds geweest om de deelgebieden van dit schema zo goed mogelijk aan de werkelijkheid te koppelen door het stellen van vragen aan personen die verstand hebben van en ervaring hebben met deze werkelijkheid. Dit betekent echter niet dat direct is gevraagd naar de factoren zoals ze in tabel 4.1 beschreven staan. Hiervoor zijn twee redenen, ten eerste kan een dergelijke directe vraag het interview onnodig ingewikkeld maken. Een vraag als “Hoe bepalen normen uw keuzes in het programmeer proces” sluit weliswaar goed aan op de theorie maar is moeilijk te begrijpen voor een respondent.

Ten tweede kan het stellen van vragen naar bepaalde processen bepaalde antwoorden uitsluiten die mogelijk van belang zijn voor het onderzoek. Het direct focussen op middelen zou kunnen veroorzaken dat bepaalde invloeden op keuzes niet genoemd worden.

De andere kant van deze overwegingen is echter dat het wel van belang is dat alle aspecten van de structuratietheorie aan bod komen in het interview. Het is om deze redenen dat gekozen is voor interviews die naarmate het interview vordert gestructureerder worden. In eerste instantie is gevraagd hoe het algemene proces van een klantwens naar opgeleverde software verloopt. Deze vraagt biedt de respondent de mogelijkheid om het proces te beschrijven zonder gehinderd te zijn door opgelegde structuren.

Vervolgens is aan de hand van het antwoord van de respondent ingegaan op de verschillende stappen in dit proces waarbij steeds de vraag centraal heeft gestaan waarom het (deel)proces op een bepaalde manier is ingericht. Dit is in wezen een verdere concretisering van de vraag hoe acties beïnvloedt worden door de institutionele context van de organisatie en hoe de context bepaalt wordt door acties van individuen. Bij het bespreken van een dergelijk (deel) proces is tevens gevraagd naar andere mogelijke

invloeden die de respondent niet aandraagt in de vorm van interpretaties, macht en middelen en normen.
De keuze van deze methode betekent dat het uiteindelijk aan de onderzoeker is om de resultaten te interpreteren en categoriseren naar de drie modaliteiten en de twee aspecten van interactie. Dit is echter niet zo vreemd, een onderzoeker bezit de kennis en het inzicht om tijdens het interview in te springen op belangrijke zaken en deze achteraf te organiseren. Het alternatief zou zijn om de vragen vooraf sterk te structureren, dit zou echter betekenen dat het aan het inzicht van de respondent ligt om het juiste antwoord zo volledig mogelijk te geven.
Kort samengevat wordt het bovenstaande schema gebruikt als hulpmiddel van de onderzoeker om vragen te structureren maar het ligt niet vast hoe het vraaggesprek zal verlopen. Achteraf wordt de tabel ingezet om de data samen te vatten en tot een logisch geheel te structureren. Het vraaggesprek kent twee algemene onderwerpen: het aanpassen van software op de klant en het inbouwen van autonomie in de software.

4.3 Sampling

De onderzochte organisaties zijn verzameld worden via de sneeuwbalmethode. Softwarebedrijven weten zelf wie hun concurrenten en partners zijn. Deze methode bood de mogelijkheid om bedrijven te vinden die voldoen aan de belangrijkste eis van dit onderzoek: ze moeten software ontwikkelen die bedoeld is voor intern gebruik bij organisaties. Omdat er zeer veel organisaties zijn die wel software ontwikkelen maar wellicht niet voldoen aan deze eis bleek het logischer om gebruik te maken van de kennis die softwarebedrijven al hebben.
Bij elke organisatie is gestreefd naar het interviewen van twee personen, een persoon die zich bezig houdt met het contact met de klanten en een respondent die zich bezig houdt met het werkelijk

ontwikkelen en programmeren van de software. Het kan zijn dat deze twee functies zich bevinden in een en dezelfde persoon. In dat geval heeft er maar één interview plaatsgevonden. De klant zelf is de enige stakeholder die in dit proces niet is meegenomen. Dit onderzoek is echter op zoek naar algemene tendensen en daarom is gekozen om het de softwareproducent als onderzoeksobject te gebruiken. Dat maakt het mogelijk om uitspraken te doen over de bedrijfstak. De resultaten van dit onderzoek zijn gepresenteerd in het volgende hoofdstuk.

5 Resultaten

Het doel van dit onderzoek is een theorie formuleren die de werkelijkheid goed kan beschrijven. De theorie die is geformuleerd in het derde hoofdstuk is in het vorige hoofdstuk verder uitgewerkt tot een onderzoekbaar geheel. Het doel van dit hoofdstuk is de resultaten van dit onderzoek presenteren. Pas in het laatste hoofdstuk wordt de theorie zelf weer aangehaald om te bekijken hoe bruikbaar deze is gebleken.

De theorie kent twee belangrijke delen. De eerste is de manier waarop software wordt aangepast op de klantorganisatie. Het tweede deel stelt de vraag welke processen een rol spelen bij het vormen van de autonomie binnen de software. Dit hoofdstuk is dan ook naar deze twee delen opgedeeld. Bij het presenteren van de resultaten zijn deze verder opgedeeld naar de drie modaliteiten van de structuratietheorie. We vangen echter aan met een bespreking van een aantal algemene kenmerken.

5.1 Algemene kenmerken

In totaal zijn er zes interviews gehouden bij vier bedrijven. Bij twee bedrijven zijn twee personen geïnterviewd. In beide gevallen is één van deze twee een leidinggevende en de ander een programmeur. Bij de bedrijven waar maar één persoon ondervraagd is, vervult deze persoon beide rollen. Om de resultaten van dit onderzoek eenduidig te presenteren is constant de term leidinggevende gebruikt om te duiden op de persoon die zich bezighoudt met de coördinatie van het ontwikkelingsproject.

De bedrijven hebben tussen de drie en acht mensen fulltime in dienst. Twee van de bedrijven houden zich voornamelijk bezig met

het maken en onderhouden van CMS[3] systemen. Eén bedrijf heeft als hoofdactiviteit het leveren van CRM[4] systemen. In de interviews is alleen ingegaan op deze hoofdactiviteiten. Eén ontwikkelt allerlei applicaties voor bedrijven zonder dat gesproken kan worden van een trend hierin.

Tevens is gebleken dat alle organisaties grofweg dezelfde fasen doorlopen bij het ontwikkelen van de software. Allereerst is er een intakefase met de klant welke een set met doelen volgt, een functioneel ontwerp. Vervolgens wordt dit ontwerp omgezet in een technisch ontwerp waaruit duidelijk blijkt wat daadwerkelijk geprogrammeerd gaat worden. Daarna wordt de software zelf gemaakt. Na de testfase waarin de opdrachtgever de kans krijgt terugkoppeling te geven over de software volgt ten slotte de implementatie. In deze fase wordt de software opgeleverd en is deze klaar voor gebruik bij de klant. Deze fasen zijn niet altijd even duidelijk te onderscheiden en kunnen elkaar overlappen maar ze zijn altijd in deze chronologische volgorde gevonden.

5.2 Aanpassing software en de klantorganisatie

Het doel van dit deel van het onderzoek was het in kaart brengen van het proces waarbij software wordt aangepast op verschillende kenmerken van de klantorganisaties. Uit de interviews is gebleken dat de initiatieven om de match van de software met de klantorganisatie te verbeteren terug te brengen zijn tot twee specifieke gebieden. De inventarisatie van de wensen van de klant en het presenteren van de software aan de klantorganisatie. In de volgende paragrafen zal voor de drie domeinen van de

[3] CMS staat voor Content Management System, een digitale omgeving waarin individuen met een minimum aan technische kennis de inhoud van internetpagina's aan kunnen passen.

[4] CRM (Customer Relation Management) systemen stellen gebruikers in staat klantgegevens bij te houden en deze op verschillende wijzen te organiseren en te gebruiken.

structuratietheorie besproken worden wat deze gebieden precies in houden en hoe de acties op de gebieden tot stand komen in de organisaties. Aan het eind van deze bespreking wordt een en ander samengevat in een tabel.

5.2.1 Interpretatieve schema's

In dit domein is bekeken hoe de structuur van de organisatie en de acties van individuen leiden tot bepaalde keuzes in het ontwerpproces. Hier wordt met name gelet op de visies die verschillende actoren op software hebben. Wat zien zij als het doel van de software en hoe komt dit terug in het product.
De eerste context waar een softwarebedrijf mee te maken heeft zijn de wensen van klanten. Deze heeft eigen visies op software en probeert deze waarschijnlijk terug te laten komen in het eindproduct.
Het contact met de klant verloopt in bijna alle gevallen via één persoon van de klantorganisatie. Dit is zelden een lagere gebruiker maar meestal een leidinggevende. Aan de andere kant is het ook bijna altijd de leidinggevende van de ontwikkelingsorganisatie die contact heeft met de klantorganisatie tijdens het eerste proces van vormgeving. De klant komt in de meeste gevallen met een algemeen probleem waarvoor een oplossing gevonden moet worden. Dit probleem is in de meeste gevallen niet sterk gedefinieerd.

"We hebben [...] in totaal één klant gehad die echt een lijst van A tot Z had van dit moet het gaan doen. [...] Maar over het algemeen is het zo dat een klant inderdaad bepaalde wensen heeft en dat het wel geformuleerd wordt maar dat het door ons helemaal ontcijferd moet worden. Wat zijn nou daadwerkelijk de wensen van de klant. Het ligt niet echt op een presenteerblaadje."

Deze leidinggevende ziet het dan ook als zijn taak om deze wensen te vertalen in een concept van wat de klant echt nodig heeft. In alle gevallen wordt dit proces gevormd door een serie vraaggesprekken met de klant.

“De ene klant weet precies wat hij wil en heeft eigenlijk al bijna de vragen ingevuld. Ik heb dit nodig en dat wil ik zo en zo en zo gemaakt hebben. Het is vaak niet de beste oplossing maar ze weten dan wat dingen en denken dat het daarmee kan. Vaak kan dat ook wel maar hebben wij een betere oplossing. Dus dan komen we met een aangepast voorstel, dus een voorstel zoals wij denken dat het goed is. “

Uit dit proces blijkt dat de ontwikkelaar weliswaar integer omgaat met de wensen van de klant maar deze wel aanpast naar eigen inzicht. De ontwikkelaar brengt in het vertalen van de een probleem ook eigen kennis van zaken mee en dit vindt zijn neerslag in de uiteindelijke oplossing.
Wat zingeving betreft kunnen we voor de leidinggevende concluderen dat deze simpele problemen uit de context (in dit geval de wensen van de klant) haalt en deze door interactie met deze context omzet in problemen die daadwerkelijk door software op te lossen zijn.

De programmeur ziet de software veelal als een set functionele eisen waarbij bekeken moet worden hoe ze opgelost moeten worden. Het doel van het programma verdwijnt hier niet geheel naar de achtergrond maar het technische verhaal treedt meer naar de voorgrond.

“Wij proberen de wensen van de klant altijd in het algemene CRM neer te zetten zodat wij niet een nieuwe versie neer hoeven te zetten en daar aparte wijzigingen in door moeten voeren. Dus soms zwakken we wensen van klanten af zodat het in ons systeem past. En soms gaat dat gewoon niet en moet je gewoon aparte wijzigingen doorvoeren in het systeem.”

De interpretatie van het programma kan dus wat betreft de programmeur gezien worden als een technische uitdaging om wensen van de klant op te lossen. De programmeur ziet dus niet alleen de problemen van de klant maar beziet deze ook uit een technische context, waardoor zijn acties veel meer gericht zullen zijn op het technisch haalbaar maken van de doelen.

5.2.2 Middelen

Het belang van dit domein ligt in de macht die actoren bezitten om de vorm van de software te beïnvloeden. Dit is van belang om te doorgronden welke individuen de mogelijkheid hebben om met hun acties ook daadwerkelijk resultaat te boeken.
Uit het bovenstaande werd al enigszins duidelijk dat de meeste macht bij het vormen van de software ligt bij de interactie tussen de leidinggevende van de ontwikkelingsorganisatie en de opdrachtgever. Deze beide bezitten de meeste middelen om de vorm van de software te bepalen.
Alle onderzochte bedrijven bezitten een structuur waarbij intakegesprekken de eerste vorm van de software bepalen, het zogenaamde functioneel ontwerp. Dit voorbeeld is de beschrijving van het productieproces door een programmeur.

“Zelf heb ik eigenlijk weinig van doen met de klant. De accountmanager maakt het eerste contact en omdat de

accountmanager ook vaak de projectleider is gaat hij al eerste met de klant om de tafel zitten om zoveel mogelijk te analyseren wat de klant daadwerkelijk wil. En uiteindelijk komt daar weer in samenspraak met de techniek en, afhankelijk van hoe gecompliceerd het is, wordt ik daar dan ook bij betrokken. Daar volgt dan een functioneel ontwerp uit. En als dat ontwerp is goedgekeurd door de klant en de afdeling techniek. Dan wordt het functionele ontwerp vertaald naar het technisch ontwerp."

Dit voorbeeld beschrijft een redelijk typische situatie. Het doel van de gesprekken met de klant is een functioneel ontwerp. Voor dit ontwerp wordt de programmeur eventueel gevraagd of alles technisch mogelijk is maar in principe bepalen de leidinggevende en de klant de vorm van dit ontwerp.
Deze formele structuur is echter niet de enige verdeling van middelen die van belang is bij de vorming van de software. De leidinggevende en hun klant worden in hun middelen beperkt door zaken als geld, tijd en mogelijke oplossingen.

"We hebben niet oneindig veel mensen hier werken dus het houdt ergens op. Sommige dingen die een klant wil zijn inderdaad heel moeilijk te maken dus daar ben je lang mee bezig. En vaak heeft die klant daar ook geen geld voor."

Dit proces betekent een serieuze afbakening in wat de werkelijke oplossing kan bevatten en dus ook een beperkte mogelijkheid om de software aan te laten sluiten op de klantorganisatie.
Ook de programmeur kent de beperkingen. Met techniek is niet alles mogelijk. Door inzicht in deze beperkingen krijgt de programmeur hindermacht in het ontwerpproces.

“[…] een klant kan denken dat hij een goede oplossing bedacht heeft maar dan blijkt dat dat voor internetcommunicatie inefficiënt is. Dus bijvoorbeeld een klant wil een fotoalbum en als je met de cursor over een thumbnail rolt dan wordt de foto in het groot naast de thumbnails weergegeven. Ja dat kan de klant wel bedacht hebben maar als wij dan daarop reageren dan zeggen we van “Ja dat gaan we dus niet doen”want als je met die cursor over de foto’s heen gaat dan gaat dat ding dus grote foto’s downloaden. Dus hebben we die klant aangeraden, doe dat dan als je erop klikt.”

De unieke kennis van de programmeur over wat mogelijk is binnen de voorwaarde van een goed werkend systeem maakt dus dat deze ook zijn macht kan laten gelden over de vorm van de software. Er zijn echter geen aanwijzingen gevonden dat dit verder gaat dan de technische aspecten van het proces.
Wat wellicht nog belangrijker is, zijn de standaard systemen die gebruikt worden voor het leveren van de software. Deze beperken in verschillende mate de vormgeving van de software en worden gebruikt bij drie van de vier onderzochte bedrijven. Deze standaard systemen leveren het voordeel op dat niet voor elke klant het wiel opnieuw uitgevonden hoeft te worden maar hebben als nadeel dat er standaard mogelijkheden uitgesloten worden.

“Het [standaard systeem] *is simpel en het heeft een aantal randvoorwaarden waarbinnen je je kan bewegen. Dus er zit bijvoorbeeld geen workflow in. Als klanten workflow willen dan zeggen we ‘ja sorry, maar dan moet je dat niet met ons CMS doen’. Als klanten een webwinkel willen: ‘ook sorry maar dat zit niet in ons CMS’. Dus je hebt een aantal randvoorwaarden waarbinnen je je kan bewegen”*

Het bovenstaande antwoord is echter niet altijd de standaard, het volgende voorbeeld laat zien dat het ook anders kan.

“Als ze alleen een basissysteem willen, krijgen ze alleen een gebruikersnaam en wachtwoord en kunnen ze zelf aan de slag. Zitten er wijzigingen in dan ligt het een beetje aan hoeveel wijzigingen. Zijn het een paar namen van velden gewijzigd, ja dat is zo gedaan. Dat wordt ook direct gedaan. Zijn het volledige nieuwe functionaliteiten dan gaat het eventueel over de periode van een maand.”

Het moge duidelijk zijn dat zodra er een standaard is deze ook van invloed zal zijn op de uiteindelijke software. Het standaard systeem zal namelijk altijd functioneren als een fundering voor de software. Het is echter niet duidelijk in hoeverre dit standaard systeem aan te passen is aan de klantorganisatie.
We weten nu dat alle partijen gebonden zijn aan de standaard structuur van het bedrijf waarin programmeurs een adviserende rol hebben in het ontwerpproces. Ze kunnen dit advies echter aanwenden om op technische moeilijkheden te wijzen. Leidinggevenden en klanten hebben meer macht maar worden op hun beurt beperkt door geld, tijd en de standaard systemen die gebruikt worden. Dit hoofdstuk gaat echter over de match tussen software en de klantorganisatie en het is daarom nuttig om nog even stil te staan bij de invloed die verschillende actoren in die organisatie hebben.

De rol van de werknemers die de software gaan gebruiken komt zelden aan bod bij de ontwikkelaar. Het enige voorbeeld wat hier gevonden is, is de rol die deze groep speelt tijdens de testfase van de software. Hier kan het zijn dat ze input geven over wat zij van de software vinden, het is dan echter alsnog aan de leidinggevende bij

de klantorganisatie om te beslissen of de software nog gewijzigd wordt naar de wensen van de klant.

“Soms in overeenstemming met de opdrachtgever worden ook de mensen die er daadwerkelijk gebruik van gaan maken toegang verleend tot een testomgeving. En dan kan er natuurlijk nog wel in een keer een opmerking komen in de trend van “maar dit begrijpen we niet, of dit klopt niet, of dit ontbreekt” en dan volgt er natuurlijk weer een project uit.”

Mocht een eindgebruiker dus wensen hebben dan worden deze mogelijkerwijs weer opgepakt door de opdrachtgever om het gehele proces van ontwikkeling weer opnieuw te starten. De vorm van het softwareontwerpproces zet de eindgebruiker dus nagenoeg buitenspel. Opmerkingen die ze hebben moeten eerst door de opdrachtgever om verwerkt te worden en zelfs als deze ze oppakt is het de vraag hoe deze een nieuwe ontwikkelproces zullen overleven.

5.2.3 Normen

Wat sterk opvalt bij alle interviews met leidinggevende is hun drang om integer zaken te doen. Ze zijn allen van mening dat een klant helpen bij het oplossen van een bepaald probleem een norm is waaraan voldaan moet worden.

“[…] dan proberen wij als partner, dat is eigenlijk wat wij het liefst ook hebben, niet dat klanten niks weten maar dat we als partner samen gaan nadenken over de beste oplossing. Waar bij het eigenlijk altijd tot de beste oplossing komt. Want eigenlijk zitten we dan in een soort brainstormsessies waarin je gaat nadenken over wat de klant nu eigenlijk wil.[…] het liefst hebben we gewoon

klanten dat we in het hele proces van conceptfase tot eindproduct mee mogen denken."

Hier moet echter opgemerkt worden dat het in alle gevallen ging om het oplossen van het probleem zelf. In geen van de gevallen zijn aanwijzingen gevonden dat deze denkwijzen verder gingen naar bijvoorbeeld autonomie voor de werknemers.
Ook programmeurs lijken deze drang naar klantgerichtheid aan te hangen, zelfs met de bijkomende complicaties van de wensen in het systeem zetten.

"In eerste instantie overleg je met de klant het idee. Daarna moet je zelf nog gaan nadenken over hoe het in het systeem komt. En gaandeweg laat je de klant ook controleren van 'klopt het ook met jullie idee?' ."

Na wat doorvragen blijkt echter dat ook het intact houden van het systeem een belangrijke norm is. Dit blijkt uit de volgende twee voorbeelden.

"Wij proberen de wensen van de klant altijd in het algemene CRM neer te zetten zodat wij niet een nieuwe versie neer hoeven te zetten en daar aparte wijzigingen in door moeten voeren. Dus soms zwakken wensen van klanten af zodat het in ons systeem past. En soms gaat dat gewoon niet en moet je gewoon aparte wijzigingen doorvoeren in het systeem."

"Wij bepalen dan of het generiek toepasbaar is, zo niet dan proberen we het net effetjes aan de passen zodat het wel generiek werkbaar is en anders krijgt de klant een eigen versie."

In principe worden overschrijdingen van het functionele denken dan ook niet toegestaan binnen de organisatie.

“Je merkt ook wel dat als je review doet met een klant dat je daar wel de mening van een programmeur er door hoort. Aan de andere kant is het wel altijd de klant die zeg maar bepaalde wensen of verwachtingen hebben en daar voldoe je aan of daar voldoe je niet aan. Als een programmeur kan uitleggen aan een klant dat het anders kan of moet dan heb je die vrijheid.”

5.2.4 Samenvatting

Zonder over te gaan op een algemene conclusies is het bovenstaande samengevat in tabel 5.1. Hierin is de bovenkant van de tabel een uitsplitsing van de gevonden context en is de onderste rij een weergave van de acties van de verschillende actoren die leiden tot de vorm van het softwareproject. Het is belangrijk hierbij te bedenken dat het doel van deze tabel niet is om het hele ontwerpproces van de software bloot te leggen, maar meer om duidelijk te krijgen welke keuzes gemaakt worden om de software beter aan te laten sluiten op het klantbedrijf. Het is onvermijdelijk dat we hier ook keuzes tegenkomen die niet gemaakt worden met het klantbedrijf in het achterhoofd.

	Interpretatieve schema's	Macht en Middelen	Normen
Institutionele context	**Leidinggevende** ziet wensen van de klant als goed maar onvolledig. **Programmeurs** zien goed werkende software als een technisch systeem en minder als een product voor klant.	**Programmeur** heeft meeste kennis van technische mogelijkheden. **Bestaand systeem** kan vorm van software bepalen. Beperkte informatietoegang tot wensen **eindgebruiker**.	Integer zakendoen is een belangrijke norm voor de **leidinggevende**.
Acties individuen	**Leidinggevende** bepaalt vorm software door klantgesprek en kennis over werking software. **Programmeur** werkt vanuit functionele eisen en niet direct uit wensen van de klant. **Klant** heeft een relatief simpele probleemstelling.	**Leidinggevende** bepaalt vorm van software maar wordt beperkt door functioneel ontwerp, beschikbare middelen en standaard systeem. **Programmeur** geeft technische gevaren / moeilijkheden aan maar volgt in principe functioneel ontwerp. **Eindgebruiker** alleen invloed via eindtest en via opdrachtgever	**Leidinggevenden** werken naar een goede oplossing voor de klant.

Tabel 5.1 Het algemene softwareontwikkelingsproces

Aan het slot van dit deel van de resultaten is duidelijk geworden dat er meer bij het proces van ontwikkeling gedaan wordt dan het verwerken van de wensen van de klant. Het bovenstaande geeft de processen weer die een rol spelen bij aanpassing van software op de klantorganisatie. De meeste resultaten staan in een breder kader, ze zijn daardoor echter niet minder belangrijk. Ze kunnen namelijk een antwoord geven op de vraag waarom software wel of niet op een bepaalde wijze wordt aangepast aan de klant.

In de conclusie wordt verder ingegaan op de consequenties van deze resultaten, eerst zal echter in de volgende paragrafen besproken worden hoe autonomie in de software geprogrammeerd wordt.

5.3 Software en autonomie

Ook de ontworpen autonomie is volgens de structuratietheorie geanalyseerd om het tweede deel van de theorie te onderzoeken. De resultaten zullen om die reden ook gepresenteerd worden naar de drie modaliteiten van deze theorie. Een aantal van de resultaten zullen overlappen met de resultaten uit de vorige paragrafen, om die reden zullen deze wat bondiger besproken worden. Waar de resultaten specifiek voor de autonomie gelden zullen ze uitvoeriger besproken worden.

5.3.1 Zingeving

Over het algemeen lijkt autonomie geen belangrijk concept in de softwareontwikkeling. Er zijn geen aanwijzingen gevonden dat klanten, leidinggevende of programmeurs zich direct bezighouden met dit concept. Wel is echter gebleken dat automatiseren van bepaalde processen in een organisatie bijna inherent te maken heeft met het beperken van autonomie.

“In een zekere zin wordt het [de werkprocessen] wel aan banden gelegd omdat ons systeem een scherm is met een aantal selectievakjes en invulvakjes en een submit button en nog wat andere tools. En voorheen was dat en een computer met een Excel bestand en wat papierwerk.”

Anders gesteld wordt software gezien als een manier om werkprocessen te formaliseren en het is dus bijna automatisch een beperking van de autonomie.

“Je merk dat ze die [bedrijfsprocessen] aan banden willen leggen zodat er niet van afgeweken kan worden, omdat, wanneer je dat wel kan, er fouten in kunnen sluipen en omdat soms ook de snelheid van de afwikkeling daardoor nadelig beïnvloed wordt.”

Het belangrijkste zingevingsprincipe is hier dus het begrip automatiseringsproces. Het feit dat processen via de computer verlopen betekent beperking van de middelen. Dit wijst erop dat software bijna inherent structuren bevat die autonomie kunnen verminderen.
Hier moet echter opgemerkt worden dat dit mechanisme niet geldt voor de verdeling van de autonomie. Deze keuze naar de invoering van een hiërarchisch systeem wordt met name gemaakt door de leidinggevende. Als hij dit nodig acht zal hij dit adviseren aan de klant.

“Toen hebben we dus aangeven of aanbevolen om zo’n toegangsniveau of beheer in te voeren. Dat dus profielen die bewerkt zijn door gebruiker X gecontroleerd kunnen worden door Y […] Dus dan hebben ze een controlerende functie op elkaar. En uiteindelijk is er een niveau boven geplaatst van iemand die gewoon alles ziet en kan bewerken. En degene die het project heeft opgestart, die heeft de super user rechten gekregen.”

Natuurlijk kan deze wens ook komen vanuit de klant zelf.

“Ik heb nu een klant, die heeft gewoon één projectmanager en daaronder zitten drie gebruikers die leads in kunnen voeren. Maar

die kunnen ook weer niet de gegevens van elkaar zien. Dus eentje doet de webshops, eentje doet bijvoorbeeld de accountants en eentje heeft nog kunsthandels. En dat is echt wel een grote eis van hen, dat de gegevens echt geheim zouden blijven per gebruiker."

Zoals al eerder vermeld zien programmeurs wel de doelen van de software maar hebben zij nog een extra zienswijze. Software is voor hen iets wat technisch goed in elkaar moet zitten. Het moet allemaal goed werken. Ook deze zienswijze leidt indirect tot minder autonomie.

"Een van onze taken is dus de autonomie van gebruikers in te perken want dat is een van de belangrijkste onderdelen van de software die wij maken.
W: Ok, maar waarom is dat beter?
M: Omdat het funest is als iedereen alles maar kan aanpassen. Bijvoorbeeld verkeerde waarden ergens opslaan waardoor dus in principe heel je applicatie kapot kan gaan."

Software die de gebruiker te vrij laat kan als gevolg hebben dat de software niet goed meer werkt, dit is iets wat programmeurs proberen te voorkomen.
Ook hier blijkt dus dat door software te zien als een oplossing voor een probleem de leidinggevende en de klant indirect de autonomie beperken. Niet alleen door het beperken van de mogelijkheden van de gebruikers maar ook door het verdelen van middelen over verschillende gebruikers door een hiërarchisch systeem in te voeren.
Wederom ziet de programmeur de software als een technisch systeem wat goed moet werken en zal ook om die reden de autonomie willen beperken.

5.3.2 Middelen

Zowel de opdrachtgever als de klantorganisatie hebben beperkte middelen tot hun beschikking. Het gaat hier vooral om tijd en geld. Uit de interviews blijkt dat minder autonomie minder kosten met zich meebrengt.

“We proberen gewoon, zeker als er echt maatwerk inzit, het zo op te lossen dat het precies aansluit bij een wens. En daar beperk je mensen natuurlijk in[…]. Je bouwt altijd in een trechtervorm naar een specifieke oplossing. Functioneel moeten we natuurlijk exact kunnen omschrijven wat het wordt omdat we daar ook de uiteindelijke offerte op baseren. […] Kijk we kunnen makkelijk een jaar blijven bouwen voor een klant, dus we moeten het ook wel bekaderen en overal grenzen stellen. Anders weet de klant niet waar ze aan toe zijn en weten wij ook niet waar we aan toe zijn. Dus dan krijgen ze een gigantische factuur , en dan moeten wij zeggen ‘dat komt omdat we niet weten wat we gaan bouwen’.”

Het bovenstaande laat zien dat met name geld en tijd een beperking opleggen aan de vrijheid die een gebruiker van de software kan krijgen. Het kost simpelweg meer tijd en geld om die vrijheid in te bouwen. Een programma dat op één manier één actie mogelijk maakt kost minder dan een programma waarin dit op meerdere manieren kan. Het beperken van de autonomie is dus voordeliger.

Autonomie blijkt wat betreft software niet alleen te spelen op het gebied van de vrijheid van individuele werknemers. Software blijkt vaak een of meerdere soorten gebruikers te hebben. Deze groepen zijn verdeeld naar toegangsniveau, gebruikers met een hogere toegang hebben meer macht in het systeem en dus meer autonomie. Wat betreft deze niveaus is gebleken dat het over het

algemeen duurder is om deze in de software te zetten. In de praktijk lijkt dit echter in mindere mate van invloed te zijn op de uiteindelijke keuze.

W: “Is het duurder om de hiërarchie in te bouwen?”
P: “Wij rekenen daar wel een hogere prijs voor, ja.”
W: “Zeggen klanten dan wel eens van 'Doe het dan maar niet' ”
P: “Nee eigenlijk niet.”

Onder 5.2.1 bleek al dat een programmeur vooral een adviserende functie heeft in het vormgeven van software en dat de adviezen met name technisch zullen zijn. Er zijn geen aanwijzingen gevonden om aan te nemen dat dit andere adviezen zullen zijn dan diegene die komen uit de interacties op de gebieden van zingeving en normen.

5.3.3 Normen

De belangrijkste norm is al eerder verwoord als het functioneel oplossen van een probleem van de klant. Dit is al uitvoerig besproken onder zingeving. Er is echter ook een norm gevonden die het beste te verwoorden is als het goed runnen van het bedrijf. Deze norm vereist duidelijke afspraken over de vorm van de software met de daarbij behorende gevolgen voor de autonomie.

“Het liefst verkoop ik ook zo veel mogelijk en laat ik het helemaal vrij, maar dat gaat niet. Als je dat gaat doen worden het vaak ook chaos projecten dus je moet wel bekaderen. Dus het afkaderen van rechten, je moet dat gewoon doen, anders kom je ook nergens.”

De autonomie wordt hier dus beperkt om goed zaken te kunnen doen. Software kan vele kanten op. Al eerder hebben we gezien dat het inbouwen van vrijheden duurder is, maar nu blijkt ook dat het van belang is voor het bedrijf zelf om de autonomie te beperken.

Grotere autonomie zou wellicht in afspraken vastgelegd kunnen worden, maar de visie van de leidinggevenden lijkt te zijn dat het beperken leidt tot minder problemen.

5.3.4 Samenvatting

Ook dit deel van de resultaten is samengevat in een tabel (5.2). Hierin zijn enkel de bevindingen uit dit hoofdstuk wat betreft autonomie weergegeven. Nu de resultaten van het onderzoek duidelijk zijn kan het volgende hoofdstuk gewijd worden aan formulering van de conclusies op basis van deze resultaten.

	Zingeving	Middelen	Normen
Institutionele context	Software wordt door **klant** en **leidinggevende** gezien als oplossing van een probleem. Software is voor **programmeur** een goed werkend systeem. Verdeling van autonomie is voor **leidinggevende** en **klant** soms van belang	**Klant** en **leidinggevende** bezitten beperkte middelen	**Leidinggevenden** vinden dat grotere ontworpen autonomie complexere afspraken veroorzaakt.
Individuele acties	Autonomie wordt door **leidinggevende** ingeperkt door automatisering. Autonomie wordt door **programmeur** ingeperkt om goed werkende software te maken. Autonomie kan verdeeld worden door hiërarchie in software aan te brengen als **klant** en/of **leidinggevende** dit wenselijk achten.	**Leidinggevende** beperkt autonomie om middelen te besparen. Hiërarchie duurder maar toch toegevoegd	**Leidinggevenden** beperken autonomie om projecten simpel te houden.

Tabel 5.2 Softwareontwikkeling en autonomie

6 Conclusie en discussie

Gezien de bovenstaande resultaten kunnen een aantal conclusies opgesteld worden over autonomie en softwareontwikkeling. Het proces van ontwikkeling blijkt een gecompliceerd geheel met verschillende actoren, verscheidene overwegingen en een complex aan variabelen die in ogenschouw genomen moeten worden. Allereerst zullen een aantal conclusies gepresenteerd worden over het ontwikkelingsproces in het algemeen met het oog op de match tussen software en klant. In het tweede deel van dit hoofdstuk zullen de conclusies wat betreft autonomie gepresenteerd worden. Na het formuleren van een algemene conclusie, op basis van de twee deelgebieden, zal de waarde van dit onderzoek besproken worden.

6.1 Aanpassen van de software op de klant

In de hoofdstukken over de theorie en de methode is het verlangen uitgesproken om een beeld te vormen van de mogelijkheden die softwareontwikkelaars gebruiken om software aan te laten sluiten op de klantorganisatie. Dit alles om te testen hoe goed de theorie bruikbaar is voor het onderzoeken van de relatie tussen software en autonomie. Uit de overwegingen in hoofdstuk drie bleek dat dit niet simpelweg mogelijk was door alleen te kijken naar de ontworpen autonomie zelf. De in dit onderzoek geformuleerde theorie stelt dat autonomie van werknemers ook beïnvloed wordt door de aanpassing van software op de klantorganisatie. Vandaar dat ook dit fenomeen onderzocht is.

De gepresenteerde resultaten laten zien dat er drie actoren zijn die een wezenlijke rol spelen in het proces van softwarevormgeving. De klant, de leidinggevende en de programmeur. Hierin hebben

leidinggevende en klant een wat belangrijkere rol dan de programmeur. Deze heeft een meer adviserende taak.
Het ontwikkelingsproces is wellicht het beste te kenmerken als het stapsgewijs oplossen van een probleem. De software wordt met name door de klant en de leidinggevende gezien als een functionele oplossing die stapje voor stapje wordt opgebouwd. Het feit dat dit de geldende denkwijze zodanig oplossingsgericht is betekent wellicht dat het probleem goed opgelost wordt maar betekent tevens dat er weinig aandacht is voor de aansluiting van de software op de context van het klantbedrijf. Er zijn geen bewijzen gevonden voor het meenemen van zaken die buiten de directe probleemdefinitie vallen.
Dit wordt nog eens extra versterkt door de keuzes die gemaakt worden door de leidinggevende en de adviezen van programmeurs. Door het gebruik van standaardsystemen en door aanwezige kennis, heeft het resulterende systeem een aantal kenmerken die weinig te maken hebben met de klantorganisatie maar veel meer een resultante zijn van de context in het productiebedrijf. Zowel leidinggevende als programmeurs hebben in dit onderzoek aangegeven wel eens een keuze aan de klant aan te bieden omdat dit beter is om technische redenen of dat het beter past bij het standaard systeem.
De persoon die namens de klant bij het vormen van de software betrokken wordt, vervult bijna altijd een hogere functie in het klantbedrijf en is meestal geen goede representatie van de gemiddelde eindgebruiker van het systeem. De uiteindelijke eindgebruiker wordt sporadisch bij het ontwikkelingsproces betrokken als de software eigenlijk al voltooid is. De invloed die dan nog mogelijk is, is klein en verloopt zelfs dan nog via de opdrachtgever. Zelfs als zij hier zaken als problemen met de aansluiting tussen de werkprocessen en de software melden, is het maar de vraag of deze inderdaad worden meegenomen. Wellicht

heeft de symbolische waarde van deze inspraak echter een klein positief effect op de acceptatie van de software.
Hier moet wel aangegeven worden dat de normen binnen de softwareorganisatie niet de mogelijkheid uitsluiten om de klant zo goed mogelijk te helpen. Er is absoluut geen onwil om niet zo goed mogelijk naar de klant te luisteren. De beste beschrijving van de situatie is echter dat de norm niet verder gaat dan een technisch goede oplossing aan bedrijven leveren. De normen van het bedrijf streven er niet naar op deze oplossing nog eens aan te laten sluiten op de institutionele context van de klant.

- Wat gebeurt er tijdens softwareproductie om de software goed te laten passen bij de klant?

Uit het bovenstaande kunnen we concluderen dat het antwoord op deze vraag luidt dat er nagenoeg niets gebeurt op dit gebied. De wensen van de klant zijn problemen waar een technische oplossing voor gevonden moet worden. Er worden geen pogingen ondernomen door de klant of door de software producerende organisatie om de institutionele context van de klant te ontdekken. De meningen van individuen aan de onderkant van de organisatie worden slecht sporadisch en zijdelings geïnventariseerd en zelfs dan is het de vraag of deze meningen effect hebben op de vorm van de software.
De structuratietheorie geeft ook een mogelijke verklaring voor dit gebrek. Er is weliswaar de norm dat men de klant zo goed mogelijk wil helpen maar deze staat in conflict met de interpretatie dat de software een oplossing is voor een probleem, hierdoor wordt de functie van software belangrijker dan het helpen van de klant. Tevens wordt de softwarevorm bepaald door de leidinggevende die beperkt wordt in de beslissingsruimte door tijd, geld en een bestaand systeem. Dit laat weinig ruimte over voor het aanpassen van de software op de klant. Hier komt nog eens bij dat de

programmeur software interpreteert als een technisch goede oplossing. Ook dit laat weinig ruimte over voor aanpassing op de klant.
Deze factoren samen maken dat er in de institutionele context van softwareproductie alleen ruimte voor aanpassing is op het gebied van normen. Zowel beschikbare middelen als interpretatieve schema's spreken deze mogelijkheid tegen. De conclusie dat er weinig aanpassing is betekent dat de ontworpen autonomie waarschijnlijk weinig zal lijken op de werkelijke autonomie.

6.2 Ontworpen autonomie en software

Uit dit onderzoek is gebleken dat autonomie bijna inherent beperkt wordt door software. Er zijn een aantal zaken die dit proces veroorzaken.
Ten eerste is uit de vorige paragrafen gebleken dat software vooral gezien wordt als de technische oplossing van een bedrijfskundig probleem. De aard van deze problemen en de gekozen oplossingen betekenen bijna altijd het formaliseren van werkprocessen. Dit vastleggen van arbeid in technische regels kan weinig anders betekenen dan het beperken van de autonomie. De middelen die werknemers tot hun beschikking hebben, worden als het ware voor deze groep gekozen door het ontwerp van een softwaresysteem. Dit is niet alleen een wens van de klant, maar ook leidinggevenden zien dit proces op deze manier en handelen hiernaar.
Een tweede soortgelijke redenatie gaat op voor de manier waarop programmeurs software zien en in elkaar zetten. Voor hun is de technische kant veel belangrijker en maken ze graag software waar weinig mee mis kan gaan. De makkelijkste manier om dit te bereiken is het beperken van de autonomie. Door opties beperkt te houden wordt de informatiestroom binnen de software voorspelbaar. Dit leidt tot minder fouten en maakt het de software

technisch robuuster. Het beperken van autonomie is dus ook een technische voorwaarde.

Een derde overweging is de hiërarchie binnen de software. Dit raakt autonomie omdat het gaat om de verdeling van middelen. Het aanbrengen van een hiërarchie in het softwaresysteem betekent dus een beperking van de autonomie voor in ieder geval sommige gebruikers. Uit de interviews is gebleken dat hiërarchie soms door de klant zelf gevraagd wordt en soms door de leidinggevende wordt aangeraden. Er zijn echter ook gevallen waarbij geen van beide gevallen zich voordoen. Ondanks het feit dat dit duurder is, wordt er in ieder geval geregeld een verdeling van middelen in de software aangebracht. Dit zou gezien kunnen worden als een beperking van de autonomie.

Tenslotte is er een norm onder leidinggevenden om projecten niet onnodig complex te maken. Het vergroten van ontworpen autonomie betekent meer afspraken die het project ingewikkelder maken. Dit om problemen achteraf te voorkomen. Dit is voor de leidinggevenden een reden om de ontworpen autonomie beperkt te houden.

Uit dit bovenstaande kan opgemaakt worden dat interactie tussen softwareontwikkelaars en de context waarin zij zich bevinden acties oplevert die de ontworpen autonomie beperken. Dit is geenszins een bewust proces maar juist een proces wat voortvloeit uit de omgeving die softwareontwikkelaars hebben opgebouwd en de brede omgeving van het bedrijfsleven waar ze deel van zijn. De autonomie wordt beperkt door zaken als techniek, beperkte middelen, concepties over probleemoplossing en dergelijke. In het volgende paragrafen worden deze bevindingen gecombineerd met de consequenties voor autonomie.

6.3 Conclusie

Het doel van dit onderzoek is het beantwoorden van de vraag welke rol softwareproductie speelt in de vorming van de autonomie van eindgebruikers. Reeds in het theoretisch kader bleek dat dit proces een complex geheel is. Het is om die reden dat de keuze is gemaakt om niet aan de hand van een bestaande theorie de werkelijkheid te onderzoeken. De nieuwe theorie is gevormd uit een aantal bouwstenen die met name ontleend zijn aan de structuratietheorie.
Het hoofdobject van de theorie is de productie van software als sociaal proces. We zijn ervan uitgegaan dat software tot stand komt door de interactie van verschillende actoren met de institutionele context van de softwareorganisatie. Tevens is een verfijning aangebracht in de invloed van software op autonomie. Naast de autonomie die in de software besloten ligt is ook de de aanpassing van software op de klantorganisatie van invloed.
De vraag is nu in hoeverre dit conceptuele kader bruikbaar is voor het onderzoeken van softwareproductie in het kader van autonomie. De methode van de drie modaliteiten lijkt goed bruikbaar voor het analyseren van de softwareproductie, ook als het om onderdelen van het uiteindelijke product gaat.
Wat betreft het begrip ontworpen autonomie is gebleken dat dit bijna inherent beperkt wordt door de softwareproductie. Een aantal factoren spelen hier een rol. Software wordt door de klant en de leidinggevende gezien als een automatisering van een werkproces. De programmeur ziet software als een technisch geheel waarin autonomie gevaarlijk is. Ten slotte is autonomie duurder om te maken. Het feit dat verdeling van autonomie (hiërarchie) duurder is om te maken lijkt te wijzen op een nivellerende werking maar als de leidinggevende dit adviseert wordt het toch vaak ingevoerd. We kunnen dit als volgt samenvatten:

- Interpretatie als automatisering → minder autonomie

- Interpretatie dat autonomie fouten kan veroorzaken →minder autonomie
- Autonomie duurder om te maken →minder autonomie
- Hiërarchie duurder om te maken → meer autonomie
- Autonomie maakt projecten complexer → minder autonomie

Helaas kunnen we het bovenstaande niet interpreteren als een optelsom maar het laat in ieder geval zien dat het analyseren van softwareproductie op deze wijze een nuttig inzicht kan geven in de effecten op autonomie.

Het onderzoek ging ook om de aanpassing van software op de klantorganisatie. Hier blijkt de norm weliswaar om de klant zo goed mogelijk te helpen, maar dit wordt beperkt door het feit dat alleen het probleem dat de software moet oplossen bekeken wordt. Tevens bezit de softwareproducent beperkte middelen waardoor aanpassing niet eens opkomt als mogelijkheid. De machtsstructuur maakt dat de programmeur enkel werkt vanuit een door de leidinggevende bedacht functioneel ontwerp en dat de eindgebruiker bijna geen invloed heeft. Kort samengevat komt dit neer op de volgende verbanden:

- Interpretatie van software als oplossing voor een probleem → minder aanpassing op klant
- Beperkte middelen → minder aanpassing op klant
- Kleine macht eindgebruiker → kleine aanpassing op klant
- Programmeur krijgt alleen functioneel ontwerp → minder aanpassing op klant.
- Norm van klanten helpen → (mogelijk) meer aanpassing op klant

Wederom kan het bovenstaande niet gezien worden als optelsom. Tevens zijn de relaties minder sterk en eenduidig maar het bovenstaande laat wel zien hoe een norm die de goede intenties

propageert naar de achtergrond wordt geschoven door een context waarin andere zaken als belangrijker gezien worden.

Beide zijden van de theorie zijn nu in kaart gebracht en blijken goed analyseerbaar door middel van de structuratietheorie. We zien dat er bij de onderzochte organisaties bepaalde factoren aanwezig zijn die de autonomie met name verkleinen en die minimaal hun product aanpassen op de klantorganisatie.
De theorie is opgesplitst in twee delen omdat onderzoek naar de ontworpen autonomie en de uitwerking op de werkelijke autonomie tegenstrijdige resultaten opleverde. Nu kunnen we zien dat dit mogelijk aan de hand van deze theorie te verklaren is. De software mag dan in de basis autonomie beperkend werken, uit dit onderzoek blijkt ook dat de software bijna niet wordt aangepast aan de klantorganisatie. We kunnen geen uitspraken doen over de werkelijkheid, maar wel kan men verwachten dat een stuk software dat niet past bij de klantorganisatie niet goed geaccepteerd zal worden en dus de autonomie niet naar verwachting zal beïnvloeden.
Dit bovenstaande is een lastige denkstap maar goed te begrijpen als we nogmaals kijken naar de structuratietheorie. Deze theorie stelt dat technologie gevormd wordt door de manier waarop gebruikers deze zien. Ze kunnen ervoor kiezen de software anders te gebruiken dan bedoeld, vooral als deze niet past bij hun institutionele context. Als ze de technologie niet gebruiken zoals het hoort kan dit betekenen dat ook de autonomie niet zal veranderen zoals men zou verwachten als op basis van de ontworpen autonomie. Dit is de reden dat aanpassing van de software op de klantorganisatie is meegenomen in de theorie.
Uit het bovenstaande blijkt dat de theorie die in dit onderzoek geformuleerd is goed in staat is te verhelderen welke rol softwareproductie speelt in de vorming van autonomie van werknemers. De structuratietheorie biedt goede mogelijkheden het

softwareproductieproces in kaart te brengen en de scheiding van ontworpen autonomie en aanpassing kan mogelijk goed verklaren waarom ontworpen autonomie niet altijd het verwachte effect op de autonomie van de werknemer heeft.

6.4 Discussie

Een aantal opmerkingen moeten nog geplaatst worden om deze studie in het juiste licht te kunnen bekijken. Allereerst moet opgemerkt worden dat dit een exploratieve studie is en dat deze ook zo geïnterpreteerd moet worden. Terwijl er veel onderzoek is geweest naar software en kwaliteit van arbeid zijn er weinig studies geweest die zich gericht hebben op het ontwikkelingsproces binnen het softwarebedrijf en de gevolgen voor autonomie. Dit unieke perspectief is wat mij betreft meer onderzoek waardig zeker gezien de bevindingen.

Ten tweede is dit onderzoek op een aantal fronten bewust beperkt gehouden gezien de beschikbare middelen. Het moge duidelijk zijn dat een relatief kleine steekproef is genomen. Verder is het onderzoek beperkt gebleven tot een bepaalde sector van de softwareontwikkelingbranche. Er is in deze studie met name gekeken naar CMS systemen terwijl het zeer aannemelijk is dat autonomie een veel grotere rol speelt bij bijvoorbeeld ERP[5] systemen of callcenter software. De respons hierop kan er simpelweg maar één zijn. Er is inderdaad meer onderzoek nodig met meer subjecten op meer gebieden om duidelijkere conclusies te kunnen formuleren. Ik hoop ook dat dit zal plaatsvinden.

Ten slotte is er een probleem wat inherent is aan elk kwalitatief onderzoek bij organisaties, zodra er een vraag gesteld wordt over hoe een bedrijf functioneert bestaat het gevaar dat de werknemer

[5] Electronic Resource Planning, een system om via de software de flow van producten in een organisatie te regelen.

zal antwoorden naar het ideale bedrijfsmodel wat meer een streven is dan de werkelijkheid. Ik heb geprobeerd dit zoveel mogelijk te vermijden door vragen verschillend te formuleren en zoveel mogelijk vervolgvragen te stellen. Ook heb ik respondenten aangemoedigd om veel voorbeelden te geven. Ik hoop ook dat het geholpen heeft dat ik bij veel bedrijven twee mensen heb geïnterviewd waardoor ik twee visies van de werkelijkheid heb gekregen.

De gevonden theorie lijkt goed bruikbaar voor vervolgonderzoek, maar is nog maar een eerste start. De waarde van deze theorie zal pas echt blijken als onderzocht wordt hoe de aanpassing van software op de klant van werkelijke invloed is op de autonomie. De structuratietheorie van technologie stelt vast dat dit van belang is, maar de werkelijke situatie moet nog aan de werkelijkheid getoetst worden.

7 Bronvermelding

S.R. Barley, (1986) *Technology as an Occasion for Structuring: Evidence from Observation of CT Scanners*. Administrative Science Quarterly Vol. 31, pp 78-108

Ronald Batenburg, Jos Benders en Wim Scheper (2002) *Over 'Groene Weides' en 'Blauwdrukken': Enterprise Resource Planning in een groot nutsbedrijf.* In Ronald Batenburg, Jos Benders, Nick van den Heuvel, Peter Leisink, Jeroen Onstenk (red.) Arbeid en ICT in Onderzoek (pp 109-121). Utrecht: Uigeverij LEMMA BV.

L. Brooks (1997) *Structuration theory and new technology: analysing organizationally situated computer-aided design*. Information Systems Journal, Vol. 7, pp 133-151

Damborg Frederiksen H, Rose J (2003) *The social construction of the software operation: reinforcing effects in metrics programs,* Scandinavian Journal of Information Systems, Vol 15, pp 23-37

Barbara Garsons (1984) *The Electronic Sweatshop: How Computers Are Transforming the Office of the Future into the Factory of the Past,* New York: Simon & Schuster.

Anthony Giddens (1986) *The Contitution of Society,* University of California Press, Reprinted edition.

Joseph A. Maxwell (1996) *Qualitative Research Design: An interactive approach.* Londen: Sage Publications Inc.

Orlikowski, W.J., Robey, D. (1991) *Information technology and the structuring of organizations.* Information Systems Research, Vol. 2 No.2, pp.143-69

Hans D. Pruijt (1996) *The Fight Against Taylorism in Europe: Strategies, Achievements in Job Design and Technology, Setbacks, Obstacles, Changes for Upgrading Work*, Proeftschrift Erasmus Universiteit Rotterdam.

J. Rose, P. Lewis (2001) In: N.L. Russo, B. Fitzgerald and J.D. Gross (red.) *Using structuration theory in action research: An intranet development project, in realigning research in practice in information systems development: The social and organizational perspective*, Kluwer Academic Publishers, Boston (2001), pp. 273–295.

Joris van Ruysseveldt, Marco de Witte, Jasper von Grumbkow (red.) (1998), *Organiseren van mens en arbeid: Hedendaagse benaderingen van de kwaliteit van arbeid*. Heerlen: Kluwer Bedrijfsinformatie

Shoshana Zuboff (1988) *In the Age of the Smart Machine,* Baisc Books, Inc, Verenigde Staten.