De Softwarefactory

Proces, professional en performance

Scriptie Master Software Engineering

ing. Ernout van der Waard

15 augustus 2007

Opleiding:	Master Software Engineering
	Universiteit van Amsterdam
Afstudeerdocent:	Drs. Hans Dekkers
Opdrachtgever:	BedrijfX
Publicatiestatus:	Openbaar

Voorwoord

Deze scriptie beschrijft het afstudeeronderzoek van Ernout van der Waard, als afsluiting van de Master Software Engineering aan de Universiteit van Amsterdam. Het onderzoek vond plaats bij BedrijfX. BedrijfX is een internationale ICT-dienstverlener. Zij heeft ongeveer 40.000 medewerkers in dienst en heeft kantoren in 41 landen. Het bedrijf is actief op het gebied van business consultancy, systeemintegratie en IT- en business process outsourcing. Het afstudeeronderzoek heeft plaatsgevonden bij de divisie Public Sector in Rijswijk, in één van de Product Centres van BedrijfX. Het Product Centre is het softwarefactory concept van BedrijfX.

Dit onderzoek is ontstaan vanuit mijn persoonlijke interesse naar hoe een softwareproces het beste georganiseerd en geoptimaliseerd kan worden. Met name de combinatie professional en proces interesseert mij, omdat die combinatie naar mijn inzicht het succes van grote projecten maakt of breekt. Het Product Centre was een uitstekende plaats om dit onderzoek te doen, omdat het softwareproces in een softwarefactory een belangrijke plaats in neemt. De wens voor voorspelbaarheid en controle in een softwarefactory brengt interessante vraagstukken met zich mee.

Dankwoord

Bij deze bedank ik mijn afstudeerbegeleider Hans Dekkers voor zijn enthousiasme, inspiratie en nuttige feedback. Daarnaast bedank ik Ron Mooij en Reza Ahmadi voor het beschikbaar stellen van de onderzoeksplaats bij BedrijfX en voor de ondersteuning tijdens het onderzoek. De projectmanagers en developers van het Product Centre ben ik dankbaar voor het meewerken aan het onderzoek en Richard van Haaren en Ilske Verburg voor de hulp tijdens het onderzoek. Ten slotte bedank ik mijn vriendin Jacqueline voor de support en begrip voor alle avonden die ik achter de computer doorbracht.

Ernout van der Waard

Samenvatting

Het ontwikkelen van software moet voorspelbaar, goedkoper, sneller en met hogere kwaliteit. Met deze doelstelling als uitgangspunt wordt er gewerkt aan het industrialiseren van de softwareontwikkeling, en ontstaan er softwarefactory's. Het Product Centre bij BedrijfX is zo een softwarefactory. Een softwarefactory is echter niet per definitie doelmatig. De hoofdvraag van dit onderzoek is: Welke factoren beïnvloeden de doelmatigheid van het softwareproces in een softwarefactory? Inzicht in deze factoren helpt BedrijfX bij het optimaliseren van de software-factory.

Het eerste gedeelte van het onderzoek bestaat uit een literatuuronderzoek. Verschillende invalshoeken op het bestuderen van het softwareproces zijn bestudeerd. Ook is gekeken naar de complexiteit van het invoeren van een softwareproces en hoe processen beoordeeld kunnen worden.
Vervolgens is het softwareproces in het Product Centre bestudeerd. Er is gekozen voor een kwalitatieve en exploratieve aanpak, waarbij vanuit verschillende invalshoeken metingen en observaties gedaan worden. De combinatie van alle waarnemingen geeft het benodigde inzicht om de onderzoeksvragen te kunnen beantwoorden.

Bij BedrijfX is als eerste gekeken naar de mate waarin het Product Centre een softwarefactory is. Vervolgens is het softwareproces van het Product Centre vergeleken met het softwareproces in een organisatie zonder softwarefactory. Door langere tijd in de factory aanwezig te zijn, gesprekken te voeren met professionals en professionals hun werkwijze te laten vastleggen, ontstond er inzicht in het werkelijke softwareproces van het Product Centre. Daarnaast is gekeken naar de effecten van dit softwareproces op de performance en de professionals in de softwarefactory. Er is onderzocht of een softwarefactory altijd beter presteert dan een organisatie zonder softwarefactory. Hiervoor is een kwaliteits- en efficiëntiemeting gedaan bij twee developmentteams, één team van het Product Centre en één team zonder softwarefactory. Het effect van het softwareproces op de professionals in de softwarefactory is onderzocht met een gestructureerde vragenlijst en gesprekken met professionals in de factory.

De resultaten van de verschillende observaties, gesprekken en metingen verschaffen inzicht in de complexiteit van het softwareproces in een softwarefactory. Zichtbaar is geworden dat evolutie van factoryonderdelen de voordelen van de factoryaanpak kan beperken, en dat het kiezen van de scope van de softwarefactory de winst zowel positief als negatief beïnvloed. Er zijn argumenten gevonden waarom de ultieme softwarefactory (met een hele hoge standaardisatie en een hoge afstemming tussen onderlinge factoryonderdelen) niet per definitie een succes hoeft te zijn. Een hoge standaardisatie vraagt investeringen en is onderhevig aan evolutie. Een hoge afstemming tussen de verschillende componenten van de softwarefactory is moeilijk houdbaar en evolutie krijgt een grotere impact.

Inhoudsopgave

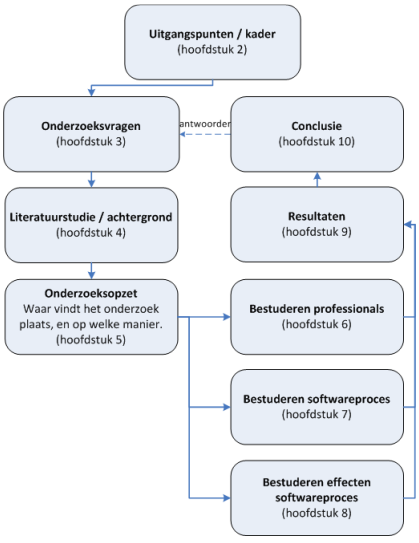
1 Inleiding

Het ontwikkelen van software moet goedkoper, sneller en met hogere kwaliteit. Met deze doelstelling als uitgangspunt wordt er gewerkt aan het industrialiseren van de softwareontwikkeling en ontstaan er softwarefactory's. Een softwarefactory kenmerkt zich globaal gezien door hergebruik van een standaard ontwikkelomgeving, hergebruik van eerder gemaakte componenten en een strikte standaardisatie en definitie van het softwareproces. Het Product Centre bij BedrijfX is zo een softwarefactory.

Softwarebedrijven die bezig zijn met het opzetten van een softwarefactory moeten hun werkprocessen hierop aanpassen. Dit is nodig om de gewenste mate van hergebruik en standaardisatie te bereiken. Het aanpassen van deze processen levert binnen deze bedrijven een aantal interessante vragen en discussies op. Er moeten bijvoorbeeld afspraken worden gemaakt over hoe de processen gaan veranderen, en hoe ervoor gezorgd wordt dat deze wijzigingen de gestelde doelen helpen bereiken. Daarbij is het zo dat bij deze proceswijzigingen negatieve effecten niet zijn uitgesloten. Veel bedrijven hebben teams die goed functioneerden vóór de komst van de factory. Men moet er zeker van kunnen zijn dat de factoryontwikkeling geen negatieve invloed heeft op de prestaties van deze teams.

1.1 Indeling van de scriptie

Hoofdstuk twee beschrijft het kader van het onderzoek. Hierin staan de uitgangspunten beschreven van het onderzoek. Vervolgens bevat hoofdstuk drie de probleemstelling, waarbij de onderzoeksvragen zijn opgenomen. In hoofdstuk vier zijn invalshoeken beschreven op het software proces. Vervolgens worden in hoofdstuk vijf tot en met acht de verschillende thema´s van het onderzoek uitgewerkt. Per stap worden de resultaten gegeven. In hoofdstuk negen zijn de resultaten gebundeld en in hoofdstuk 10 volgt de conclusie. Ten slotte zijn in hoofdstuk elf de inzichten opgenomen die tijdens het onderzoek (los van de resultaten) zijn opgedaan.



Figuur 1: Structuur de scriptie

2 Kader

Het werken met een klein, scherp en optimaal op elkaar afgestemd team van professionals klinkt voor veel softwareprofessionals ideaal (Brooks, 1995). Het opbouwen en opleiden van een dergelijk team vraagt de nodige investering, maar daarna kunnen softwareprojecten zonder problemen met een optimale kwaliteit en binnen planning worden opgeleverd. Vanwege de beperkte grootte van het team blijft organisatie en papierwerk beperkt en kan iedereen zich concentreren op de kern: goede kwaliteit software ontwikkelen.

Hoewel een dergelijk optimaal team zeker een optie is, zal het niet lang in deze vorm bestaan. Een goed team krijgt vanuit de huidige markt namelijk meer vraag dan het team kan bieden waardoor het team zal groeien. Daarnaast is het zo dat softwaresystemen steeds groter en complexer worden, waardoor één klein team niet voldoende is en er sprake is van meerdere grotere teams die aan één product werken (Cusumano, 1991). Deze nieuwe omgeving met grotere teams en complexere softwareprojecten kent nieuwe uitdagingen. Het is in deze grotere omgeving bijvoorbeeld niet langer haalbaar om alleen met excellente, ervaren professionals te werken. Het is goed voor te stellen dat er ook veel beginnende ontwikkelaars worden aangetrokken om teams voor een groot project te bemannen. En een groter team betekent meer mensen die moeten weten wat ze moeten doen, wanneer, en op welke wijze.

Kortom: deze ontwikkeling stelt eisen aan de inrichting van het softwareproces. Eén klein team kan wellicht nog functioneren zonder teveel aandacht voor organisatie en softwareproces, maar voor meerdere teams die werken aan grotere producten lijkt aandacht voor het softwareproces cruciaal.

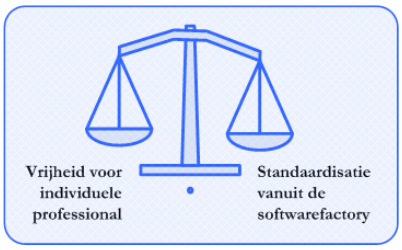
Dit onderzoek is opgezet vanuit de veronderstelling dat een optimaal ingeregeld softwareproces essentieel is voor moderne professionele softwareondernemingen, en dat de professionals in het softwarebedrijf hierin erg belangrijk en misschien wel onderbelicht zijn. Als uitgangspunt is gehanteerd dat een ideaal softwareproces naast aandacht voor het project en de activiteiten die moeten worden uitgevoerd, bovenal afgestemd moet zijn op de mensen die het werk doen. Het softwareproces dat een professional nodig heeft om optimaal te presteren, verschilt wellicht per professional. Om deze afstemming tussen professionals en proces te verbeteren is het van belang in detail naar het proces te kijken en te zien hoe verschillende professionals projecten aanpakken en tegen wat voor soorten problemen zij aanlopen. Hiermee kan de vraag worden beantwoord of verschillende professionals verschillende optimale softwareprocessen hebben, of dat er iets bestaat als één optimaal softwareproces dat voor iedereen gelijk is.

Het antwoord op deze vraag bevindt zich op het raakvlak van het Sofware Engineering vakgebied en de Sociale Wetenschappen. Het aantal onderzoeken dat op dit gebied wordt uitgevoerd is beperkt. Eén van de onderzoeken op dit gebied is van Acuña en Juristo (2003). Zij proberen de rollen die mensen tijdens het softwareproces innemen af te stemmen op de persoonlijkheidskenmerken (Acuña, Juristo, & Moreno, Modelling Human Competencies in the Software Process, 2003). Acuña en Juristo behalen goede resultaten in een experiment waarbij de rollen die mensen innemen binnen een softwareproces zijn afgestemd op hun capaciteiten (die zijn afgeleid van hun persoonlijkheidskenmerken). Bij het

experiment wordt een vermindering in fouten met 47% bereikt, en een productiviteitswinst van 44% bereikt met het afgestemde proces ten opzichte van de standaard rolverdeling. Voor elke rol hebben zij aangegeven welke capaciteiten (zoals decision making, judgement, discipline, empathy) er nodig of juist niet nodig zijn om een rol optimaal te vervullen. Dit is een voorbeeld van een experiment dat laat zien dat een proces dat aansluit op de softwareprofessionals tot goede resultaten leidt. Kennis van persoonlijkheidskenmerken van teamleden kan op nog een andere manier nuttig zijn. Met deze kennis zou men kunnen zorgen dat teamleden elkaars sterke en zwakke punten kennen. Hierdoor kan men vervolgens beter samenwerken (Rutherfoord, 2001).

Een onderzoek op basis van rollen zou ook kunnen worden opgezet op het niveau van taken. Dit is interessant voor de optimalisatie van bestaande teams met een bestaande rolverdeling. Hier is het niet meer mogelijk om de rollen af te stemmen op de personen in het team. Met de resultaten van een dergelijk onderzoek kan het softwareproces ook in bestaande teams op de softwareprofessional worden afgestemd. Het proces geeft de professional de nodige sturing op de juiste plaatsen, door bijvoorbeeld bepaalde documentatie of processtappen af te dwingen. Op plaatsen waar men verwacht dat de professional beter presteert met meer vrijheid, kan deze vrijheid worden ingebouwd.

Dit onderzoek wordt uitgevoerd vanuit het perspectief van een softwarefactory, het Product Centre van BedrijfX. Zoals in de inleiding beschreven, zijn softwarefactory's juist op zoek naar standaardisatie en generalisatie, in plaats van naar de differentiatie die in de voorgaande paragrafen is bepleit. Ook in de softwarefactory zal men beslissingen moeten nemen over op welke plaatsen het gestandaardiseerde proces van de factory sturing geeft, en waar meer vrijheid zit.



Figuur 2

In dit hoofdstuk is aangegeven waarom het softwareproces belangrijk is en dat het evenwicht tussen één gestandaardiseerd softwareproces en een op de professional afgestemd softwareproces een interessant spanningsveld is. Om het onderzoek ook daadwerkelijk uitvoerbaar te maken in de beschikbare drie maanden, zijn er keuzes gemaakt. Deze zijn terug te vinden in de opdrachtomschrijving in het volgende hoofdstuk.

3 Probleemstelling

In het vorige hoofdstuk is beschreven wat de uitgangspunten en gedachten waren voor het opstarten van het onderzoek. Deze ideeën zijn in dit hoofdstuk uitgewerkt tot concrete vragen en doelstellingen voor het onderzoek.

3.1 Introductie tot het probleem

Het Product Centre is de softwarefactory van BedrijfX. De missie van het Product Centre is het ontwikkelen van kwalitatief goede software, op een voorspelbare manier, sneller en met lagere kosten. Men probeert dit te bereiken door een gestandaardiseerde aanpak, met aandacht voor hergebruik en gebruik van BedrijfX´s offshore mogelijkheden.

Zoals in de inleiding van dit rapport al kort werd beschreven heeft deze ontwikkeling impact op de inrichting van het softwareproces. De factoryaanpak vereist wijzigingen in het softwareproces om de gewenste mate van hergebruik en standaardisatie te bereiken.

3.2 Doel van het onderzoek

Omdat het onderzoek in een bedrijfsomgeving wordt uitgevoerd, zijn hier uiteindelijk prestaties en daarmee winst de belangrijkste doelstellingen. Hoewel de rol van de professional in relatie tot het softwareproces in het onderzoek onderzocht zal worden, zal het geheel worden geplaatst in de context van de doelmatigheid van de softwarefactory. Door het factoryproces te bestuderen en te kijken naar de factoren die het proces verbeteren of verstoren, kan BedrijfX de prestaties van het factoryproces binnen het Product Centre verbeteren. Het onderzoek zal een overzicht opleveren van de factoren die de doelmatigheid beïnvloeden. Deze factoren kan BedrijfX gebruiken als meetlat waartegen veranderingen van het softwareproces kunnen worden gelegd:

- bij proceswijzigingen en optimalisaties kan men inspelen op de mogelijke effecten van deze wijziging
- beslissingen op dit gebied onderbouwen/funderen: waarom is het factory proces zoals het is?

3.3 Onderzoeksvragen

3.3.1 Hoofdvraag

> Welke factoren beïnvloeden de doelmatigheid van het softwareproces in een softwarefactory?
>
> Bedrijven hebben bepaalde doelen met het inrichten van het softwareproces als softwarefactory. Welke factoren zorgen ervoor dat deze doelstellingen op een efficiënte manier gehaald worden, en welke factoren maken het factoryproces juist suboptimaal?

Deze hoofdvraag is onderzocht door het doen van waarnemingen bij het Product Centre van BedrijfX. De resultaten van deze waarnemingen zijn daardoor representatief voor het Product Centre.
Het is echter de bedoeling om in het concluderende hoofdstuk te abstraheren van deze metingen en tot een algemene reeks factoren te komen, die ook bij andere softwarefactory's van invloed zijn.

3.3.2 Subvragen

a) In welke mate is het Product Centre een softwarefactory?

Tijdens het bestuderen van het proces in het Product Centre, werd een beperkt bureaucratisch systeem zichtbaar. Er was minder zichtbaar van de standaardisatie in de softwarefactory dan vooraf verwacht. Daarom is er besloten te kijken naar de verschillen tussen het proces bij het Product Centre en het proces in een organisatie buiten de softwarefactory. De verschillen die zichtbaar worden geven meer inzicht in de invloed van het softwarefactoryconcept op het softwareproces.

b) Zijn er duidelijke verschillen zichtbaar tussen een factory en een niet factory software proces?

Een van de uitgangspunten van het onderzoek is dat de professionals een cruciale rol innemen in softwareproductie (zie hoofdstuk 2). Het proces wordt beïnvloed door de professionals en de professionals worden ook beïnvloed door het proces. Daarnaast zijn er grote verschillen in productiviteit tussen individuele professionals (Brooks, 1995). In paragraaf 6.2 zal de 'hele goede professional' worden gedefinieerd en meetbaar gemaakt.

c) Hoe ervaren hele goede professionals en gemiddelde professionals het softwareproces binnen het Product Centre?

Het factoryconcept van het Product Centre is ingevoerd om bepaalde doelstellingen te behalen. Het factoryconcept moet zowel de kwaliteit als efficiëntie verhogen. In dit onderzoek wordt, naast naar het proces en de professional ook naar de performance gekeken.

d) Is de efficiëntie en kwaliteit van een software factory altijd hoger dan een niet factory?

De hoofdvraag die in de eerste paragraaf is geformuleerd kan vanuit verschillende perspectieven benaderd worden. In dit onderzoek wordt de procesgang waargenomen, de prestaties hiervan en de effecten op de professionals in de factory. De combinatie van deze antwoorden helpt het beantwoorden van de hoofdvraag.

4 Invalshoeken op het bestuderen van het softwareproces

'Welke factoren beïnvloeden de doelmatigheid van het proces in een softwarefactory'. In deze hoofdvraag zit een aantal aspecten die nadere specificatie vragen:

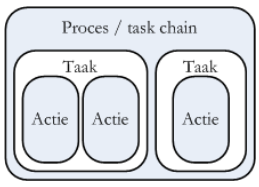
1. Proces
   Wat verstaat men onder de woorden proces en softwareproces? Hoe kan een proces zichtbaar en bespreekbaar worden gemaakt?
2. Doelmatigheid
   Welke doelen heeft men voor ogen bij het inrichten van een (factory) proces? Welke invalshoeken zijn er op het beoordelen, verbeteren en meten van processen?

Deze vragen worden in dit hoofdstuk besproken op basis van de volgende vier invalshoeken op het bestuderen van het proces: proces definitie, proces implementatie, proces verbetering en proces meting en beoordeling.

4.1 Proces definitie

4.1.1 Definitie van het woord proces

Er zijn meerdere definities voor het woord proces. In het algemeen kan gezegd worden dat processen met name gericht zijn op *hoe* men werk aanpakt in plaats van *wat* de resultaten zijn. De Van Dale (2005) definieert proces als 'geleidelijke verandering'. Uit het woord 'geleidelijk' zou men kunnen opmaken dat een proces over het algemeen uit een aantal veranderingen bestaat die samen 'het proces' vormen.



Figuur 3

Iets verder gespecificeerd kan men het proces zien als meerdere, onderling verbonden ketens van taken (Scacchi, 2001). Een keten van taken (taskchain) definieert Scacchi als een niet-lineaire reeks taken die objecten structureren en omzetten in tussen- of eindproducten. In deze definitie is ook de geleidelijkheid (een reeks taken) en verandering (structureren en omzetten van objecten) terug te vinden, die in de definitie van de Van Dale terug kwamen. Van belang is volgens Scacchi dat deze taken niet altijd deterministisch zijn. Er is dus niet altijd te voorspellen wanneer een taak uit een keten afgerond zal zijn. Daarnaast kunnen taken iteratief zijn en meerdere parallelle alternatieven bevatten. Er zijn dan op een punt in de taskchain meerdere vervolgstappen mogelijk waaruit gekozen moet worden (Scacchi, 2001).

Een taak kan worden onderverdeeld in meerdere acties. Acties zijn atomische units werk, zoals het invoeren van data in een computer (Singh, 1995). Deze acties voegen over het algemeen iets toe aan de input van het proces. De bovenstaande definities geven aan dat, wanneer er tijdens het onderzoek naar het proces gekeken wordt, er kan worden gelet op de taken en acties die worden uitgevoerd. Daarnaast kan ook aandacht worden besteed aan de volgorde die hierbij wordt aangehouden en de beslissingen die worden genomen voor bepaalde vervolgacties uit een reeks alternatieve mogelijkheden.

In de definitie van Scacchi (2001) is de mogelijke complexiteit van een proces goed zichtbaar: taken zijn niet altijd voorspelbaar, en er kunnen zich na een taak meerdere alternatieve vervolgtaken voordoen. Davenport (1993) heeft hierop een andere zienswijze: "A structured, measured set of activities designed to produce a specific output for a particular customer or market. (…) A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action. (…) Processes are the structure by which an organisation does what is necessary to produce value for its customers". Naast dat deze definitie meer gericht is op bedrijfsprocessen, verschilt deze fundamenteel van die van Scacchi. Davenport gaat uit van taken met een gedefinieerde volgorde, beginpunt, eindpunt, invoer en uitvoer. Hoewel een stuk overzichtelijker, zal de definitie van Davenport (1993) meestal geen realistische weergave zijn van softwareprocessen. Hierin is de complexiteit van de definitie van Scacchi aanwezig: in het softwareproces zijn vaak verschillende oplossingsmogelijkheden (alternatieven) en zijn niet alle activiteiten volledig vastgelegd.

Het woord proces moet niet worden verward met het woord 'procesbeschrijving'. Het proces staat voor hoe de werkzaamheden in de praktijk worden uitgevoerd. Iets dat men kan observeren en evalueren. Een procesbeschrijving (ook wel procesmodel genoemd) is echter een plan, waarbij wordt beschreven hoe werkzaamheden zouden kunnen of moeten worden uitgevoerd. De procesdefinitie van Davenport (1993) sluit in die zin beter aan bij het woord 'procesomschrijving' dan bij het woord 'proces'. Het werkelijke proces komt zelden volledig overeen met de procesbeschrijving.

Bandinelli en Fuggetta(1995) maken zelfs verschil tussen het gewenste proces, het officiële proces, het geïnterpreteerde proces en het geobserveerde proces. Het gewenste proces is het proces zoals de proceseigenaar dat in zijn hoofd heeft. De beschrijving hiervan is het officiële proces. Het officiële proces is zelden een volledig accurate beschrijving van het gewenste proces. Het officiële proces wordt vervolgens op een bepaalde wijze geïnterpreteerd door de professionals die het proces uitvoeren. Wanneer dit wordt bestudeerd, kijkt men naar het geobserveerde proces.

4.1.2 Definitie van het woord softwareproces

Hoe past het woord softwareproces in deze verhandeling over de definitie van proces? Om te beginnen is er onderscheid tussen 'het softwareproces' en 'een softwareproces'. De eerstgenoemde variant impliceert dat er één softwareproces is. In de praktijk bedoelt men hiermee vaak de verzameling van alle activiteiten die gerelateerd zijn aan de softwareproductie binnen een organisatie (IEEE Computer Society Professionals Practices Committee, 2004). De benaming 'een softwareproces' geeft aan dat er meerdere verschillende softwareprocessen bestaan.

Inhoudelijk kan het softwareproces als volgt worden gedefinieerd: "the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product" (Fuggetta, 2000). Deze definitie voor softwareproces, gaat verder dan puur 'ketens van taken'. Ook de benodigde technologieën, standaarden en procedures maken in zijn definitie deel uit van het softwareproces.

In dit onderzoek wordt een combinatie van de definities van Fuggetta (2000) en Scacchi (2001) gebruikt.
"Het softwareproces bestaat uit meerdere, onderling verbonden ketens van taken en de organisatiestructuren, technologieën, procedures en artefacten die nodig zijn bij het uitvoeren van deze taken. Een keten van taken bestaat uit een niet-lineraire reeks van taken die objecten structureren en omzetten in tussen- of eindproducten. Hierbij zijn de individuele taken niet deterministisch, mogelijk iteratief en kunnen ze meerdere alternatieve vervolgtaken bevatten."

4.1.3 Modelleren van het softwareproces

Een kenmerk van het softwareproces is dat het moeilijk zichtbaar te maken is. Veel van de activiteiten worden door ontwikkelaars gedaan en vinden plaats in de hoofden van de ontwikkelaars (Weinberg, 1998). Het life cycle model is een hulpmiddel bij het zichtbaar en bespreekbaar maken van een proces. Een life cycle model geeft aan hoe software ontwikkeld wordt of kan worden. Hierin is onderscheid te maken tussen 'prescribing lifecycle models' en 'describing lifecycle models' (Scacchi, 2001; Lonchamp, 1993). Eerstgenoemde modellen komen het meest voor, en geven een beschrijving van hoe men software zou moeten ontwikkelen. Laatstgenoemde modellen geven zicht op hoe development werkelijk verloopt, maar zijn op dit moment vanwege het ontbreken van statistische data op dit gebied nog zeer beperkt beschikbaar. Bekende 'prescriptive lifecycle models' zijn het watervalmodel, het incrementele ontwikkelen en het iteratief ontwikkelen.

Life cycle modellen geven een overzicht van het developmentproces op vrij hoog niveau. Een software procesmodel is hierin een stuk gedetailleerder. Een procesmodel geeft meer details over de ketens van taken, en de input en output van elke taak. Dergelijke procesmodellen spelen een belangrijke rol bij de implementatie van processen.

4.2 Proces implementatie

4.2.1 Redenen

Wanneer men overgaat tot implementatie van een bepaald proces, dan betekent dit kort gezegd dat men de werkzaamheden voor het ontwikkelen van software op een vooraf bepaalde manier gaat uitvoeren. Vaak wordt er uitgegaan van een bestaand of bewezen proces model, zoals Rational Unified Proces (RUP) of Dynamic Systems Development Method (DSDM).
Er zijn verschillende redenen om een proces te implementeren (Kruchten, 2002):

- Het helpt bij de communicatie tussen mensen, met name in grotere organisaties.
- Het voorkomt dat teams het proces elke keer opnieuw aan het uitvinden zijn, en opnieuw discussiëren over wat er moet worden gedaan en hoe dit moet worden gedaan.
- Er ontstaat een grotere consistentie in de opgeleverde producten
- Het proces kan een concreet onderwerp van studie worden, en helpen bij reflectie ten einde het proces te verbeteren.

4.2.2 Implementatie van proces in een softwarefactory

Dit onderzoek is gericht op het proces in een softwarefactory. Wanneer is een softwarehuis een softwarefactory? Een veel voorkomende benadering van de softwarefactory is puur vanuit de technologie. Dan is een softwarefactory een set geavanceerde tools voor het maken van (generieke) software op basis van patronen of modellen. Microsoft's invulling hiervan is bijvoorbeeld de Enterprise Framework Factory (Greenfield & Short, 2003), de Health Level Seven Factory (Regio & Greenfield, 2005) en SAP (Frankel, 2004). Deze invulling van het begrip softwarefactory wordt in de literatuur het meeste gebruikt.

In dit onderzoek wordt echter op een andere manier naar de softwarefactory gekeken. Deze wordt gezien als een andere manier van software maken, waarin hergebruik, een gestandaardiseerd proces en daardoor voorspelbaarheid een belangrijke rol spelen. De softwarefactory wordt dan meer dan een toolset, maar wordt een concept. In een software factory zoals in dit onderzoek beschreven wordt getracht om met een relatief groot aantal teams, over meerdere locaties in de wereld, op dezelfde manier te werken. Dit manifesteert zich in ondermeer een gestandaardiseerd toolgebruik en een gestandaardiseerde procesgang.

Wanneer ziet dit onderzoek een softwarehuis als softwarefactory? De organisatie moet het factoryconcept ambiëren, en activiteiten ontplooien voor het doorvoeren van standaardisatie over meerdere developmentteams of afdelingen heen. Is ambitie genoeg om een factory te zijn? Indien een organisatie de ambitie heeft om een factory te zijn, komt het in aanraking met de vraagstukken die in dit onderzoek aan bod komen. Daarnaast zijn er nog vrijwel geen organisaties die hun softwarefactory als 'af' beschouwen. Daarom is dit onderzoek gericht op organisaties die zich bezig houden met het invoeren van het factory concept en worden deze organisaties ook als softwarefactory gezien, ongeacht hoe ver men is met het invoeren hiervan.

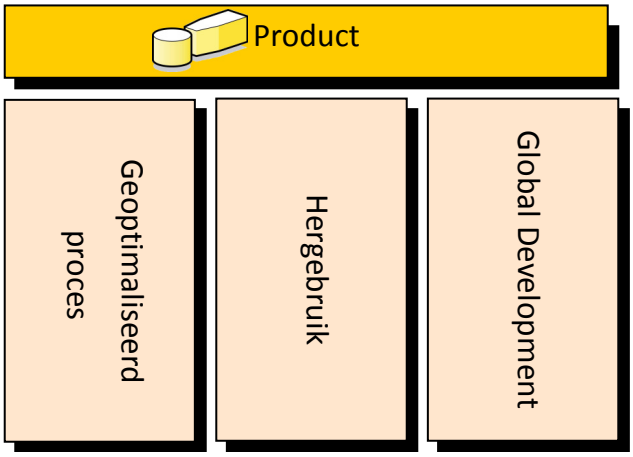
In de geschiedenis hebben grote softwarebedrijven het factorconcept toegepast. Hoewel niet onder de noemer 'factory' had IBM in 1978 een sofwareorganisatie die sterk gecentraliseerd, en georganiseerd rond functies was ingericht (Cusumano, 1991). Er waren grote afdelingen met alleen programmeurs of alleen designers in deze factory. IBM ondervond bij deze softwarefactory, door de consolidatie en integratie van de functionele disciplines (ontwerpen apart van bouwen en testen), dat het management een betere controle had over het ontwikkelen van de producten. Er werden ook nadelen ondervonden. Het aansluiten op de wensen van de klanten was met een dergelijk ver doorgevoerde factory een stuk moeilijker. De overdracht tussen ontwerp-, bouw- en testdepartementen zorgde voor problemen. Uiteindelijk is besloten terug te gaan naar de kleinere teams met daarin meerdere disciplines voor een project. Het nadeel van deze kleinere teams was dat er minder goed kon worden aangesloten op de standaarden, maar men kon hiermee wel een hogere klanttevredenheid bereiken (Cusumano, 1991).

4.2.3 Het Product Centre

De softwarefactory waarin dit onderzoek is uitgevoerd is het Product Centre van BedrijfX. Het visie-document van het Product Centre stelt het volgende:

> "De doelstelling van het Product Center is het ontwikkelen van software op voorspelbare manier, van hoge kwaliteit, in minder tijd en met lagere kosten; beter, goedkoper, sneller, voorspelbaarder."

Daarnaast heeft het Product Centre de ambitie om opdrachten verdeeld over verschillende plaatsen in de wereld uit te laten voeren. Dit heeft voordelen vanuit het oogpunt van kosten voor het beperken van time to market. Er zijn meerdere Product Centres ingericht op verschillende locaties in de wereld. Door gebruik te maken van het Product Centre in India, kan men bepaalde componenten goedkoper produceren dan in Nederland. Daarnaast kunnen componenten die in Nederland worden ontwikkeld, aan het eind van de dag in een andere tijdzone, waar de werkdag net start, worden getest. De mogelijke voordelen van het factoryconcept zoals dat van het Product Centre zijn duidelijk. Ook duidelijk is dat 'Global Development', zoals hier geschetst eisen stelt aan de organisatiestructuren en de processen. Er moet een naadloze overgang mogelijk zijn tussen verschillende teams, in verschillende landen.



Figuur 4: Het Product Centre concept

Dit onderzoek is niet gericht op de specifieke vraagstukken die Global Development met zich mee brengen. Wel is nu helder welke motivatie er is voor het gebruik van het factoryconcept door BedrijfX, en dat dit eisen stelt aan het softwareproces van de organisatie.

Bij elke vestiging van het Product Centre zijn verschillende ontwikkelstraten ingericht. Een ontwikkelstraat ondersteunt het ontwikkelproces van begin tot eind voor een specifiek platform, zoals Microsoft .NET of Java/Oracle. Het softwareproces en de tooling zijn echter over de verschillende straten en vestigingen gestandaardiseerd. Het softwareproces is gestandaardiseerd in de vorm van het Product Model. Dit model is gebaseerd op RUP, waaraan eigen processen, templates, checklist en handleidingen zijn toegevoegd. Alle medewerkers kunnen dit model benaderen via het intranet. Naast het gestandaardiseerde model, is er een standaard werkplek waarin de standaard toolset van Result is opgenomen.

4.2.4 Implementatie van een proces, en niet alleen op papier

Een van de meest complexe zaken van procesimplementatie is ervoor te zorgen dat de ideeën en afspraken niet eindigen op de plank, maar daadwerkelijk in de praktijk van meerwaarde zijn. Een procesmodel of methode kan zijn waarde pas bewijzen wanneer er in de praktijk gebruik van gemaakt wordt.

Iets dat de implementatie van (veranderde) processen in de praktijk moeilijk maakt is de natuurlijke weerstand van mensen tegen verandering (Chroust, 2002). Zaken als statusverlies, vermindering van flexibiliteit en verlies van creativiteit zijn voorbeelden van zaken die mensen als problematisch ervaren bij een verandering van werkwijze. Dit zijn dus zaken waar bij implementatie van een nieuw proces rekening mee gehouden moet worden (Herbsleb & Goldenson, 1996; Gallivan, 2002). Gallivan (2002) heeft deze weerstand tegen veranderingen bekeken voor verschillende soorten softwareengineers. Hij gaat uit van een onderverdeling in innovators en adaptors als voorspeller voor hoe gemakkelijk iemand wijzigingen in werkwijze overneemt (Gallivan, 2002). In zijn visie zijn er verschillende soorten mensen in de organisatie, waarbij de groep innovators verandering sneller omarmt dan de groep adaptors. De innovators kunnen een belangrijke rol spelen bij het doorvoeren van de veranderingen in de organisatie.

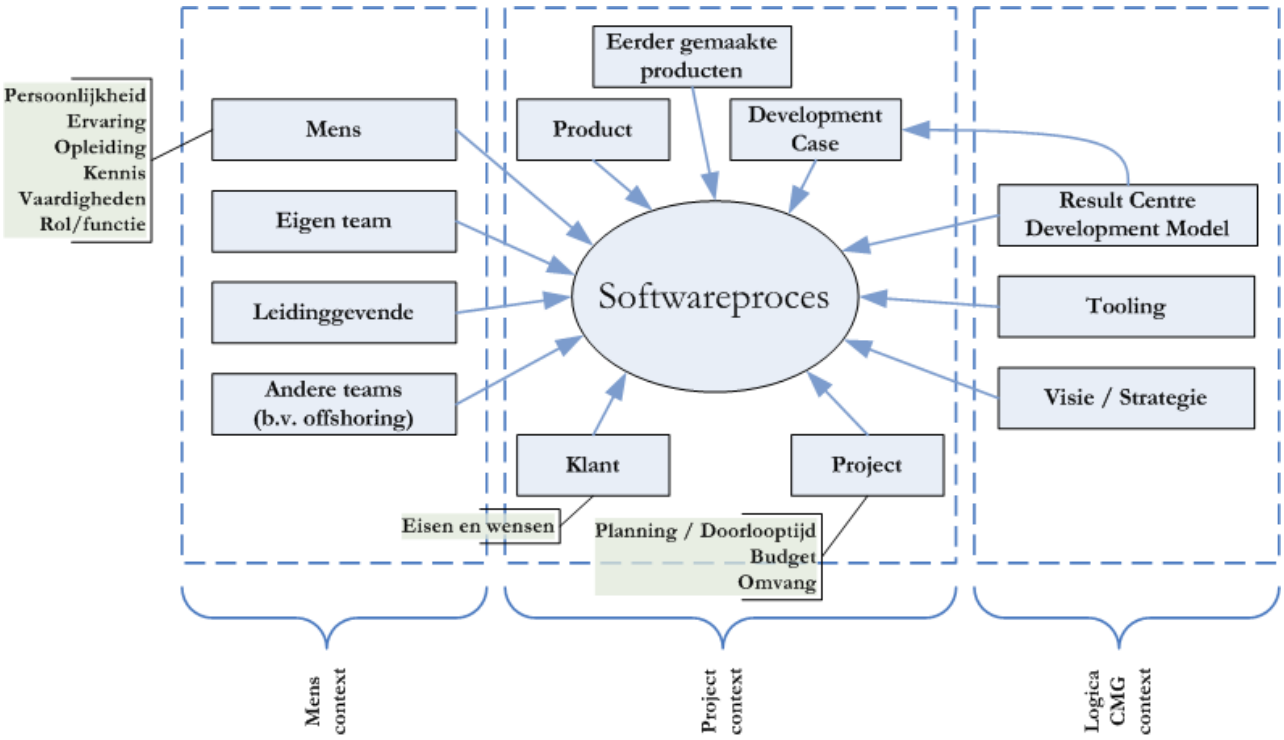
Een andere verklaring voor het feit dat nieuwe processen niet zo snel in de praktijk worden overgenomen kan de manier van denken zijn die in veel onderzoeken en procesmodellen terugkomt. Sommerville noemt dit 'mechanisch' denken (Sommerville & Rodden, 1996). Processen worden daarbij gezien als kleine computerprogramma's, met een input en een output en deterministisch gedrag. Mensen gedragen zich echter niet deterministisch, zo vervolgt Sommerville. Ze zijn onderhevig aan veranderlijke doelen en worden beïnvloed door een groot aantal factoren. Procesmodellen zijn soms onvoldoende gericht op het toepassen of uitvoeren van het model (Lehman, 1988).

4.2.5 Beïnvloeding van het proces

De factoren die mensen beïnvloeden, en daarmee het softwareproces beïnvloeden, zijn ook aandachtspunten in de softwarefactory. Een softwareproces wordt niet alleen bepaald door de geldende standaard werkwijze en de wil van de professional om zich daar aan te conformeren. Er zijn meer invloeden van belang.

In de praktijk komt de volgende situatie regelmatig voor: de klant wil resultaat op zeer korte termijn, en opleveren op zeer korte termijn is voorwaarde voor het gunnen van de opdracht.
Indien het project wordt geaccepteerd, is er wellicht geen tijd voor een volledig functioneel ontwerp, of een technisch ontwerp zoals dat volgens de standaard werkwijze is voorgeschreven. Het kan zijn dat men in bepaalde situaties dit type opdrachten wil uitvoeren, bijvoorbeeld gemotiveerd vanuit commercieel oogpunt. Het is in dat geval belangrijk om bewust hiervoor te kiezen, en de invloed die dit heeft op het softwareproces zichtbaar te hebben.

Naast het geschetste voorbeeld zijn er nog tal van zaken van invloed op het softwareproces. Ter voorbereiding op het bestuderen van het proces in de praktijk is een aantal mogelijke invloeden in beeld gebracht in onderstaand model.



Figuur 5: Invloeden op het softwareproces

Het softwareproces wordt voor een aanzienlijk gedeelte beïnvloed door de mensen die het proces uitvoeren. De programmeurs, de teamdynamiek, de leidinggevende en de samenwerking met andere teams, het beïnvloedt allemaal hoe het proces verloopt. Ook zaken uit de projectcontext beïnvloeden het proces. De klant stelt bijvoorbeeld bepaalde eisen, of eerder gemaakte producten worden nu op eenzelfde manier gemaakt. Ten slotte is er vanuit de BedrijfX context invloed vanuit het gestandaardiseerde developmentmodel en de gestandaardiseerde tooling.

4.2.6 Het toepassen van een proces

De meeste softwarehuizen zijn ingericht als projectorganisatie. Een standaard aanpak of proces is geïmplementeerd wanneer het wordt toegepast bij projecten. Hierbij ontstaat een aantal nieuwe uitdagingen. Een van de uitdagingen is het vertalen van het standaard proces naar de projectsetting in de praktijk. De procesbeschrijving is altijd geabstraheerd om toepasbaar te zijn op uiteenlopende projecten. Hierdoor is de procesbeschrijving niet meer specifiek genoeg om naadloos aan te sluiten op de praktijksituatie. Dit heeft als gevolg dat men in de praktijk bij projecten in situaties terecht kan komen waarin de taskchain niet aansluit op de werkelijkheid.

Het kan zijn dat de standaard werkwijze voorschriften heeft die niet van toepassing zijn op de projectsituatie, of dat er zaken ontbreken die wel nodig zijn bij het project, maar waarvoor geen standaard bestaat. Op deze punten volgt een aantal stappen die niet gedefinieerd zijn. Deze stappen brengen de gedefinieerde taskchain weer op gang, of zorgen voor overgang naar een andere taskchain. Deze niet gedefinieerde maar essentiële taken worden ook '*articulatie*' genoemd (Scacchi, 2001). De articulatie is essentieel voor het projectsucces, omdat er zonder articulatie al snel vertraging optreedt omdat de juiste vervolgtaken niet opgestart worden. Een belangrijk voorbeeld van dergelijke articulatie is het consulteren van experts (Sommerville & Rodden, 1996). In veel projecten komt het voor dat professionals een collega consulteren, voor dat een bepaalde beslissing genomen wordt. Het raadplegen van een expert kan een belangrijke impact hebben op het projectsucces. Toch is deze activiteit zelden onderdeel van de procesbeschrijving.

Met het toepassen van een standaard proces, ontstaan er ook direct projecten en situaties waarin men afwijkt van het standaard proces. Deze afwijkingen kunnen we zien als articulatie van het proces (invullen van de zaken die niet gedefinieerd zijn), maar in sommige gevallen gaat het ook om het afwijken van wel gedefinieerde zaken. Deze afwijkingen kunnen we *proces inconsistenties* noemen (Sommerville, Sawyer, & Viller, 1999). Deze inconsistenties kunnen zitten in de wijze waarop mensen het proces uitvoeren, organiseren of waarnemen. Deze inconsistenties zijn niet per definitie negatief, en komen in drie vormen voor (Sommerville, Sawyer, & Viller, 1999):

- De inconsistentie is bij toeval ontstaan (er zijn geen specifieke redenen voor de inconsistentie). Op deze punten kan aansluiting op de standaarden mogelijk worden hersteld.
- De inconsistentie komt voort uit redundantie (er zit overlap in processen) Deze redundantie in het standaardproces kan mogelijk worden weggenomen.
- De inconsistentie is een voorbeeld van een goede aanpak (een succesvolle variant op het standaard proces). Deze succesvolle varianten kunnen mogelijk worden opgenomen in de standaard werkwijze.

4.3 Proces verbetering

Naast definitie en implementatie van processen, is ook Software Process Improvement (SPI) een eigen vakgebied geworden. Ook het Product Centre heeft hier aandacht voor. Een van de pijlers van het Product Centre is 'Geoptimaliseerd Proces'.

4.3.1 Resultaten die met SPI behaald zijn

Er is onderzoek gedaan naar het effect van de procesverbeteringen in de praktijk. Deze SPI initiatieven, zoals bij Motorola (Diaz & Sligo, 1997; O' Hara, 2000) en Hughes Aircraft (Humprey, Snyder, & Willis, 1991) zijn vaak gebaseerd op CMMi (Capability Maturity Model), en hebben positieve resultaten bereikt. Bij de succesvolle initiatieven valt op dat er in het bijzonder aandacht is besteed aan het betrekken van de uitvoerende software professionals bij de SPI (Rainer & Hall, 2001). Dit zijn de professionals met domeinkennis, die weten hoe het er echt aan toe gaat in de organisatie. Het betrekken van de software professionals bij procesverbetering lijkt dan ook van essentieel belang in succesvolle SPI projecten.

Van negatieve ervaringen met SPI projecten is minder informatie bekend. Dit zou kunnen volgen uit het feit dat minder succesvolle resultaten bij bedrijven wellicht minder snel gepubliceerd worden.

4.3.2 Aanpak van SPI

Er zijn twee stromingen te herkennen in hoe men met procesverbetering omgaat. Deze kunnen globaal gezien gegroepeerd worden als top-down en bottom-up.
De bottom-up variant gaat uit van de groep mensen die de meeste activiteiten in het proces uitvoeren. In het artikel 'enough about process: what we need are heroes' geeft J. Bach (1995) aan dat gedrag en daarmee werkwijze niet los is te zien van de mens. De vraag moet volgens hem niet alleen gaan over 'wat is het proces?' maar dus ook 'wie is het proces?'. Er zijn meerdere onderzoeken die succes rapporteren met deze bottom-up aanpak (Badddoo, Hall, & Wilson, 2000; Sommerville & Rodden, 1996).

Naast de bottom-up aanpak is er de top-down aanpak waarbij men het proces ziet als een set regels en afspraken, die van hogere hand bepaald worden. Er is vaak een apart 'proces improvement team' dat zich bezig houdt met het specificeren van hoe het proces er uit zou moeten zien. In de uiterste vorm zijn alle processen zodanig gedetailleerd gespecificeerd, dat het gehele softwareproces geautomatiseerd kan worden. 'Software Processes are Software Too' van Osterweil (1987) geeft hiervoor een pleidooi. In zijn visie kan *proces programming* er voor zorgen dat processen worden geautomatiseerd in programma's. Dit maakt de processen herbruikbaar en duidelijk zichtbaar. Deze mechanische zienswijze van Osterweil is becommentarieerd door ondermeer Lehman (1988; 1987). Voorwaarde voor het volledig automatiseren van het proces is namelijk dat het proces gedefinieerd of gemodelleerd is. Lehman is van mening dat het definiëren van het proces een belangrijk deel uitmaakt van het softwareproces zelf, en dat daarom volledige automatisering onhaalbaar is. In een meer recente publicatie adresseert Osterweil de dynamiek van projecten en stelt hij dat hiermee kan worden omgegaan door de punten waarop flexibiliteit benodigd is te specificeren (Osterweil L. J., 2003).

In de praktijk zal de SPI benadering binnen een bedrijf vaak een mengvorm zijn van de bottom-up aanpak en de top-down aanpak. Hierbij wordt bijvoorbeeld wel gewerkt met een apart team dat processen specificeert, maar worden de medewerkers die de processen uitvoeren hierbij betrokken door middel van workshops of trainingen. Een andere aanpak is het gebruik van change agents (Gallivan, 2002). Change agents zijn software engineers uit verschillende teams en lagen in de organisatie die betrokken worden bij SPI. Zij zullen de acceptatie van wijzigingen die SPI met zich meebrengt versoepelen en de kennis verspreiden over de organisatie.

4.3.3 SPI door een betere afstemming van professionals op het proces

Zoals eerder genoemd is een voorwaarde voor SPI succes een aansluiting van het nieuwe proces op de professionals die het werk uitvoeren. Dit kan bijvoorbeeld door de taken die mensen tijdens het softwareproces uitvoeren af te stemmen op de persoonlijkheidskenmerken (Acuña, Juristo, & Moreno, Modelling Human Competencies in the Software Process, 2003). Acuña en Juristo behalen goede resultaten in een experiment waarbij de rollen die mensen innemen binnen een softwareproces zijn afgestemd op hun capaciteiten (die zijn afgeleid van hun persoonlijkheidskenmerken). Voor elke rol hebben zij aangegeven welke capaciteiten (zoals decision making, judgement, discipline, empathy) er nodig of juist niet nodig zijn om een rol optimaal te vervullen.

De onderzoeken van Acuña en Juristo (2003) wijzen er op dat persoonlijkheidskenmerken een voorspeller kunnen zijn voor prestaties binnen een bepaalde rol. Op taakniveau zou men ook afstemming op de softwareprofessional kunnen proberen te vinden. Het wordt dan mogelijk het softwareproces zodanig in te richten dat de juiste mensen vrijheid krijgen bij de juiste taken uit het proces en sturing en begeleiding ontvangen op specifieke punten (op basis van de persoonlijkheidskenmerken).

4.3.4 SPI en bureaucratie

Sommige vormen van SPI brengen een vorm van extra bureaucratie met zich mee. Dit komt bijvoorbeeld doordat in de nieuwe processen extra controle of documentatiestappen zijn opgenomen. Conradi en Fugetta (2002) stellen dat SPI altijd gericht zou moeten zijn op leren, en niet op controleren. De houding die de organisatie hierin aanneemt, lijkt van invloed op de bureaucratie die ontstaat. Ten aanzien van deze bureaucratie kan men twee stellingen innemen: Bureaucratie vermindert creativiteit, zorgt voor minder voldoening en motivatie bij medewerkers. Aan de andere kant kan men stellen dat bureaucratie de benodigde sturing geeft aan medewerkers en verantwoordelijkheden verduidelijkt waardoor stress wordt verminderd en individuen meer effectief gaan werken.

De verklaring voor deze twee standpunten ten aanzien van eenzelfde begrip ligt volgens Adler en Borys (1996) in het feit dat hiermee twee soorten bureaucratie worden beschreven: Enabling en Coercive bureaucratie. Van beiden soorten bureaucratie hebben zij de eigenschappen op een rij gezet:

	Enabling bureaucratie (helpende bureaucratie)	Coercive bureaucratie (dwingende bureaucratie)
Repair	Het proces bevat een manier om fouten ongedaan te maken.	Afwijking van de normale werkwijze is verdacht.
Internal transparancy	Uitleg van de belangrijkste stappen en best-practices. Onderliggende theorie van de processen wordt toegelicht.	Definitie van taken.
Global transparancy	Zicht op de rest van de organisatie, welke schakel doe ik in het totale proces.	Alleen toegang tot de informatie die nodig is voor de eigen taken. Global transparancy is een risico.
Flexibility	De gebruiker geeft het systeem taken.	Het systeem geeft de gebruiker taken.

Tabel 1: Soorten bureaucratie (Adler & Broys, 1996)

Een andere indeling van bureaucratie is die in bureaucratie en post-bureaucratie (Maravelias, 2003) of Taylorism en post-Taylorism (Pruijt, 2000). Bureaucratie staat volgens Maravelias (2003) voor standaardisatie, formalisatie, centralisatie en functionele specialisatie. Post-bureaucratie verandert de relatie tussen de organisatie en het individu. Doordat gedefinieerde rollen en taakomschrijvingen vervagen, maakt de organisatie aanspraak op het gehele individu. Waar bureaucratie de karakteristieken van individuen probeert te normaliseren, is post-bureaucratie erop gericht deze

verschillen tussen personen optimaal te benutten door ze meer vrijheid te geven. In een post-bureaucratie is een meer gedecentraliseerde verdeling van macht in de organisatie. De doelstelling van beide soorten bureaucratie is echter gelijk. Beide soorten bureaucratie hebben als doelstelling het gedrag van de mensen zodanig te beïnvloeden, dat dit positief is voor de productie in de organisatie. In de post-bureaucratie vindt deze beïnvloeding plaats door het ontbreken van een duidelijke rol en taak. Het ontbreken van een duidelijke identiteit in de organisatie zorgt er voor dat individuen hun eigen vaardigheden en kennis optimaal gaan gebruiken om zich te onderscheiden (Maravelias, 2003). Taylorism en post-taylorism heeft sterke overeenkomsten met de indeling van Marvelias. Hierbij geeft Pruijt (2000) nog als verfijning aan, dat de kans bestaat dat er binnen de vrijheid die het post-Taylorism met zich mee brengt, een vorm bureaucratie of Taylorism ontstaat binnen de teams. Dit is mogelijk bij teams waar teamleden hun werkwijze opleggen aan andere teamleden, zelfs in een organisatie die post-Taylorism aanhangt.

Bij het bestuderen van het softwareproces in de softwarefactory is het interessant deze verschillende soorten bureaucratie in beeld te nemen. Er is een spanningsveld tussen de enabling en coercive vormen van bureaucratie. Het voorkomen van fouten is hiervan een goed voorbeeld. De meeste processen bevatten mechanismen om fouten te voorkomen of te constateren. Bij SPI initiatieven is er ook aandacht voor het voorkomen van fouten. Alle fouten voorkomen vraagt echter een bepaald soort bureaucratie, waarin er bepaalde keuzes zijn gemaakt met betrekking tot Repair (zie Tabel 1). Hebben medewerkers bijvoorbeeld de ruimte om fouten te maken en deze te herstellen, of worden fouten gerapporteerd en als afwijkend gezien? Een andere vraag is of het proces het maken van bepaalde fouten voorkomt, of dat de fouten slechts terug zijn te vorderen op wie ze heeft gemaakt (gedocumenteerde fouten). Deze afwegingen tussen dwingende en helpende bureaucratie beïnvloeden het softwareproces en vervolgens ook de mate waarin dit proces doelmatig is.

In dit rapport zal naast het woord bureaucratie ook gesproken worden van '*sturing'*. Stel men heeft een werkwijze X. Wanneer een bureaucratie wordt ingevoerd die exact werkwijze X voorschrijft, dan ervaart men dit niet als een verandering of als een bureaucratisch systeem. Daarom wordt gesproken van sturing, hiermee wordt de situatie bedoeld waarin men zich anders gedraagt dan zonder bureaucratisch systeem.

4.3.5 Wat maakt SPI moeilijk

Over het algemeen blijken SPI initiatieven die één software engineering project ondersteunen succesvoller dan algemene SPI initiatieven die op een breed scala aan software engineering projecten zijn gericht (Börjesson & Mathiassen, 2003). Bij project gerichte SPI kunnen change agents de wijzigingen koppelen aan de projectdoelstellingen en daarmee samen met de softwareengineers succes behalen. Bij de algemene SPI initiatieven is het moeilijker voor SPI change agents om de algemene wijzigingen te koppelen aan projectdoelstellingen. Softwarefactory's proberen juist dit laatstgenoemde te bereiken: één SPI initiatief voor een scala aan projecten.

Daarnaast zijn er vaak minder standaarden gedefinieerd die meerdere teams overspannen. Er is bijvoorbeeld niet gedefinieerd hoe samenwerking tussen de verschillende teams dient te verlopen en workflowsystemen werken vaak binnen teams, maar niet daarbuiten. Belangrijk bij deze standaard

over meerdere projecten heen is de relatie tussen het management van de projectteams en de rest van de organisatie (Rada, 2000).

In de literatuur komen commitment van het management, betrokkenheid van de medewerkers en training / begeleiding van medewerkers als belangrijke succesfactoren voor bij SPI programma´s (Niazi, Wilson, & Zowghi, 2003).

4.4 Proces meting en beoordeling

Om te kunnen vaststellen of de verbeteringsintiatieven ook daadwerkelijk zinvol zijn en hun vruchten afwerpen, is het nodig te kunnen meten welke verbeteringen er bereikt worden. Ook voor het vergelijken en onderzoeken van processen zijn meetgegevens en beoordelingen van belang. Deze paragraaf beschrijft twee veelgebruikte methoden voor het meten en beoordelen van het ontwikkelproces.

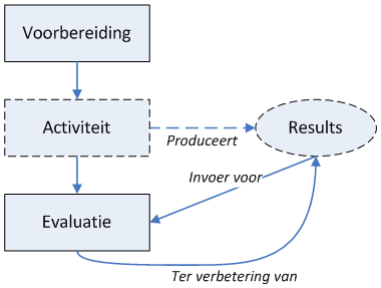
4.4.1 Certificeren

Capability Maturity Model Integration [CMMi] en International Organisation for Standardiszation [ISO] vergelijken de situatie binnen een organisatie met een ijkpunt. Een belangrijk verschil tussen ISO en CMMi is, dat CMMi een groeimodel is (uit meerdere levels bestaat) en ISO een meer statisch en generiek model is waarbij geen opdeling in levels is: men is ISO gecertificeerd of men is het niet.

Organisaties die aan de CMMi eisen voldoen kunnen worden gecertificeerd. Deze certificeringen meten ondermeer de kwaliteit / volwassenheid van het proces. CMMi bestaat uit 5 niveaus, waarin een organisatie kan groeien. Een niveau 1 organisatie kan als volgt worden weergegeven:

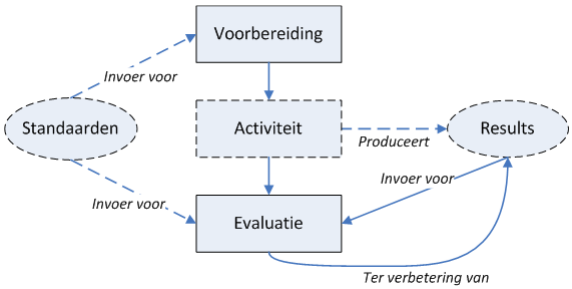
Figuur 6: CMMi niveau 1, puur het werk uitvoeren (Caputo, 1998)

In een niveau 2 organisatie is meer een voorbereiding- en evaluatiefase opgenomen. Met behulp van de evaluatiefase worden de resultaten verbeterd.



Figuur 7: CMMi niveau 2, nadenken voor en na de activiteit, om zeker te zijn van een goed verloop (Caputo, 1998)

In een niveau 3 organisatie worden de resultaten van evaluaties omgezet in standaarden. Deze standaarden kunnen vervolgens bij andere projecten van waarde zijn.



Figuur 8: CMMi niveau 3, gebruik de ervaringen uit het verleden (Caputo, 1998)

Zoals bij alle certificeringstrajecten bestaat het gevaar dat het verbetertraject op den duur alleen nog gericht is op de certificering. Het is voor bedrijven dan belangrijker geworden, ook vanuit commercieel oogpunt, om zo snel mogelijk een bepaalde status te behalen, dan om werkelijk de kwaliteit te verbeteren (Bach, 1994). Het is binnen veel organisaties herkenbaar dat werkzaamheden voor een audit anders worden aangepakt of tijdelijk worden verbeterd, zodat het certificaat gehaald kan worden. De verbeteringen die de certificering in potentie kunnen opleveren worden dan niet volledig gehaald.
Daarnaast wordt met name het proces gemeten, en niet de kwaliteit van het product. De benadering is namelijk dat een goed proces ook een kwalitatief goed product betekent, alleen is dit laatste niet altijd het geval (Heemstra, Kusters, & Trienekens, 2001).

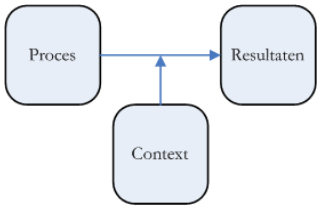
4.4.2 Metrieken

Naast het certificeren zijn ook metrieken een manier om het proces te beoordelen. Vaak wordt op basis van de resultaten van een proces gemeten. De gemeten waarden op de resultaten van het proces (bijvoorbeeld op het product), geven een beeld van de efficiëntie en de kwaliteit die het proces oplevert.

Voor het meten van efficiëntie zijn het aantal functiepunten (FP) per uur of het aantal regels code (LOC) veel gebruikte metrieken (Heemstra, Kusters, & Trienekens, 2001). Het functiepuntensysteem is een manier om de functionele omvang van een systeem in te schatten. Deze techniek gaat uit van de te realiseren functionaliteit en het tellen van bepaalde elementen daar in. Op basis van de getelde waarden wordt een aantal functiepunten bepaald voor een stuk te maken of gemaakte functionaliteit. De FP methode wordt vaak gebruikt voor het begroten van de bouwinspanning voor informatiesystemen.

Het aantal regels broncode per uur is een andere manier om de efficiëntie van een developmentproces te meten. Deze metriek kan beïnvloed worden door veel andere zaken zoals complexiteit van functies, programmeertaal en complexiteit van bestaande code. Het toevoegen van enkele regels complexe code kan bijvoorbeeld veel meer tijd kosten dan het toevoegen van een grote lap gegenereerde of eenvoudige code.

Naast metrieken die de efficiëntie zichtbaar maken, zijn ook metrieken voor kwaliteit van een proces interessant. Hier worden, net zoals bij de efficiëntie metrieken, de resultaten van een proces gebruikt voor het bepalen van de kwaliteit van het proces. Een veelgebruikte manier is het tellen van het aantal fouten dat in de geproduceerde software wordt gevonden tijdens de test. Het aantal fouten per 100 regels code kan dan een indicatie zijn voor de kwaliteit van het proces (Heemstra, Kusters, & Trienekens, 2001).



Figuur 9

De genoemde metrieken meten op basis van het resultaat van het proces. Er wordt dus niet direct gemeten op het proces. Zoals in Figuur 9 aangeduid wordt, is ook de context van invloed op de resultaten. Bij het tellen van het aantal fouten in de software kan de context het aantal fouten verhogen of verlagen. Een groep uitzonderlijke programmeurs of een eenvoudig programma, heeft naar verwachting minder fouten dan een programma binnen een andere context. In dit geval zegt het verschil in fouten per 100 regels niet direct iets over het softwareproces. Omgevingsinvloeden zullen aandacht moeten krijgen bij het interpreteren van deze metrieken. Een overzicht van mogelijke omgevingsinvloeden is reeds gegeven in paragraaf 4.2.5.

5 Opzet van het onderzoek

Dit hoofdstuk geeft de vertaling van de onderzoekvragen (wat) naar de onderzoeksmethode (hoe). Er wordt beschreven op welke manier er antwoorden gezocht worden op de gestelde vragen.

5.1 Onderzoekstype

"Welke factoren beïnvloeden de doelmatigheid van het proces in een softwarefactory?"
Dit is een open onderzoeksvraag die gericht is op het begrijpen van een situatie. Voor dit type vragen is empirisch en exploratief kwalitatief onderzoek geschikt. Bij dit type onderzoek kan er gedetailleerd naar bijvoorbeeld één softwareproces gekeken worden, om met de gedane waarnemingen een inzicht te krijgen in de factoren die de doelmatigheid beïnvloeden.

Het exploratieve karakter van het onderzoek komt voort uit het feit dat er nog beperkt inzicht is in mogelijke onderzoeksuitkomsten.

5.2 Onderzoeksomgeving

Er worden metingen gedaan bij twee bedrijven. Eén bedrijf is BedrijfX, waar binnen het Product Centre een factoryconcept wordt toegepast. Het Result concept is geïntroduceerd in paragraaf 4.2.3. Daarnaast wordt bij een ander bedrijf gemeten, waar niet volgens het factoryconcept wordt gewerkt. Een team binnen dit bedrijf wordt als referentiegroep gebruikt. Door te meten in verschillende omgevingen, kan worden bekeken wat het factoryconcept nu precies veranderd, en welke effecten zichtbaar zijn.

De bedrijfsomgeving zonder factoryconcept bestaat vijf jaar, en hierin werken teams zo veel mogelijk zonder sturing. Teams in deze organisatie worden beoordeeld op de prestaties, maar mogen zelf de werkwijze en tools bepalen. Tijdens dit onderzoek werkt een team van vier professionals mee uit deze organisatie. De organisatie is bij het onderzoek betrokken als referentie, het onderzoek is gericht op BedrijfX.

In het kader van "The proper place to study elephants is the jungle, not the zoo" (McLean, 1973), is besloten metingen uit te voeren bij projecten in de praktijk en niet uit te gaan van een labsituatie. De reden hiervoor is dat de waarnemingen uit het onderzoek zo betrouwbaar mogelijk moeten zijn, en de werkwijze van mensen zeer snel beïnvloed wordt door een veranderingen in de omgeving. Bij waarnemingen in een labsituatie is het zeer moeilijk uit te sluiten welke invloed de labsituatie heeft op de manier waarop het proces is verlopen en daarmee op de waarnemingen die gedaan worden.

Een tweede reden om waarnemingen in de praktijk te doen is het voorkomen van de mechanische kijk op het softwareproces, zoals eerder aan bod kwam in hoofdstuk 4.2.4. Een mechanische kijk op proces zou de toepasbaarheid van onderzoeksuitkomsten in de praktijk namelijk moeilijker maken (Sommerville & Rodden, 1996).

5.3 Metingen

Om de onderzoeksvraag te kunnen beantwoorden, is een aantal concrete deelvragen geformuleerd. Deze deelvragen staan in dienst van de hoofdonderzoeksvraag. De verwachting is dat het niet alleen de antwoorden op de subvragen zijn die helpen de hoofdvraag te beantwoorden, maar dat het onderzoeken van de subvragen zelf een zeker inzicht oplevert. Dit inzicht, dat voortkomt uit de meetresultaten, gesprekken en observaties helpt vervolgens bij het beantwoorden van de hoofdvraag.

De performancemeting die tijdens het project is uitgevoerd is hier een goed voorbeeld van. De resultaten van de performancemeting zijn niet alleen de cijfers die gemeten zijn. De waarnemingen die zijn gedaan bij het uitvoeren van de meting helpen bij verkrijgen van inzicht in het proces en de zaken die de doelmatigheid ervan beïnvloeden.

In dit onderzoek zijn subvragen geformuleerd. Per subvraag is beschreven op welke wijze er onderzoek wordt gedaan om deze vraag te kunnen beantwoorden. De verschillende metingen die zijn uitgevoerd worden in elk in een eigen hoofdstuk in detail beschreven.

Vraag	Onderzoekswijze
In welke mate is het Product Centre een softwarefactory?	In hoofdstuk 4 zijn de eigenschappen van een softwarefactory benoemd. De waarnemingen bij het bestuderen van het proces (hoofdstuk 7) en de resultaten van het proces (hoofdstuk 8) zullen worden gebruikt voor het beantwoorden van deze vraag.
Zijn er duidelijke verschillen tussen een factory en een niet factory softwareproces?	In paragraaf 4.1 en 4.2 is beschreven wat software proces is en welke implementaties er mogelijk zijn. Binnen twee organisaties wordt in detail het softwareproces binnen twee projecten gevolgd. De aanpak van de software professionals wordt geregistreerd en geanalyseerd. Daarnaast worden er gesprekken gevoerd met verschillende medewerkers in de factory en buiten de factory over de gehanteerde werkwijze. De details van deze analyse zijn opgenomen in hoofdstuk 7.
Hoe ervaren hele goede professionals en gemiddelde professionals het softwareproces binnen het Product Centre?	In hoofdstuk 6 wordt beschreven hoe onderscheid wordt gemaakt tussen goede en gemiddelde professionals. Vervolgens is op basis van observaties (hoofdstuk 7) en een vragenlijst (hoofdstuk 8) een antwoord op deze vraag gezocht.
Is de efficiëntie en kwaliteit van een software factory altijd hoger dan een niet factory?	Tijdens de observatie van het proces (hoofdstuk 7) werden indicaties zichtbaar die iets zeggen over de efficiëntie en kwaliteit van het proces. In paragraaf 4.4 zijn manieren beschreven waarop het proces meetbaar kan worden gemaakt. Op basis hiervan is er een performancemeting uitgevoerd. De resultaten zijn opgenomen in hoofdstuk 8.

Tabel 2: Onderzoeksvragen

Zoals zichtbaar is er een relatief groot aantal vragen en metingen in de onderzoeksaanpak opgenomen. Deze verschillende metingen komen voort uit het exploratieve karakter van het onderzoek en kunnen allemaal indicaties opleveren van factoren die de doelmatigheid van het softwareproces beïnvloeden. In hoofdstuk 9 zullen de resultaten van de verschillende metingen worden samengevoegd in de conclusie.

Elk van de geformuleerde subvragen kan op zichzelf een hoofdvraag zijn van een onderzoek. Aan het onderzoeken van de verschillen in kwaliteit tussen een factory en een niet-factory bijvoorbeeld, kan zelfstandig onderzoek worden gewijd. Het is belangrijk om te vermelden dat het onderzoek naar de deelvragen gericht is op het doen van waarnemingen vanuit verschillende bronnen en met verschillende meetmethoden. Dit vergroot het inzicht in het onderzoeksgebied, en dit inzicht helpt bij het beantwoorden van de hoofdvraag.

5.4 Validatie

Bij elk onderzoek, kwalitatief of kwantitatief, is aandacht voor validatie onderdeel van de onderzoeksaanpak. In de aanpak wordt rekening gehouden met de validiteit van de resultaten die de aanpak gaat opleveren. Burke Johnson (1997) maakt onderscheid in validatie op drie verschillende vlakken:

Validiteit van de beschrijving	Hoe accuraat is de feitelijke beschrijving van die de onderzoeker oplevert van het gedane werk?
Validiteit in de interpretatie	In welke mate zijn de uitspraken, gedachten, intenties van de geïnterviewde begrepen en goed geïnterpreteerd door de onderzoeker?
Validiteit van de theorie	In welke mate is de door de onderzoeker gevormde theorie op basis van de gemeten gegevens plausibel en te verdedigen?

Tabel 3: Soorten validatie (Johnson, 1997)

Voor alle drie de bovengenoemde soorten validatie is de onderzoeker, en vertekening door de onderzoek een punt dat kan leiden tot problemen. Daarnaast zijn er nog andere effecten die het onderzoek kunnen vertekenen. Uit een aantal mogelijke maatregelen zijn de volgende gekozen:

- Gegevens verzamelen over een (relatief) langere periode
  In dit onderzoek zijn de metingen verricht over een aantal maanden.
- Triangulatie van databronnen en onderzoekstechnieken
  Tijdens het onderzoek zijn verschillende bronnen geraadpleegd, en zijn verschillende onderzoekstechnieken gebruikt.
- Gebruik van “Low inference descriptors” in de resultaten
  Door op sommige gebieden directe quotes bij de resultaten op te nemen van de deelnemers aan het onderzoek, kan de validiteit en aannemelijkheid van bepaalde conclusies worden versterkt.
- Anonieme deelname
  De deelnemers deden zoveel mogelijk anoniem mee aan het onderzoek, om sociaal wenselijke antwoorden te minimaliseren.

- Voorkennis gelijk houden
  De deelnemers aan het onderzoek kregen voor de deelname aan het onderzoek een introductie op het onderzoek. Met deze introductie kan de onderzoeker de deelnemers ongewenst beïnvloeden. Tijdens het onderzoek is gewerkt met een introductie op papier, die bij alle deelnemers hetzelfde is.

6 Eén factory, één proces: verschillende professionals

Het kader van dit onderzoek wierp de volgende vraag op: “Is er één optimaal softwareproces, of hebben verschillende professionals een verschillend optimaal softwareproces?” Het antwoord op deze vraag heeft impact op de doelmatigheid van het softwareproces in de factory. In een factory wordt aan één optimaal proces gewerkt. Mochten de verschillende professionals het proces bijvoorbeeld verschillend ervaren, dan kan dit de doelmatigheid van de factory beïnvloeden.
Voordat de geplande metingen en observaties zijn uitgevoerd, is gekeken naar de software-professionals in de factory.

6.1 De softwareprofessional

In dit rapport wordt het woord softwareprofessional en professional gebruikt voor hetzelfde fenomeen. Elke medewerker die werkt aan software binnen een professionele organisatie als het Product Centre wordt in dit onderzoek een ‘softwareprofessional’ genoemd. In werkelijkheid zal dit echter niet één homogene groep zijn. Als men kijkt naar het standaard factoryproces zullen verschillende professionals hier verschillend mee om gaan.

> Een stelling die bij dit onderzoek een rol speelt:
> Goede professionals willen zelf invulling geven aan hun (standaard) werkwijze. Ze zullen daarom een standaard werkwijze binnen de factory sneller als knellend ervaren dan gemiddelde professionals.

Als er een verschil is tussen hoe de goede professionals in de organisatie het factoryproces ervaren en de hoe gemiddelde professionals dit doen, dan zal dit van invloed zijn op de doelmatigheid van het proces, omdat beide groepen hetzelfde proces dienen te hanteren in de factory.

6.2 Het groeperen van de professionals

Er is een instrument geselecteerd waarmee de professionals kunnen worden ingedeeld in de groepen goed / gemiddeld. Welke manieren zijn er om professionals te karakteriseren, en welke manier zal de eventuele verschillen in softwareproces het beste zichtbaar maken tijdens de observaties en metingen?

Er zijn verschillende mogelijkheden om professionals te classificeren. De Meyers Briggs Type Indicator (MBTI), competentie vragenlijsten en capaciteiten testen zijn algemeen toepasbare testen. Testen specifiek gericht op verschillen tussen software engineers zijn er een stuk minder beschikbaar. Dit wordt bevestigd door onderzoeksinstituut NOA, dat verbonden is aan de Vrije Universiteit en gespecialiseerd is in de ontwikkeling van meetinstrumenten rond Human Resources. Op de momenten in het onderzoek waar andere vakgebieden geraakt worden, zoals in dit geval Human Resources, zijn experts betrokken bij dit onderzoek voor validatie van de aanpak.

De MBTI is een veel gebruikt instrument dat regelmatig terug komt in onderzoek naar software professionals. Het meest voorkomende MBTI type is Introversion, Sensing, Thinking, Judging (ISTJ). De andere persoonlijkheidstypen komen ook voor, maar in mindere mate (Capretz, 2002). Welke combinatie van MBTI typen een succesvolle softwareprofessional maakt, is echter nog niet aangetoond. De onderzoeken bij softwareprofessionals op basis van MBTI geven wisselende signalen. MBTI kan een indicatie geven, maar is niet geschikt als test (Kerth, 1998).

Een andere manier om software professionals te groeperen is te kijken naar de competenties. In ´Essential Competencies of Exceptional Professional Software Engineers´ (1995) onderzoekt Turley de verschillen tussen exceptionele en niet-exceptionele software professionals. In het artikel heeft Turley per softwareprofessional 38 competenties geïdentificeerd. 9 van de competenties uit zijn onderzoek bleken een significante relatie te hebben met exceptionele of niet-exceptionele professionals. De volgende competenties hadden een significante relatie, en zijn tevens de definities die gebruikt worden in dit onderzoek:

Exceptionele software engineers	Niet-exceptionele software engineers
1. Mastery of skills and techniques	1. Driven by desire to contribute
2. Maintains 'big picture view'	2. Responds to schedule pressure by sacrificing parts of the design process
3. Exhibits & articulates strong beliefs and convictions	3. Seeks help
4. Proactive role with management	4. Willingness to confront others
5. Helps Others	

Tabel 4: Exceptionele en niet-exceptionele software engineers

De indeling in exceptionele en niet-exceptionele professionals van Turley heeft als voordeel dat deze specifiek voor softwareprofessionals gemaakt is. Het NOA heeft aangegeven dat, indien een test is toegespitst op de eigen omgeving van het subject, de antwoorden van deelnemers betrouwbaarder zijn. Er is besloten om de indeling van Turley te benutten in het vervolg van het onderzoek. In de volgende paragraaf is meer informatie opgenomen over de details van deze indeling.

6.3 Details van de Q-sort

Uit de vorige paragraaf blijkt dat de indeling van Turley bruikbaar is voor het onderzoek. De developers in de factory waarvan het proces bestudeerd wordt, zijn ingedeeld in groepen exceptioneel en niet-exceptioneel. De volgende belangrijke beslissing was met welk instrument deze indeling gerealiseerd zou worden. Het letterlijk uitvragen van de door Turley gevonden competenties in een gesprek is niet voldoende betrouwbaar, omdat het risico op sociaal wenselijke antwoorden dan waarschijnlijk te groot wordt. Daarnaast ontstaat er mogelijk verschil in wat men precies verstaat onder de genoemde competenties.

Turley heeft de details van zijn onderzoek, waaronder de set kaartjes en zijn meetmethode, beschikbaar gesteld voor dit onderzoek. Daardoor kon zijn instrument gebruikt worden voor het indelen van de professionals in de groepen exceptioneel en niet-exceptioneel. Het meetinstrument is

als volgt opgebouwd: De competenties zijn uitgewerkt op 38 kaartjes met een definitie en een aantal gedragingen die de competentie het beste omschrijven. Zeven software engineers, uit twee teams van het Product Centre werden gevraagd deze kaartjes te sorteren, naar de mate waarin ze de kaartjes het meeste bij hun eigen gedrag vonden passen. Hierbij is, net zoals door Turley (1995), gebruik gemaakt van de Q-Sort, waarbij software professionals de kaartjes sorteren in zeven stapels. De Q-Sort is een onderzoeksmethode die in 1953 door de Psycholoog William Stephenson is ontdekt. Hierbij moeten de zeven stapels een normaalverdeling hebben. Dit wil zeggen, op de verschillende stapels mogen respectievelijk 2,4,7,12,7,4,2 kaartjes liggen. Kaartjes die de software professional het meest bij zich vind passen liggen helemaal links (maximaal 2), en kaartjes die het minst passen liggen helemaal rechts (maximaal 2). De software engineer zal dus een afweging moeten maken tussen welk gedrag echt bij hem past, en welke dingen minder passen.

DEFINITION
I proactively seek the assistance of others in learning, researching, designing, understanding, debugging, or checking results.

KEY BEHAVIORS

- ✓ I ask previous implementers to explain their designs.
- ✓ I ask other engineers to critique or evaluate my designs.
- ✓ I survey others to create lists of alternatives.

Item #2

DEFINITION
He spends a significant amount of time assisting others in the completion of their tasks or influencing broad organizational direction.

KEY BEHAVIORS

- ✓ He acts as a lab-wide consultant for process or product issues.
- ✓ He reviews, directs, or influences the work of other engineers.
- ✓ He assists other engineers with their tasks in an effort to complete the project.
- ✓ He teaches engineering skills to other engineers.

Item #3a

Figuur 10: Voorbeeld van twee competentie kaarten

Negen competenties zijn volgens het onderzoek van Turley gerelateerd aan exceptionele en niet-exceptionele software engineers. Op basis van de 9 bijbehorende kaarten worden de professionals ingedeeld in de groepen 'exceptioneel' en 'niet-exceptioneel'.

Naast de Q-Sort met 38 kaartjes is er voor dit onderzoek een vereenvoudigde Q-Sort voorbereid met slechts de 9 belangrijkste kaartjes (met een significante relatie voor exceptioneel of niet-exceptioneel zijn), herschreven naar derde persoonsvorm (zie Figuur 10). Deze Q-Sort in de derde persoon is gebruikt om de invalshoek van managers en teamleden te verwerken in de beoordeling, om de betrouwbaarheid te verhogen.

In dit onderzoek is de Q-Sort van Turley gedaan door de 7 professionals van BedrijfX. Voor elk van deze 7 professionals heeft hun manager en een teamlid de beperkte Q-Sort over hen ingevuld. Deze drie visies over de gedragingen en competenties van één persoon worden per competentie vergeleken. Indien de visies teveel uit een lopen, dan is de beoordeling van de persoon niet betrouwbaar, en wordt die competentie niet meegewogen in de bepaling van de exceptionele of niet-exceptionele groepen. Er is gekozen om bij een variantie die groter is dan 2,5, de gescoorde competentie niet mee te rekenen en te laten vervallen. In dat geval is de meting niet betrouwbaar, omdat de verschillende bronnen teveel van visie verschillen.

In totaal zijn van de 72 gemeten waarden die van invloed zijn op de meting er 5 niet meegeteld, omdat deze een variantie gaven van meer dan 2,5.
In onderstaande lijst is de positie van de competentiekaart in de ordening vertaald naar een score van -3 tot 3. Bij de beperkte Q-Sort is dit van -2 tot 2. Deze lijst toont de 9 competenties van exceptionele en niet-exceptionele software engineers, zoals deze voor één professional zijn ingevuld.

#	Competency	Zelf	Team	Manag.	Var.
2	Seeks help from others	3	2	1	1
3	Helps others	-2	-2	-1	0,33
16	Responds to schedule pressure by sacrificing parts of design process	-3	0	2	6,33
20	Proactively attempts to influence project direction by influencing management	0	0	0	0
21	Driven by desire to contribute	-2	1	1	3
25	Exhibits and articulates strong beliefs and convictions	-3	0	0	3
27	Willingness to Confront Others	1	-1	0	1
29	Mastery of skills and techniques	0	-1	-1	0,33
38	Maintains ´big picture´ view	-1	1	-2	2,33

Tabel 5: Voorbeeld van de uitkomst van een Q-Sort van professional #1, er is triangulatie op basis van 3 bronnen zichtbaar.

6.4 Resultaten van het bestuderen van de professionals

Het resultaat van het uitvoeren van de 7 Q-Sorts door de professionals over zichzelf, de 7 Q-Sorts over een teamlid, en de 7 Q-Sorts door de projectmanager van de professional hebben geresulteerd in twee groepen. Eén groep van 3 exceptionele software engineers, en een groep van 4 niet-exceptionele software engineers. Hier is als voorwaarde genomen dat een exceptionele professional meer dan +1 moet scoren op de competenties van exceptionele engineers en minder dan -1 op de competenties van niet-exceptionele engineers.

6.4.1 Exceptionele groep

Persoon	Score op E-Competenties	Score op NE-Competenties
Professional #2	7,0	-3,0
Professional #20	8,0	-1,0
Professional #21	2,0	-3,0

Tabel 6: De exceptionele groep

6.4.2 Niet-exceptionele groep

Persoon	Score op E-Competenties	Score op NE-Competenties
Professional #1	-3,0	4,0
Professional #3	-3,0	4,0
Professional #4	-3.0	1,0
Professional #18	-2.0	2.0

Tabel 7: De niet-exceptionele groep

Los van de indeling met behulp van de Q-Sort zijn ook de twee projectmanagers van de teams gevraagd een inschatting te maken van welke professionals de exceptionele professionals in het team zijn. De inschatting van de managers kwamen overeen met de uitkomst van de Q-Sort.

6.5 Aandacht voor vertekening bij deze meting

De selectie van de professionals die mee hebben gedaan aan de beschreven meting is gedaan door de projectmanagers binnen het Product Centre die bereid waren mee te werken aan het onderzoek. Zij hebben een aantal professionals aangewezen voor het onderzoek. Hierbij is ervoor gezorgd dat er mensen met verschillende kwaliteiten en ervaringen meededen. Door deze selectiewijze is er geen invloed vanuit de professionals zelf (de professionals kiezen niet zelf of ze meedoen aan het onderzoek).

Een vertekening die wel gesignaleerd is, is dat managers met een team met zeer hoge werkdruk minder snel mee doen aan het onderzoek. Alle teams hebben echter op verschillende momenten te maken met gestegen werkdruk, ook de teams waarbij de meting is gedaan. Verwacht wordt dat daarom het effect van deze selectie op de resultaten beperkt zal zijn.

6.6 Samenvatting

De professionals die deelnamen aan het onderzoek zijn ingedeeld in twee verschillende groepen: exceptioneel (3 professionals) en niet-exceptioneel (4 professionals). Dit is gedaan op basis van de Q-Sort van Turley (1995). Van de competenties heeft Turley bepaald welke bij de exceptionele groep hoort, en welke bij de niet-exceptionele groep. De resultaten die bij de andere metingen en observaties in het onderzoek worden behaald, zullen worden bekeken in het licht van deze twee groepen. Ervaren deze twee groepen professionals het proces binnen de factory verschillend? Hoe ziet de procesgang van deze groepen er uit?

7 Het proces zichtbaar maken

Om inzicht te verkrijgen in de factoren die de doelmatigheid van het softwareproces in een softwarefactory beïnvloeden, is het nodig het factoryproces zichtbaar te maken. Er is voor gekozen hierbij meerdere bronnen te raadplegen en verschillende onderzoekstechnieken te gebruiken (triangulatie).
Het proces is op de volgende manieren bekeken:

- Een detailwaarneming van het proces bij drie development teams
- Een globale bestudering van het proces bij twee development teams
- Het bespreken van het proces met een aantal softwareprofessionals

7.1 Het proces in detail bekijken

7.1.1 Meetwijze

Om het softwareproces van een professional te bestuderen is het bekijken van de procesbeschrijving niet voldoende. Hierin staat slechts beschreven hoe het proces er uit zou moeten of kunnen zien (*prescriptive*) (Scacchi, 2001). Om de werkelijke procesgang in kaart te brengen is het nodig de werkwijze in de praktijk te bestuderen (*descriptive*). Er is voor gekozen om de software professional bij de constructiefase (het daadwerkelijk programmeren van de applicatie) te volgen. Er zijn verschillende manieren overwogen voor het registreren van de werkwijze van een professional. Een van de mogelijkheden is observatie, bijvoorbeeld aan het bureau van de professional. Deze methode kan echter de werkwijze van de professional beïnvloeden, omdat de professional zich erg bewust is van de onderzoeker. De professional kan daardoor meer ´gewenst´ gedrag gaan vertonen of met meer precisie te werk gaan dan normaal gesproken het geval is.

Een andere optie is het gebruik van een screenrecorder. Een screenrecorder op de computer van de professional zou de handelingen van de professional op het scherm kunnen vastleggen, maar heeft het nadeel dat deze methode alleen vastlegt *wat* de professional doet, en niet *waarom.* Dit laatste is echter voor het onderzoek juist erg interessant. Daarnaast beperkt de screenrecorder methode zich tot het registreren van de computerhandelingen, terwijl de handelingen die niet met de computer worden uitgevoerd voor het onderzoek ook inzichtelijk moeten worden. Het gaat dan bijvoorbeeld om overleg tussen professionals of het doorlezen van bepaalde documentatie.

Er is voor gekozen om de professionals zelf te laten bijhouden welke activiteiten ze hebben ontplooid tijdens een deel van het project. De manier van registreren is bewust zo open mogelijk gehouden, hierdoor wordt de meting zo min mogelijk beïnvloed. Om ervoor te zorgen dat er geen waardevolle informatie wordt vergeten is gekozen voor een registratie-interval van 2 uur. Elke 2 uur werd van de professional gevraagd om de taken op te schrijven die de professional had uitgevoerd. Daarnaast werd gevraagd om bij elke taak te registreren hoeveel tijd hieraan was besteed. Met deze tijdregistratie kan zichtbaar worden hoe de taken zich tot elkaar verhouden qua aandacht en inspanning in het project.
Bij het registratieformulier was een voorbeeldregistratie opgenomen om een indicatie te geven van het detailniveau dat voor het onderzoek gewenst was.

De registratie had, gezien de beschikbare tijd van de professionals voor het onderzoek, een tijdsspanne van maximaal 2 dagen.

Vier professionals werd gevraagd deze registratie bij te houden. Hierbij bleek dat professionals deze manier van registreren in de praktijk erg moeilijk vonden. Sommige professionals dachten niet aan het registreren tijdens de werkzaamheden, of registreerden te algemene gegevens zodat deze niet bruikbaar waren. Een verklaring hiervoor kan zijn dat deze methode, ook wel dagboek methode genoemd, veel van de personen in kwestie vraagt. Deze methode vraagt van mensen om tijdens hun werkzaamheden ook met regelmaat aan de registratie te denken. Daarnaast is reflexie nodig met betrekking tot de eigen werkzaamheden en kan het confronterend zijn (Kalton & Schuman, 1980).

Om ervoor te zorgen dat de geregistreerde gegevens meer bruikbaar werden is er een korte checklist gemaakt bij het registratieformulier. Deze checklist helpt de professionals bij het registreren door aan te geven bij welke stappen in het proces men kan registreren. De checklist bestond uit de onderdelen in Tabel 8.

Bij het registreren kan je denken aan de acties die je hebt uitgevoerd in de volgende fasen van het proces.
Werkwijze bepalen
Probleem begrijpen
Oplossing bedenken
Oplossing realiseren
Testen of het werkt
Testen of het werkt naar specificatie: validatie
Evaluatie

Tabel 8: Checklist die is gebruikt bij registraties.

De checklist maakte de resultaten al wat meer bruikbaar voor onderzoek, maar waren meer gestuurd dan de volledig open registraties. Bij het tweede team werd de procesgang bekeken door tijdens de werkzaamheden op regelmatige basis een gesprek te voeren met de software professional over zijn werkwijze. Deze gesprekken gaven het beste inzicht in hoe de procesgang van de professional er daadwerkelijk uitziet. Nadeel van deze manier van onderzoeken is dat het risico op beïnvloeding van de resultaten door de onderzoeker toeneemt. Dit risico is beperkt door open beginvragen te gebruiken, en vervolgens alleen door te vragen op de onderwerpen die de professional zelf ter sprake brengt. De resultaten hiervan werden genoteerd en later geanalyseerd.

Ook door professionals in een bedrijf buiten de factory werd de werkwijze geregistreerd. Dit bedrijf maakt ook maatwerk software, maar heeft geen factoryaanpak. In dit bedrijf is er geen gemeenschappelijk proces en is er veel vrijheid voor professionals om hun eigen tools, templates en werkwijze te bepalen. De registraties van de professionals van binnen de factory en de registraties van buiten de factory kunnen inzicht geven in welke voor- of nadelen beide aanpakken kunnen hebben. Deze inzichten kunnen gebruikt worden bij het identificeren van de factoren die van invloed zijn op de doelmatigheid van de factoryaanpak.

7.1.2 Resultaten

Drie teams van vier professionals hebben meegedaan aan de registraties. Eén team van vier professionals uit een bedrijf zonder factoryaanpak en twee teams van vier professionals van binnen het Product Centre. Doel van de registratie was het krijgen van inzicht over hoe de aanpak van het Product Centre en de aanpak buiten het Product Centre van elkaar verschillen. Er werden een aantal opvallende zaken zichtbaar tijdens de registratie die tijdens de bouwfase is uitgevoerd.

Bij de teams in de factory hadden de exceptionele professionals (uit de groep die is ingedeeld zoals beschreven in hoofdstuk 6) ander type werk dan de niet-exceptionele professionals. Alle developers die werden gevolgd maakten deel uit van het development team, en houden zich bezig met het bouwen aan de applicatie. De taken van de exceptionele professionals zagen er echter volledig anders uit dan de taken van de niet-exceptionele professionals. De exceptionele professionals nemen binnen het development team de positie in van architect. De taken van de exceptionele professionals bestaan uit het neerzetten en onderhouden van de architectuur, en het technisch begeleiden van de andere developers in het team. Deze taken blijken nog slechter tastbaar dan de taken van de niet-exceptionele professionals. Bij deze taken komt veel redeneren kijken, en veel kennis van de verschillende mogelijke oplossingen en hun consequenties. De niet-exceptionele professionals werken vervolgens met name aan het uitwerken van de onderdelen van de applicatie, en gaan daarbij vaak uit van het voorbeeld dat de architect heeft neergezet. Daarmee is het kwaliteitsniveau dus voor een groot gedeelte afhankelijk van de architect in het team, en sluit het factoryconcept de noodzaak van deze exceptionele professional nog niet uit.

Een andere constatering was dat de professionals van één Result team een eigen specifieke standaard hadden ontwikkeld voor de taak 'bouwen van een webservice voor project xyz'. Deze standaard was, af te lezen aan de opmerkingen van verschillende teamleden in het document, tot stand gekomen in samenwerking met meerdere teamleden. Tijdens het maken van deze standaard werd bijvoorbeeld door teamleden gevraagd: 'in welk geval wordt dit onderdeel van de standaard uitgevoerd?'. Deze '*als-dan'* informatie is vervolgens aan het standaard document toegevoegd. Het document geeft in die zin invulling aan drie gebieden:

1. Structuur (welke fasen moet je door bij het bouwen van de webservice)
2. Inhoud (hoe voer je elke stap uit)
3. Aanwijzigen over gebruik (wanneer welke stap)

Dat deze standaard is ontstaan binnen het project, is een voorbeeld van de articulatie van de factory standaarden die in paragraaf 4.2.6 is genoemd. Er is een gebied dat niet door de factorystandaard wordt ingevuld, en dus ontstaat er een eigen productstandaard die dit opvangt.
Opvallend is dat deze standaard heel dicht op de techniek zit. Er staan bijvoorbeeld naamgeving-conventies in en aanwijzigen voor het gebruik van bepaalde tooling. De standaard is heel specifiek: meer dan mogelijk is voor een factory waarin meer gebouwd wordt dan dit type webservices. De scope van deze standaard is daarmee ook beperkt. De standaard is geschikt voor één specifieke activiteit.

De softwarefactory is zichtbaar in templategebruik bij ontwerpen en in sommige tools die vanuit de factory worden gebruikt. Elk team heeft een bepaalde aanvulling hier op gemaakt, bijvoorbeeld door een specifieke code generator te gebruiken die een performancewinst oplevert. Ten slotte is de infrastructuur binnen de factory (standaard Product Centre werkplekken) zichtbaar, alleen zijn de voordelen hiervan waarschijnlijk vooral in de opstartfase van een project goed waarneembaar. De twee Product Centre projectteams die mee deden aan het onderzoek waren beiden in een reeds lopend project.

De aanpak van ontwikkelaars binnen en buiten de factory is erg gelijkend. De fasering zoals zichtbaar in Tabel 9 komt bij alle drie de teams terug. Tijdens de developmentfase lijkt met name standaard tooling en standaard infrastructuur zichtbaar verschil te maken. Gestandaardiseerde stappen die gestuurd zijn vanuit de factory zijn niet waargenomen. Het is voor te stellen dat dit een eigenschap is van de programmeer taak. In software development is over alle zaken naast het programmeren, zoals specificatie en project management meer bij de professionals duidelijk over bijbehorende formele processen.

Activiteit bepalen: Kiezen in ontwerpdocument welke feature gerealiseerd moet worden.
Begrijp ik wat er moet gebeuren?
Welke code raakt dit in het systeem?
Aanpassen van de database indien nodig
Aanpassen van het model indien nodig
Bouwen van de business logica en het model indien nodig
Testen van (een gedeelte van) de nieuwe feature
Bouwen van de GUI
Testen van het geheel lokaal; Is de lokale testomgeving nog representatief?
Testen op een testserver
Versturen van het geheel naar het broncode systeem

Tabel 9: De fasering van de bouwfase

In het team buiten de factory dat mee heeft gedaan aan de registratie kwam naar voren dat de pragmatische aanpak zonder vast proces een negatief effect kan hebben. Eén van de developers kwam er tijdens de registratie achter dat hij de feature die hij aan het bouwen was onderschat had. Hij had aan het begin van de werkzaamheden de feature als vrij eenvoudig ingeschat. Daarom had hij besloten geen aandacht te besteden aan technisch ontwerp of het bedenken van de oplossing. Vervolgens was hij direct gestart met bouwen, maar liep tegen problemen aan die met een goede voorbereiding mogelijk voorkomen hadden kunnen worden.

In de factory werd hergebruik zichtbaar op twee niveaus. Bij het bouwen van de applicatie wordt gekeken naar andere plaatsen in de applicatie, of in eerder gemaakte applicaties en wordt waar mogelijk code overgenomen. Dit is een vorm van hergebruik op teamniveau. Ook op factoryniveau was hergebruik zichtbaar. Zo zijn er zelf ontwikkelde platforms zoals Oracle2Go en E-Platform. Deze platforms bevatten een basisraamwerk waarmee men snel nieuwe applicaties kan opbouwen.

7.2 Het proces van een afstand observeren

7.2.1 Meetwijze

De beperkte tijdsspanne van de gedetailleerde waarneming die beschreven is in de vorige paragraaf zou ervoor kunnen zorgen dat sommige interessante ontwikkelingen op procesniveau niet worden geregistreerd. Het is denkbaar dat een gedeelte van de belangrijke verschillen tussen de factoryaanpak en de niet-factoryaanpak vooral zichtbaar wordt in uitzonderlijke gevallen. Daarom is besloten de Critical Incident Technique [CIT] toe te passen. Deze techniek is een geschikt hulpmiddel voor kwalitatief onderzoek waarbij men de complexiteit van de werksituatie zichtbaar wil maken (Stitt-Gohdes, Lambrecht, & Redmann, 2000).

Er zijn twee belangrijke principes voor de CIT. Ten eerste is bij dit type analyses het rapporteren van feiten te prefereren boven het rapporteren van interpretaties, waarderingen of meningen. Het tweede principe is dat rapporten het beste kunnen gaan over zaken die in de ogen van de (kundige) observator belangrijk zijn. Bij het bestuderen van een bepaalde activiteit, worden alleen de incidenten eruit gelicht die volgens de observator een belangrijke invloed hebben op de geobserveerde activiteit. Samengevat is de methode gericht op het rapporteren van feiten die een grote invloed hebben. De techniek is gedurende 2 maanden toegepast bij observaties bij twee teams van het Product Centre bij BedrijfX.

Bij de CIT wordt een activiteit geanalyseerd en wordt in het bijzonder gelet op incidentele gevallen die de complexiteit van de activiteit zichtbaar maken. De techniek bestaat uit de volgende stappen (Flanagan, 1954):

- Bepalen van het doel van de analyse
- Plannen ontwikkelen voor het inwinnen van feitelijke incidenten rondom de activiteit
- Inwinnen van de gegevens
- Analyseren van de gegevens

Bepalen van het doel van de analyse

Het doel van de analyse was het verkrijgen van een breder beeld van de complexiteit van het softwareproces, als aanvulling op de gedetailleerde observatie die eerder was uitgevoerd. Het type incidenten waar tijdens dit gedeelte van het onderzoek op werd gelet is niet vooraf vastgesteld. De analyse is gericht op de developmentactiviteiten. Hieronder wordt de programmeertaak verstaan, maar ook bijbehorende communicatie en documentatie.

Plannen ontwikkelen voor het inwinnen van feitelijke incidenten

Bij het doen van de waarnemingen met de CIT is de observator de persoon die incidenten identificeert, en beoordeelt. De observator bepaalt of een incident wel of niet van belang is voor het onderzoek. Daarom is van belang dat de observator zelf voldoende kennis heeft van de activiteit die bestudeerd wordt. In dit onderzoek zijn de observaties door de onderzoeker zelf uitgevoerd, omdat de onderzoeker zelf kennis heeft van de developmentactiviteit en incidenten kan identificeren. De onderzoeker heeft 2 maanden het developmentproces gevolgd, door de activiteiten van de developers te volgen en deel te nemen in vergaderingen.

Inwinnen en analyseren van de gegevens

De resultaten van de uitgevoerde analyse zijn opgenomen in de volgende paragraaf.

7.2.2 Resultaten

Tijdens de periode van observatie werd een relatief groot project opgestart, waarin een van de teams die gevolgd werd deelnam. Dit project was groter dan de andere projecten die normaal gesproken door het team werden uitgevoerd, en daarom stelde dit extra eisen aan het proces. De uitgangspunten in het kader (zie hoofdstuk 2) sluiten hier op aan. Het team moet samen werken met een aantal andere teams, en daarom werd ook gedeelde tooling en werkprocessen over meerdere teams van toepassing. Op dit punt zou het Product Centre concept uitkomst moeten bieden, doordat de werkwijze van de verschillende teams en de gebruikte tools overeen komen en op elkaar aansluiten.

Het team dat meewerkte aan het onderzoek bleek echter slechte ervaringen te hebben met één factorytool: het broncode managementsysteem. Deze tool bleek in het verleden geselecteerd voor de factory en succesvol, maar tegenwoordig voldeed dit systeem niet meer aan de eisen van het ontwikkelteam. Deze constatering zorgde voor een dilemma: moet op dit punt de factorystandaard gehanteerd worden, zodat het uitwisselen van code met andere teams gemakkelijker gaat, of moet er een eigen standaard gekozen worden, zodat het team zelf beter presteert?

Professionals buiten het team stonden achter gebruik van de standaard, maar het team zelf vond deze tool niet werkbaar. In eerste instantie werd na overleg gebruik van de standaard tool gehandhaafd. Er werden nieuwe werkafspraken gemaakt over hoe men om de beperkingen van de tool heen kon werken. Na enkele dagen werken met de tool bleek de performance van de tool zeer slecht.
Transacties met het broncode managementsysteem namen excessief veel tijd in beslag. Na discussie en overleg werd besloten dat het team toch het zelf gekozen broncode managementsysteem voor dit project kan gebruiken, en dat de standaard in dit geval zal wijken.

Dit incident, dat tijdens de observatie is waargenomen, is een illustratie van een vraagstuk dat een softwarefactory altijd heeft: ‘Wanneer verkies ik een andere tool boven de standaard tool’, of ‘Wanneer verkies ik een andere werkwijze boven de standaard werkwijze?’. Hoe de organisatie met dit soort vragen omgaat lijkt relevant voor de doelmatigheid van het softwarefactory proces. Een standaard tool of werkwijze waar men te lang aan vast houdt, zou kunnen zorgen voor een suboptimaal functioneren. Maar ook een differentiatie in de werkwijze voor elk project heeft negatieve effecten, en zal beperkt moeten worden.

7.3 Het proces bespreken

7.3.1 Meetwijze

Er zijn open interviews gehouden met verschillende medewerkers over het softwareproces in het Product Centre. Bij alle interviews werd een introductie op het onderwerp gegeven en werd vervolgens doorgevraagd op de onderwerpen die de geïnterviewden opwierpen. Er zijn gesprekken gevoerd met twee competence managers, drie projectmanagers en zeven developers.

7.3.2 Resultaten

Standaardisatie

In alle gesprekken, zowel bij managers als bij developers kwam naar voren dat standaardisatie en een factoryaanpak belangrijk is. Er werden verschillende redenen genoemd voor standaardisatie. Een van de redenen die genoemd werd door projectmanagers was het kostenaspect. "*Je kunt niet meer mee doen zonder standaardisatie*". De managers verwachten een kostenbesparing door het gebruik van standaardisatie. "*Mensen hoeven dan niet meer opnieuw het wiel uit te vinden*". Een andere reden die ter sprake kwam is het uitbesteden van werk aan het Product Centre in India. Een projectmanager gaf aan dat de overdracht naar het andere Product Centre alleen lukt indien er volgens een vast stramien wordt samengewerkt. De standaardisatie wordt ook genoemd als belangrijke voorwaarde voor CMMi certificering. Het Product Centre is momenteel bezig met het behalen van CMMi level 3. Een vastgelegde werkwijze die ook door de teams wordt gevolgd is hiervoor een vereiste.

Ook groei van de Product Centres vraagt om een gestandaardiseerde en gedocumenteerde werkwijze. Projectmanager: "*In andere vestigingen in het land wil men ook een (bepaald type) ontwikkelstraat neerzetten. De andere locaties vragen dan aan ons hoe wij het aanpakken. Het is dan van belang dat we weten hoe onze aanpak er uit ziet, en dat deze zo gedocumenteerd is dat deze overgedragen kan worden*." Bij de beschrijving van een werkwijze binnen één straat blijken gebruiksinstructies van belang. "*Om de overdracht van werkwijze naar andere vestigingen te vergemakkelijken is er iemand aangetrokken die een development case heeft opgesteld. (…) De development case is een document waar het proces voor een specifiek project wordt uitgewerkt en bepaalde practices worden geselecteerd."* In dit document was echter onvoldoende zichtbaar wanneer bepaalde acties gedaan moeten worden: als er een klein project wordt uitgevoerd, welke onderdelen van de standaard werkwijze zijn dan essentieel?

Conformatie aan de standaard

Het gebruik van de standaard werkwijze is volgens de perceptie van de managers per team erg verschillend. Volgens een projectmanager is dit vooral afhankelijk van de ervaring die de mensen hebben binnen het Product Centre. De meer ervaren developers, die langer binnen Result werken, passen over het algemeen meer de standaard werkwijze toe. Deze mensen zijn ook meer op de hoogte hoe de standaard werkwijze er uit ziet, en waar ze alle details ervan kunnen vinden.

"*Individuele afwijkingen van de aanpak of tooling komt nog wel voor, maar dit is iets waar we vanuit projectmanagement echt van af willen*." Tegen eigen werkwijzen en tools zijn buiten de efficiëntie en kwaliteitsvraagstukken ook hele praktische bezwaren. "*Als iemand uit het team ziek is, is zijn afwijkende werkwijze of toolgebruik erg verstorend. (…) Specifieke wachtwoorden van tools of kennis*

van de aanpak ontbreken in die gevallen." Als dit vlak voor een oplevering gebeurt, heeft het een grote impact. Eigen gekozen tools of werkwijzen leveren dus projectrisico's op.

Het komt ook voor dat het de wens van de klant is om af te wijken van de standaard werkwijze, geeft een projectmanager aan. Een krappe deadline zorgt er dan voor dat bepaalde ontwerp- of specificatiestappen verkort worden uitgevoerd. Dit is een dilemma: helpt men de klant het beste door in te spelen op hun behoeften, of door bepaalde kwaliteitsstandaarden te handhaven?

Tools

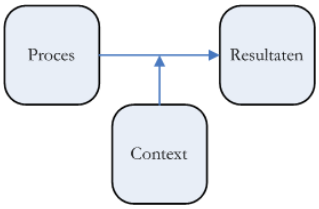
Bij de gesprekken met de developers lag de nadruk met name op tooling. Het belang van een standaard werkwijze werd erkend, en het belang van goede en op elkaar aansluitende tooling die deze standaard werkwijze ondersteunt werd door diverse developers onderstreept. "*Het belang van goede toolings is heel groot*", gaf een developer aan. Verschillende overwegingen en ideeën met betrekking tot tooling kwamen ter sprake.

Een belangrijke keuze die de tooling in de factory betreft, is of men kiest voor één set tools over alle ontwikkelstraten heen, of voor differentiatie van tools per ontwikkelstraat. Binnen het Product Centre wordt momenteel één toolselectie van over alle straten gehanteerd. Developers geven in de gesprekken argumenten aan ten gunste van deze keuze: minder training bij het wisselen van medewerkers tussen de straten, en een betere en gemakkelijkere samenwerking tussen straten doordat dezelfde tools gebruikt worden. Er komen echter ook argumenten tegen deze beslissing naar voren. Een van de developers geeft aan dat de tools die gekozen zijn niet optimaal zijn binnen elke straat, en komt met een andere visie op het probleem. "*Je zou de tooling moeten selecteren die het development werk het beste ondersteunt. Op dit stuk in de ontwikkelcyclus kan het meeste winst behaald worden door gebruik van optimale tooling. Vervolgens dienen de tools in de andere schakels van de ontwikkelcyclus, zoals de analysefase en de testfase, hierop afgestemd te worden. (…) De voordelen die je hiermee behaalt in termen van traceability, kwaliteit en ontwikkelsnelheid zijn belangrijker dan het trainingsvoordeel dat je behaalt door de keuze voor één toolset*."

Hoewel de werkwijze en taakbeschrijvingen in de factory een langere tijd gelijk kunnen blijven, is dit bij tools een ander verhaal, vermeldt een developer. Elke 2 jaar komt er bijvoorbeeld een nieuwe versie van de gebruikte IDE uit. We zien tools of werkwijzen verouderen en er moeten keuzes gemaakt worden over hoe hiermee om wordt gegaan in de factory. Op dit moment kunnen developers die onderhoud uitvoeren op de factory, deze uren nog niet verantwoorden. Er is geen speciaal budget voor individuele teams om de softwarefactory te onderhouden.

8 Effecten van het softwareproces op het resultaat

Zoals eerder beschreven kan het bestuderen van het proces direct, door het volgen en bestuderen van de procesgang bij projecten. De resultaten hiervan zijn beschreven in het vorige hoofdstuk. Het is ook mogelijk om, wanneer het proces is uitgevoerd, te kijken naar de resultaten die dit heeft opgeleverd. In deze resultaten kunnen namelijk indicaties worden gevonden voor bepaalde zaken die in het proces verbeterd kan worden.



Figuur 11

Een voorbeeld: het is voor te stellen dat bij de observatie van het proces geen opvallende zaken worden gevonden, maar dat in het uiteindelijke softwareproduct excessief veel bugs zitten. Dit laatste geeft dan een indicatie dat er bij het maken van de software iets niet optimaal is verlopen. In dit hoofdstuk wordt gekeken naar resultaten op drie gebieden:

- Efficiëntie-effecten
- Kwaliteitseffecten
- Effect op de softwareprofessionals

8.1 Efficiëntie-effecten

8.1.1 Meetwijze

Om effecten van de factoryaanpak op de efficiëntie van een team af te tasten, is er een meting gedaan bij de programmeerfase van twee projecten. Eén project binnen het Product Centre, en één project in een organisatie buiten het Product Centre. Het project buiten het Product Centre is volledig vrij ingericht en de ontwikkelaars kiezen zelf de gebruikte tools of werkwijze.

Zoals eerder beschreven is naast het proces ook de context van invloed op het resultaat. Als de context van de twee projecten niet gelijk is, kunnen resultaten niet zomaar vergeleken worden. Voor de meting zijn twee projecten gekozen die reeds lopen, het gaat in beide gevallen om systemen die al in productie genomen zijn, en waarbij in een project van rond de 210 uur inspanning een stuk functionaliteit wordt aangebouwd aan een bestaand systeem. Beide projecten zijn daarnaast gebouwd in een webomgeving met een 4GL taal. In alle twee de projecten werd gebruik gemaakt van code generatoren, de hoeveelheid gegenereerde code is apart geteld in de efficiëntiemeting. De twee projecten werden beiden uitgevoerd door een team van vier professionals. Hoewel bij het kiezen van de twee projecten zoveel mogelijk contextfactoren gelijk zijn gekozen, is het niet haalbaar alle verschillen in omgevingsfactoren uit te sluiten.

De geproduceerde resultaten worden meetbaar gemaakt door het gebruik van functiepunten, en regels code. De functiepunten zijn geteld volgens de specificaties van de Nederlandse Software Metrieken Gebruikers Associatie (NESMA, 1996), door een professional die ervaring heeft met het doen van dergelijke tellingen. Hierbij zijn de ontwerpdocumenten, het datamodel en de screenshots van de uiteindelijke applicaties als input gebruikt voor de telling. De telling van de regels code is uitgevoerd met behulp van de tool SLOCCount, door Wheelers. Het programma telt de fysieke SLOC, hiermee worden alle regels broncode bedoeld die eindigen op een regeleinde, die niet leeg zijn, en die niet alleen uit commentaar bestaan.

Ten slotte is ook het aantal uren dat gebruikt en begroot is opgenomen in deze meting. Deze gegevens zijn verstrekt door de projectmanagers van de projecten, en komen uit de tijdschrijfsystemen die tijdens het project zijn gebruikt.

8.1.2 Resultaten

	Uren begroot	**Uren gebruikt**	**SLOC toename totaal**	**SLOC toename handmatig**	**SLOC gegenereerd**	**Netto functiepunten**	**Bruto functiepunten**	**Uren per FP**	**SLOC per uur**	**SLOC per FP**
Product Centre, Project X	211	298,5	13412	4209	9203	32	40	9,3	14	132
Buiten Product Centre, Project Y	208	206	11024	5318	5706	54	61	3,81	27	98
Verschil in %	-1,4 %	-31,0%	-17,8%	26,3%	-38,0%	68,8%	52,5%	-59,0%	92,9%	-25,8%

Tabel 10: Resultaten performance-effecten

In de resultaten van deze meting is zichtbaar dat het aantal uren per functiepunt buiten de factory 59% lager ligt, dan binnen de factory. Een (gedeeltelijke) verklaring hiervoor kan zijn dat de voordelen van een softwarefactory vooral aan het begin en einde van het project zichtbaar worden. Omdat deze meting is uitgevoerd bij een lopend project, zijn mogelijke voordelen in de opstarttijd voor een project in de softwarefactory niet zichtbaar. Ook is in de factory een aanzienlijke overschrijding van de uren zichtbaar ten opzichte van wat was begroot. Dit is besproken met het projectteam, en is veroorzaakt door extra ondersteuning bij de klant en een onderschatting van de complexiteit. Deze overschrijding is opmerkelijk voor een factoryproject, omdat uiteindelijk juist voorspelbaarheid ten doel is gesteld.

De resultaten uit dit hoofdstuk worden geïnterpreteerd in hoofdstuk 9.

8.2 Kwaliteitseffecten

8.2.1 Meetwijze

Het proces dat gevolgd is zal ook van invloed zijn op de kwaliteit van de geproduceerde software. Daarom is bij de twee projecten ook gekeken naar de fouten die in de software worden gevonden, tijdens de integratietest (door een tester) en tijdens de acceptatietest (in samenwerking met de klant). De testgegevens komen uit het bugtrackingsysteem dat bij beide projecten is gebruikt. Configuratiefouten, wijzigingsaanvragen en fouten die tijdens developertests zijn gevonden zijn niet opgenomen in de meting.

8.2.2 Resultaten

	SLOC totaal	SLOC toename totaal	SLOC toename handmatig	SLOC gegenereerd	FP geproduceerd	Bugs bij integratietest	Bugs bij integratie + acceptatietest	Bugs per FP	Bugs per 1000 SLOC
Product Centre, Project X	267437	13412	4209	9203	32	4	2	0,1875	1,425
Buiten Product Centre, Project Y	118415	11024	5318	5706	54	13	5	0,3333	3,384
Verschil in %	-55,7%	-17,8%	26,3%	-38,0%	68,8%	225,0%	150,0%	77,8%	137,5%

Tabel 11: Resultaten kwaliteitseffecten

In de meting die gedaan is bij deze twee projecten, is zichtbaar dat er met de softwarefactory aanpak 77% minder fouten in de software zijn gevonden dan bij de aanpak buiten de factory. Per 1000 regels code, zijn er buiten de factory 137% meer fouten gevonden dan binnen de factory. Dit verschil in fouten is interessant, en daarom is specifieker gekeken naar de soorten fouten die optreden.

Bij project X, in de factory, bestaan de fouten die gevonden zijn bij de integratietest met name uit validatie fouten van invoer. De acceptatietest bracht één algoritmefout aan de orde (het systeem voert een actie uit zonder dat dit gewenst is) en één failure waardoor het systeem niet verder kan gaan met de actie die het aan het uitvoeren is. Bij Project Y, buiten de factory, werd tijdens de integratietest 5 fatale fouten geconstateerd. Bij deze fouten stopt het systeem en kan het de actie niet afmaken. Daarnaast werden 6 fouten gevonden waarbij het systeem niet werkte zoals verwacht. Ten slotte werden 2 fouten gevonden die met compatibiliteit met de client browser te maken hadden. Tijdens de acceptatietest waren er 2 fatale fouten, en 3 fouten waarbij het systeem niet werkt als verwacht, maar zonder foutmelding.

De resultaten uit dit hoofdstuk worden geïnterpreteerd in hoofdstuk 9.

8.3 Effect op de software professionals

8.3.1 Meetmethode

Het softwareproces in de factory zal ook effect hebben op de professionals die het werk uitvoeren. De professionals ervaren het softwareproces, en hun visie op het proces is waardevol voor het onderzoek. Eventuele nadelige of positieve effecten die de professionals ervaren, kan de doelmatigheid van het softwareproces beïnvloeden.

Besloten is om deze informatie met behulp van een anonieme vragenlijst in te winnen. De verwachting is dat mensen in een anonieme vragenlijst vrijer spreken over hun werkelijke meningen en ervaringen. Daarnaast heeft de vragenlijst als voordeel dat exact dezelfde vragen aan alle respondenten voorgelegd zijn, en dat de uitkomsten daarvan vervolgens sneller vergelijkbaar zullen zijn. Bij de professionals die de vragenlijst hebben ingevuld, was vooraf onderscheid gemaakt in de groepen 'exceptionele professional' en 'niet-exceptionele professional', zodat verschillen in antwoorden tussen deze groepen kunnen worden onderzocht. De vragenlijst bestaat uit een mix open en gesloten vragen.

De volgende vragen zijn de uitgangspunten van de vragenlijst:

- Vragen met betrekking tot sturing vanuit de software factory
    - Ervaart de professional sturing op zijn werkwijze?
    - Accepteert de professional sturing op zijn werkwijze?
    - Wat gebeurt er indien de sturing toeneemt?
- Vragen met betrekking tot de eigen invloed op de werkwijze
    - Kan de professional zijn eigen werkwijze beïnvloeden?
    - Kan de professional de factory werkwijze beïnvloeden?
- Vragen met betrekking tot prestaties van de factorywerkwijze
    - Zorgt de factory werkwijze voor betere resultaten?
    - Hoe zou de professional de factory werkwijze verbeteren?

De vragenlijst is opgebouwd uit feitenvragen, beoordeling/perceptie vragen, positief geformuleerde vragen en negatief geformuleerde vragen. De combinatie van deze verschillende soorten vragen moeten resulteren in een genuanceerd beeld van de mening van de professional, en vertekening door de vraagstelling zo veel mogelijk voorkomen. Een vraagstelling die een feit probeert te meten, maar een mening uitvraagt, moet bijvoorbeeld worden voorkomen.

Bij het maken van de vragenlijst is rekening gehouden met het onderscheid in feitenvragen en meningvragen. Hierbij is de volgende regel gebruikt om het onderscheid te bepalen tussen deze twee typen vragen: het antwoord op een feiten vraag is ook te verkrijgen vanuit een andere bron dan de ondervaagde persoon (Kalton & Schuman, 1980). Mening of perceptievragen zijn juist alleen bij de ondervraagde persoon te achterhalen.

Ook het feit of een vraag positief of negatief is geformuleerd kan de uitslag van de vragenlijst beïnvloeden (Krosnichk, 1999). Daarom zijn dezelfde hoofdvragen uitgewerkt in verschillende positief en negatief geformuleerde vragen. Naast de formulering is ook de volgorde van de vragen in de lijst van invloed op de uiteindelijke resultaten (Kalton & Schuman, 1980). In de gebruikte vragenlijst wisselen de verschillende soorten vragen (feiten, mening) elkaar af, en zijn gerelateerde vragen niet direct achter elkaar gezet.

Het is van groot belang dat de vragen voor de verschillende respondenten hetzelfde betekenen, en hetzelfde betekenen voor de onderzoeker en de respondenten (Fowler, 1992; Gendall & Hoek, 1990). Bijvoorbeeld: Wat men precies verstaat onder *´de factory´* is bijvoorbeeld van groot belang. Daarom is hiervan een uitleg gegeven aan het begin van de vragenlijst. Voor de vragenlijst is een pretest gedaan onder studenten van de studie Software Engineering. Het doel hiervan was eventueel onduidelijk taalgebruik te elimineren.

Bij gesloten vragen kunnen de antwoordmogelijkheden ook de resultaten van de vragenlijst beïnvloeden. Het werkt het beste wanneer alle antwoordopties een waarde hebben (Kitchenham & Pfleeger, 2002). In deze vragenlijst is gekozen voor ´volledig juist´,´juist´,´onjuist´,´volledig onjuist´. Tussen deze antwoordopties is geen neutrale optie opgenomen. De professionals die de vragenlijst invullen zullen een keuze moeten maken tussen ´juist´ of ´onjuist´. Dit voorkomt dat mensen de neutrale optie gaan gebruiken, terwijl ze eigenlijk ´geen mening´ bedoelen. De optie 'geen mening' is apart opgenomen, los van de andere antwoordopties.

Omdat dit een kwalitatief onderzoek is, gaat het in de vragenlijst met name om de kwaliteit van de antwoorden, en het signaleren van eventuele opvallende antwoorden. Het is nodig om te kunnen beoordelen in hoeverre de professionals nadenken over het proces. Antwoorden van professionals die een uitgesproken mening hebben over het softwareproces kunnen dan meer aandacht krijgen dan professionals waarbij dit niet het geval is. In de vragenlijst is getracht dit te bereiken door een aantal open vragen te stellen waaruit de affiniteit en visie op het softwareproces te achterhalen. Een van de vragen is als volgt:

"Deze vraag gaat over wat jij doet om goede kwaliteit programmacode te produceren, die op tijd klaar is. We stellen deze vraag aan alle deelnemers aan deze enquête. Hoe zorg je ervoor dat jij op tijd goede code oplevert?"

8.3.2 Resultaten

Dit kwalitatieve onderzoek vraagt om een andere behandeling van de resultaten uit de vragenlijst dan in kwantitatief onderzoek. In dit onderzoek:

- is extra aandacht voor de open vragen
- krijgen specifieke afwijkende waarden voor één professional aandacht in plaats van dat ze uitgefilterd worden

De vragenlijst is ingevuld door de zeven professionals, waarvoor ook de Q-sort is gedaan voor het identificeren van de exceptionele professionals.

Sturing en standaardisatie

Alle professionals die de vragenlijst hebben ingevuld vinden de sturing en standaardisatie essentieel om producten op een concurrerende wijze te produceren. Alle professionals zijn bereid om hiervoor vrijheid in te leveren. Dit bevestigt het beeld uit de gesprekken die zijn gevoerd met de professionals. Uit de standaard tools die worden gebruikt komt een standaard IDE, een standaard framework en een standaard bugtracker het meeste voor. Ook de templates van de factory worden veel gebruikt. Opvallend is dat de checklists van de softwarefactory zelden worden gebruikt. Slechts één van de professionals gebruikt deze checklists bij zijn of haar werk.

Sommige onderdelen van de factory worden als knellend ervaren. Het gebrek aan ondersteuning bij storingen in de infrastructuur wordt genoemd, en het standaard broncode beheer systeem. Er zijn ook twee professionals die geen last hebben van negatieve elementen in de softwarefactory. Ten slotte merkt één professional het toepassen met de standaard werkwijze RUP op als negatieve invloed vanuit de factory. Hierbij spreekt hij van de "RUP dwangbuis". Als uitleg wordt hierbij vermeld dat er in sommige gevallen een sfeer is waarbij je het als ontwikkelaar erg goed doet als je alle documenten netjes oplevert.

Invloed op de eigen werkwijze

Alle professionals hebben het gevoel invloed te hebben op de eigen werkwijze en die van het team. Geen van de professionals voelen zich beperkt in het gebruiken van eigen tools binnen het team, en dit gebeurt bij 5 van de 7 professionals. Eigen tools en templates die veel worden gebruikt zijn code generatoren, een eigen versiemanagement systeem en aangepaste standaard templates.

Verbetering

Alle professionals hebben bij het inleveren van de vragenlijst aangegeven ideeën te hebben voor verbetering. Daarbij kwamen versiebeheer, serverinrichting en support en kennisverzameling en overdracht voor als punten voor verbetering. Ook had een professional het voorstel om het inzicht in de meest gebruikte tools binnen de factory te verbeteren. Eén professional gaf zijn visie op de softwarefactory: "Een factory is de manier om software met een voorspelbare kwaliteit binnen een voorspelbare tijd te bouwen doordat effectief met de beschikbare resources (vooral menskracht) kan worden omgegaan (minder tijd stoppen in niet productieve zaken). Echter een software factory die (onverwacht) niet functioneert of niet wordt onderhouden zal tot uitlopen van planningen en frustraties leiden."

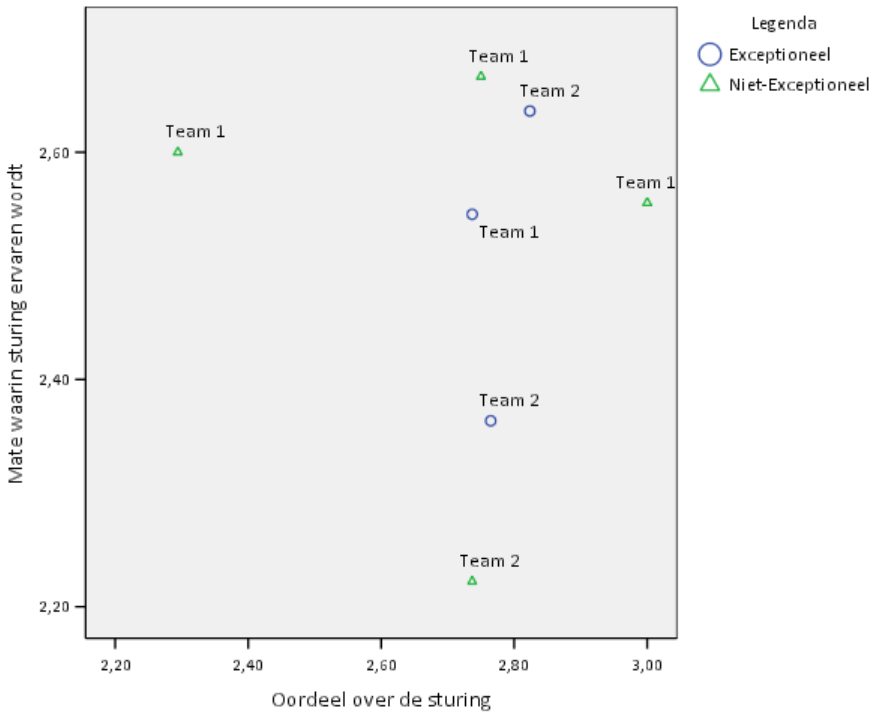
Verschil tussen exceptionele en niet-exceptionele professionals

Een van de verwachtingen bij deze meting was dat exceptionele professionals meer sturing ervaren in de softwarefactory. Dit type professional is naar verwachting het meest bezig met het verbeteren van de eigen werkwijze, en wil daarin mogelijk meer inspraak. Deze gedachte is ook terug te vinden in het werk van Fitzgerald (1997).

Van de niet-exceptionele professional is de verwachting dat het oordeel van de professional op de sturing positiever is, omdat deze professional minder behoefte heeft aan vrijheid in het bepalen van de aanpak. Om dit te kunnen beoordelen zijn de antwoorden van de exceptionele en niet-exceptionele professionals geanalyseerd. De antwoorden op de vragen zijn gegroepeerd op de volgende twee schalen:

- A .Mijn werkwijze wordt gestuurd (+), ik bepaal mijn eigen werkwijze (-)
- B. De sturing in de werkwijze is positief (+), de sturing in de werkwijze is negatief (-)

Schaal A is de samenvoeging van de antwoorden op 11 vragen en schaal B is de samenvoeging van 18 vragen uit de vragenlijst. De vragen die negatief waren geformuleerd zijn opnieuw gecodeerd, zodat deze vergelijkbaar werden met de positief geformuleerde vragen.



Figuur 12: Resultaten effecten op de professionals (hoger is meer/positiever)

De resultaten van deze meting zijn in Figuur 12 zichtbaar gemaakt. Het plaatje geeft geen bevestiging van de stelling. De exceptionele professionals waren verwacht in linkerbovenhoek van het schema, met een hoge ervaring van de sturing op de Y as, en een lage beoordeling van de sturing op de X as. De exceptionele professionals bevinden zich echter in het middel van het plaatje, en hebben geen negatief beeld van de sturing, ook al wordt deze door twee van de exceptionele professionals wel duidelijk ervaren.

De niet-exceptionele professionals zijn verspreid over het plaatje. Er zijn professionals in deze groep die de sturing sterk ervaren, maar deze niet positief beoordelen. Ook zijn er professionals die de sturing niet sterk ervaren, maar hier een positieve beoordeling aan geven. Door tevens de teamaanduiding toe te voegen aan de professionals in het plaatje, is zichtbaar dat Team 1 gezamenlijk de sturing het meeste merkt, terwijl Team 2 er gezamenlijk het beste oordeel over heeft.

9 Resultaten

In de voorgaande hoofdstukken zijn de bevindingen beschreven die tijdens dit onderzoek zijn gedaan. In dit hoofdstuk wordt op basis van deze bevindingen antwoord gegeven op de onderzoeksvragen die aan het begin van het onderzoek zijn gesteld. Vervolgens is de conclusie van het onderzoek opgenomen.

9.1 Onderzoeksvragen

9.1.1 In welke mate is het Product Centre een softwarefactory?

In dit onderzoek is een softwarefactory gedefinieerd als een andere manier van software maken, waarin hergebruik, een gestandaardiseerd proces en daardoor voorspelbaarheid een belangrijke rol spelen. De elementen hergebruik, standaardisatie en voorspelbaarheid zijn ook terug te zien in de resultaten van de observaties en metingen in het Product Centre.

Hergebruik

In het Product Centre is er sprake van hergebruik op verschillende niveaus. Ontwikkelaars kijken bij het ontwikkelen van een nieuw stuk applicatie naar hoe een probleem op een andere plaats in de applicatie is opgelost. Deze oplossing wordt dan waar mogelijk hergebruikt. Dit soort hergebruik was zowel in het Product Centre als buiten de factory zichtbaar. Een ander soort hergebruik is het opnieuw gebruiken van componenten die daar speciaal voor gemaakt zijn. Het gaat dan om (onderdelen van) applicaties die zelf ontwikkeld zijn met het doel deze bij meerdere projecten opnieuw te kunnen inzetten. Bij het Product Centre is deze vorm van hergebruik zichtbaar in de oplossingen Oracle2Go en het E-platform. Deze oplossingen zijn standaard raamwerken met standaard componenten waardoor applicaties sneller kunnen worden ontwikkeld.

Gestandaardiseerd proces

In de softwarefactory zijn twee soorten standaardisatie geconstateerd. Er is een standaardisatie op factoryniveau, en een standaardisatie op teamniveau. De standaardisatie op factoryniveau werd zichtbaar door de gezamenlijke set tools en templates die door de teams gebruikt werden tijdens de gedetailleerde observatie van een developmentactiviteit. In de mate waarin deze standaard werd toegepast waren de teams relatief vrij. Dit blijkt uit de vragenlijst die de professionals van twee teams hebben ingevuld. De professionals uit beide teams gaven aan invloed te hebben op de aanpak van het werk. Dit was ook zichtbaar in het gebruik van de standaard. Het gebruik van de standaard onderdelen van de factory bleek verschillend per team. Bij de twee teams in de factory bij het onderzoek betrokken waren, werd bijvoorbeeld de standaard IDE door alle projectleden gebruikt. Het gebruik van de beschikbare checklists uit de factory was echter nog erg beperkt, en ook het werken volgens de procesbeschrijvingen uit het Product Model was per team verschillend. Het verschillend doorvoeren van de standaardisatie, maakt dat eenzelfde project door twee verschillende teams anders kan worden aangepakt.

Tijdens het observeren van de werkwijze in detail, werden ook teamstandaarden zichtbaar. Deze standaarden werden in het team zelf gemaakt, omdat deze de efficiëntie en kwaliteit van werken in het team ten goede kwamen.

Voorspelbaarheid

Het element voorspelbaarheid komt terug in het visiedocument van het Product Centre. Voorspelbaarheid is één van de doelstellingen voor het toepassen van de factoryaanpak. Tijdens het onderzoek is gekeken naar hoe deze voorspelbaarheid terug te zien is in de praktijk bij het werken in het Product Centre.

Zoals beschreven is er verschil in de mate van standaardisatie van de verschillende teams van het Product Centre en is er sprake van *procesinconsistentie*. Dit blijkt uit gesprekken met projectmanagers, het gedetailleerd observeren van de werkwijze van twee development teams en het volgen van de developmentteams. Deze verschillende aanpak beperkt de voorspelbaarheid van de projectresultaten. Eenzelfde project kan bij twee verschillende teams op een andere wijze worden aangepakt, en daardoor tot verschillende resultaten leiden.

In het onderzoek is ook gekeken naar de resultaten die bij projecten in de factory behaald werden. Bij de performancemeting die in de factory is uitgevoerd werd een overschrijding van 41% op de geplande uren zichtbaar. In een bespreking met de projectmanager werd duidelijk dat de overschrijding is ontstaan doordat er extra ondersteuning werd geboden aan de klant (ondermeer bij de installatie) en omdat de complexiteit was onderschat. Hoewel er duidelijke redenen voor deze overschrijding zijn, geeft dit aan dat het factoryconcept dit type invloeden niet altijd ondervangt. Er zijn gevallen waarin, in de softwarefactory, bij een reeds lopend project, toch inschattingsfouten voorkomen. Hierdoor is de voorspelbaarheid in die gevallen beperkt.

Resultaat

De elementen hergebruik, standaardisatie en voorspelbaarheid zijn terug te vinden in het Product Centre. Het hergebruik is zichtbaar in de vorm van het E-platform en Oracle2Go. De standaardisatie over verschillende teams heen is ook zichtbaar, al is de mate van gebruik nog verschillend. Er zijn ten slotte gevallen waarin de voorspelbaarheid nog niet optimaal is, maar de voorspelbaarheid is wel als doelstelling opgenomen in het visiedocument van het Product Centre. De drie elementen uit de softwarefactory zijn dus vertegenwoordigd in het Product Centre, zij het in verschillende mate. Zoals in hoofdstuk 4 beschreven, is het inrichten van een softwarefactory een proces op zich, waarbij het acceptabel en realistisch is dat de factory nog in ontwikkeling is. Daarom is het Product Centre een softwarefactory, zij het met een nog beperkt doorgevoerde standaardisatie.

9.1.2 Zijn er verschillen zichtbaar tussen een factory en een niet factory software proces?

Ten eerste is het van belang vast te stellen wat we precies verstaan onder proces. Er zijn definities die het woord softwareproces zien als een ideale situatie, zoals organisaties het op papier vastleggen. In dit onderzoek is uitgegaan van een definitie die de complexiteit van het softwareproces laat zien:
"Het softwareproces bestaat uit meerdere, onderling verbonden ketens van taken en de organisatiestructuren, technologieën, procedures en artefacten die nodig zijn bij het uitvoeren van deze taken. Een keten van taken bestaat uit een niet-lineraire reeks van taken die objecten structureren en omzetten in tussen- of eindproducten. Hierbij zijn de individuele taken niet deterministisch, mogelijk iteratief en kunnen ze meerdere alternatieve vervolgtaken bevatten."

Werkwijze

Tijdens het onderzoek is er gekeken naar de verschillen die in de praktijk zichtbaar zijn tussen het softwareproces in een softwarefactory en het softwareproces in een organisatie zonder softwarefactory. Het softwareproces in een bedrijf zonder factoryaanpak en in het Product Centre is geregistreerd. De stappen die programmeurs zetten om tot resultaat te komen blijken echter bij beide omgevingen erg gelijkend. Het is mogelijk dat dit veroorzaakt wordt door het feit dat het programmeren zelf een erg ontastbare activiteit is, waarbij standaard werkwijzen of methoden vaak niet gedetailleerd beschrijven hoe dit aangepakt moet worden.

Standaardisatie

Een overeenkomst tussen de twee werkwijzen was ook dat er in beide omgevingen een vorm van standaardisatie ontstaat, zonder dat dit is vastgelegd in formele afspraken. In de softwarefactory hadden exceptionele professionals zelf templates of procesbeschrijvingen gemaakt die specifiek op een bepaald project waren toegespitst. Buiten de factory was er ook sprake van zulke standaardisatie initiatieven. Mogelijk komt dit doordat professionals uit zichzelf de neiging tot standaardisatie hebben. Deze vorm van standaardisatie kan, ten opzichte van de factory standaard, gezien worden als articulatie. Onderdelen die niet zijn bepaald door de factory standaard, worden door de teams zelf ingevuld.

De verschillen in standaardisatie die zijn waargenomen tussen de factory en niet-factory omgeving zitten met name in de standaard tooling en standaard infrastructuur van de softwarefactory. Hoewel de mate van het gebruik van de standaard tools verschilt per team, was een zeker gedeelte van de toolset en infrastructuur afkomstig uit de set van standaard Product Centre onderdelen. Een ander belangrijk verschil was de aanwezigheid van een complete procesbeschrijving in de softwarefactory. Deze beschrijving was niet aanwezig in de organisatie buiten de softwarefactory.

Vrijheid

Bij één van de stappen in het onderzoek werd zichtbaar dat de exceptionele professionals de rol van architect hebben in de softwarefactory. Dit is een andere rol dan de niet-exceptionele professionals innemen. De exceptionele professionals hebben hierdoor meer vrijheid in de werkwijze en technische basis, bijvoorbeeld bij het samenstellen van een framework of stack. De vrijheid in het kiezen van een bepaalde werkwijze is binnen de factory dus geconcentreerd bij enkele exceptionele professionals. Buiten de factory hebben alle professionals deze mogelijkheid.

De vrijheid van de exceptionele professionals binnen de factory is wel beperkt tot één technologie platform. Dit komt omdat de ontwikkelstraten die zijn ingericht software bouwen voor een bepaald platform. Dit betekent dat bepaalde tools of databases niet gebruikt kunnen worden in de factory.

Bureaucratie

Een factoryaanpak en standaardisatie brengt een zekere bureaucratie met zich mee. In hoofdstuk 4 werden twee vormen van bureaucratie beschreven, de helpende en dwingende bureaucratie. De resultaten van de metingen geven ook inzicht in welk type bureaucratie er aanwezig is in de softwarefactory. De indeling van Adler en Broys (1996) is gebaseerd op de onderdelen repair, internal transparancy, global transparancy en flexibility.

Op het gebied repair is vanuit de factory een helpende bureaucratie. In de organisatie is bijvoorbeeld een testfase ingebouwd als onderdeel in het proces. De resultaten hiervan worden binnen het team gecommuniceerd naar de programmeurs (en niet gerapporteerd naar management). Dit is een voorbeeld van helpende bureaucratie zichtbaar op het gebied van repair.

De internal transparancy is in de factory zichtbaar in het resultmodel, de trainingen en presentaties die worden gegeven en de kennisoverdracht tussen de exceptionele en niet-exceptionele groep. Dit zijn voorbeelden van momenten waarop er inzicht is in de eigen werkwijze, en waarom deze zo is.

Global transparancy is in het Product Centre duidelijk zichtbaar in het Product Model. Het Product Model geeft aan iedereen inzicht in alle processen in de softwarefactory, ongeacht welke rol iemand inneemt.

Flexibility is binnen de factory zoals verwacht beperkter dan buiten de factory. Er zijn vaste tools waarmee gewerkt moet worden, en de RUP aanpak is standaard. Eén exceptionele professional benoemde de 'RUP dwangbuis', waarbij met name werd verwezen naar de sfeer die in sommige gevallen in de factory aanwezig kan zijn. Het ging hierbij met name om bepaalde documenten of stappen uit RUP die worden verwacht, zonder dat daar in de ogen van de professional een duidelijke reden voor is / doel voor is. Dit is een voorbeeld van een incident waarbij dwingende bureaucratie zichtbaar is (het systeem geeft de professional opdrachten in plaats van andersom). Dit is echter relatief beperkt zichtbaar geworden bij andere teams of professionals, alle professionals hadden naar eigen zeggen invloed op de eigen werkwijze.

Vergeleken met de situatie die is waargenomen buiten de factory, is te zeggen dat beide omgevingen het aansluiten op de helpende bureaucratie. In de factory is de flexibiliteit beperkter dan buiten de factory, met name op het gebied van platform of tool keuze. Internal transparancy, global transparancy en repair zijn in de factory beiden aanwezig van de helpende bureaucratie variant.

Resultaat

Tijdens de observaties die zijn gedaan, bleek de werkwijze van een professional binnen de factory en buiten de factory erg gelijkend. In de stappen die gezet werden om tot realisatie van de software te komen waren geen verschillen zichtbaar. Ook was in beide omgevingen een vorm van standaardisatie

zichtbaar, doordat teams zelfstandig eigen standaarden definiëren. De verschillen die zijn waargenomen zitten met name in standaard tooling en infrastructuur, een beperking in de vrijheid van niet-exceptionele professionals, en een beperking in vrijheid door de vaste platformkeuze in de softwarefactory. Daarnaast was de beschikbaarheid van de procesbeschrijving alleen waargenomen in de softwarefactory, al leidde dit niet tot een andere werkwijze in de programmeerfase.

9.1.3 Hoe ervaren hele goede professionals en gemiddelde professionals het softwareproces binnen de softwarefactory?

Er is er een systematiek gekozen om de professionals van het Product Centre in te delen in de groepen exceptioneel en niet-exceptioneel. Dit is gedaan met behulp van de methode van Turley en Bieman (1995). Bij beide groepen is de werkwijze geobserveerd, zijn er gesprekken gevoerd en is er een vragenlijst afgenomen.

Een van de hypothesen bij deze subvraag was dat exceptionele professionals meer sturing ervaren in het Product Centre. Dit type professional is naar verwachting het meest bezig met het bepalen van de eigen werkwijze, en wil daarin mogelijk meer inspraak dan een niet-exceptionele professional (Fitzgerald, 1997). Voor de niet-exceptionele professional is de hypothese dat het oordeel van de professional op de sturing positiever is, omdat deze professional minder behoefte heeft aan vrijheid in het bepalen van de aanpak.

Geen verschil tussen exceptionele en niet-exceptionele met betrekking tot standaardisatie

Op basis van de metingen in de vragenlijst worden beide hypothesen verworpen. De vragen van de vragenlijst zijn gecombineerd in twee schalen ‘mate waarin de standaard / sturing wordt ervaren’ en ‘mate waarin de standaard wordt gewaardeerd’. Er was geen relatie zichtbaar tussen de groepen exceptioneel en niet-exceptioneel en het ervaren van de standaard/sturing. Ook in de waardering van de professional voor de standaardisatie was geen verschil zichtbaar tussen de exceptionele en niet-exceptionele groep. Een mogelijke verklaring hiervoor werd zichtbaar tijdens het bekijken van het softwareproces. Daarbij bleek de exceptionele professional ander werk te doen dan de niet-exceptionele professional. De exceptionele professional neemt de rol in van architect. Deze rol brengt meer vrijheden met zich mee. Zo heeft de architect invloed op het raamwerk van de applicatie en de werkwijze van het team. Dit kan ervoor zorgen dat de exceptionele professional minder merkt van de sturing in de factory en voldoende vrijheid heeft.

Beide groepen geven in de vragenlijst aan dat ze invloed op de eigen werkwijze en de werkwijze van het team hebben, niet alleen de exceptionele professionals. Binnen de teams bestaat een relatief vrije omgeving, die kan worden vergeleken met teams buiten een factory.

Redenen en bereidheid tot standaardisatie

Uit de vragenlijst en de gesprekken bleek dat de managers en de professionals allen het nut in zien van de standaardisatie. Er werden verschillende redenen genoemd voor standaardisatie:

- Kostenbesparing
- Het uitbesteden en overdragen van het werk is gemakkelijker
- Noodzakelijk voor CMMi certificering
- Duidelijke standaarden versoepelen de inrichting van nieuwe straten

Het feit dat alle professionals het nut inzien van de standaardisatie is ook terug te zien in de houding ten opzichte van verandering. De aan het onderzoek deelnemende professionals zijn allen bereid om vrijheid in te leveren voor de standaard werkwijze.

Resultaat

Er is geen duidelijke relatie waargenomen tussen de groepen exceptionele en niet-exceptionele professionals, en hoe zij het softwareproces in het Product Centre ervaren en waarderen. Een reden hiervoor kan zijn dat de exceptionele professionals meer vrijheid krijgen vanuit hun rol in de factory. In hun rol als architect merkt deze groep minder van de sturing in de softwarefactory.
Ten slotte is waargenomen dat de professionals uit beide groepen en de betrokken projectmanagers allen het nut inzien van de standaardisatie in het softwareproces. De betrokkenen noemen verschillende redenen waarom standaardisatie gewenst is, en zijn tevens bereid hiervoor vrijheid in te leveren.

9.1.4 Is de efficiency en kwaliteit in een softwarefactory altijd hoger dan in een niet factory?'

Bedrijven hebben verschillende doelen met het inrichten van de ontwikkelafdeling als softwarefactory. De twee belangrijkste doelen zijn efficiency en kwaliteit. In deze paragraaf worden de resultaten van het onderzoek samengebracht die helpen bij het beantwoorden van de subvraag: Is de efficiency en kwaliteit van een software factory altijd hoger dan een niet factory?

Er zijn verschillende onderzoektechnieken gebruikt om deze vraag te beantwoorden. Er is ondermeer een performancemeting gedaan bij één team binnen de factory, en één team buiten de factory. Bij deze meting is de performance gemeten in efficiëntie en kwaliteit. De efficiëntie wordt berekend op basis van SLOC (regels code) per uur en FP (functiepunten) per uur. Uit de efficiencymeting werd zichtbaar dat bij de twee teams waarbij gemeten werd het aantal benodigde uren per FP 59% lager lag buiten de factory. De performance op dit gebied lag in dit geval dus hoger buiten de softwarefactory dan in de softwarefactory. Het hogere aantal uren per functiepunt binnen de factory werd veroorzaakt doordat er extra ondersteuning is geleverd aan de klant bij installatie, en doordat het koppelen met een component van een derde partij meer tijd kostte. Deze meting is niet representatief voor alle werkzaamheden van het team in de factory. Daarvoor zouden meer metingen gedaan moeten worden. De meting geeft echter wel aan dat het factoryconcept op dit moment niet alle projectinvloeden controleert, en daardoor niet per definitie een betere performance heeft dan een project dat zonder factoryconcept werkt.

Bij de twee teams waar de performancemeting is uitgevoerd is over dezelfde projecten ook een kwaliteitsmeting uitgevoerd. Een veelgebruikte manier om kwaliteit van software meetbaar te maken

is het meten van het aantal fouten, oftwel bugs, die in de software zitten. Het aantal bugs per functiepunt binnen de factory ligt bij deze meting 77,8% lager dan bij de organisatie buiten de factory. Op dit punt heeft de softwarefactory in de gemeten situatie dus een betere performance dan de niet-factory.

Evolutie

Naast de performancemeting gaven ook de andere onderzoeksmethoden inzicht in de performance effecten binnen de factory. Zo is er een concreet voorbeeld waargenomen van de veroudering van een factoryonderdeel. Een van de elementen van het Product Centre, het broncode-managementsysteem, voldeed niet meer volledig aan de eisen die de projecten hier aan stelden. Vanwege functionaliteit en performancegebrek bleek het gebruik van deze tool binnen het project niet meer geschikt. Men wilde graag een alternatieve tool gebruiken. Vanwege de standaardisatie in de factory kon men echter niet direct overstappen op een andere tool. Er was tijd nodig voor het nemen van de beslissing voor één van beide tools. Deze keuze moest niet alleen vanuit het project, maar ook vanuit de factory worden verantwoord. Het gebruik van tools die niet meer volledig aansluiten bij de projecten of de teams in de factory, werkt negatief op de performance van de factory. In de waargenomen situatie is er beperking van de projectvoortgang, en lijkt de tool die niet meer volledig past geen performancevoordeel op te leveren maar een performancenadeel.

Standaardisatie

De mate waarin de standaarden in de factory worden toegepast is nog erg verschillend binnen de teams en de managers waarmee gesproken is. Sommige teams gebruiken bijvoorbeeld wel de factorytemplates en bepaalde tools, andere teams niet. De literatuur geeft enkele verklaringen voor verschillen in acceptatie en gebruik van standaarden. Mensen hebben een natuurlijke weerstand tegen veranderingen (Chroust, 2002), en sommige standaarden zijn mechanisch opgezet (Sommerville & Rodden, 1996) en daardoor moeilijk inzetbaar in de praktijk. Een andere reden is dat standaardisatie voor een hele range projecten moeilijker is dan voor één specifiek project (Börjesson & Mathiassen, 2003). Om iets te zeggen over het effect van deze differentiatie in gebruik van de standaarden is van belang te weten wanneer en op welke punten men afwijkt van de standaard. Wordt er afgeweken omdat de standaard werkwijze niet voldoende past bij het project, dan kan de afwijking een positieve uitwerking op kwaliteit en efficiëntie hebben (Sommerville, Sawyer, & Viller, 1999). Wordt er echter van de standaard afgeweken, bijvoorbeeld omdat kennis over de standaard ontbreekt, dan kan het zijn dat deze afwijking de performance en kwaliteit negatief beïnvloedt. Beide varianten zijn waargenomen tijdens de observaties in de softwarefactory. Zo is er een team die een eigen gekozen codegenerator gebruikt voor het genereren van code en unittests omdat hiermee veel tijd bespaard wordt en minder fouten worden gemaakt. Dit kan een positief effect hebben op de performance. Een ander team maakt geen gebruik van de checklists die de factory biedt. Dit heeft mogelijk een negatief effect op de performance, doordat fouten minder snel geconstateerd worden. De verschillen tussen het gebruik van standaardisatie zijn dus niet per definitie te kenmerken als positief of negatief voor de performance van de softwarefactory.

De professionals

Een belangrijke invalshoek in dit onderzoek is die van de professional. De eerder beschreven standaardisatie zal direct van invloed zijn op de performance van de factory. Deze standaard kan mogelijk ook indirect van effect zijn, namelijk door de wijze waarop de professionals het werken met de standaardisatie ervaren. Daarom zijn in dit onderzoek metingen gedaan naar hoe de professionals het factoryproces ervaren, door middel van een vragenlijst. Er werden in de vragenlijst enkele elementen genoemd die als knellend werden ervaren in de factoryaanpak. Elementen die werden benoemd was het gebruik van bepaalde tools, de RUP werkwijze en de sfeer hier omheen, en het broncodemanagementsysteem. Uit de andere metingen bleek echter dat dit uitzonderingen waren, en dat over het algemeen de teams relatief vrij zijn in hun omgang met de standaard. Uit de meting bleek dat de exceptionele groep niet meer last hebben van sturing door de factorystandaarden dan de niet-exceptionele groep. Alle professionals gaven aan zelf invloed te hebben op de eigen werkwijze en die van het team. Ook waren alle professionals bereid om vrijheid in te leveren, als gevolg van de standaard aanpak. Deze constateringen geven geen indicaties voor een negatieve ervaring van de standaardisatie door de professionals. Professionals die achter de gebruikte aanpak staan, presteren naar verwachting beter dan professionals waar bij dit niet het geval is (Erez, Earley, & Hulin, 1986). De factoryaanpak in het Product Centre zal op dat gebied dus geen negatieve invloed hebben op de performance.

Resultaat

De omstandigheden in de factory zijn nog niet zodanig gecontroleerd dat de efficiëntie binnen de softwarefactory altijd hoger ligt dan buiten de factory. Er zijn invloeden zoals derde partijen die mee ontwikkelen aan de software of de wensen van klanten die het proces beïnvloeden en minder voorspelbaar maken. Daarnaast beïnvloedt de evolutie van factoryonderdelen de efficiëntie in sommige gevallen negatief. Onderdelen die geselecteerd worden zijn op een bepaald moment niet meer optimaal. Tenslotte is het effect van de softwarefactory bekeken op de professionals die in de factory werken. Hierbij is zichtbaar dat het proces de professionals naar eigen zeggen voldoende vrijheid geeft. De acceptatie van de standaard werkwijze door de professionals kan positief doorwerken op de performance in de softwarefactory.

10 Conclusie

Welke factoren beïnvloeden de doelmatigheid van het softwareproces in een softwarefactory?
Deze hoofdvraag is aan het begin van het onderzoek gesteld. Het onderzoek heeft inzicht opgeleverd in factoren die een belangrijke rol spelen in de doelmatigheid van de softwarefactory. De doelen die het Product Centre heeft gesteld zijn: voorspelbaarheid, hoge kwaliteit in minder tijd en met lagere kosten. In de volgende twee paragrafen worden de factoren beschreven die tijdens het onderzoek zichtbaar zijn geworden. De doelen 'minder tijd en lagere kosten' zijn ondergebracht onder de noemer 'efficiëntie'.

10.1 Factoren die de voorspelbaarheid beïnvloeden

Bij het bestuderen van het softwareproces zijn factoren zichtbaar geworden die van invloed zijn op de voorspelbaarheid binnen het Product Centre. Van *hergebruik* wordt verwacht dat dit de voorspelbaarheid van projecten in de factory vergroot, en hergebruik is waargenomen in het Product Centre. Daarnaast is het *gebruik van standaarden* van invloed op de voorspelbaarheid. Wanneer alle teams een bepaalde checklists toepassen, kan een bepaald type fouten worden voorkomen en maakt dit het geheel meer voorspelbaar. De standaard checklists worden binnen het Product Centre echter nog verschillend gebruikt per team.

Ook de mate van het gebruik van de standaard tools of processen kan de voorspelbaarheid beïnvloeden. Ten slotte werd tijdens het onderzoek ook zichtbaar dat invloeden van buitenaf de voorspelbaarheid negatief kunnen beïnvloeden. Het gaat dan bijvoorbeeld om een opdrachtgever die specifieke wensen heeft. De mate waarin de softwarefactory deze externe invloeden kan controleren, lijkt ook van invloed op de voorspelbaarheid van het softwareproces.

Wanneer de softwarefactory er in slaagt het aanwezige hergebruik te behouden en te optimaliseren, het gebruik van de standaard factoryonderdelen meer te laten toenemen en externe invloeden kan beperken, zal dit naar verwachting de voorspelbaarheid van de factoryprojecten in het Product Centre verbeteren.

10.2 Factoren die de efficiëntie en kwaliteit beïnvloeden

Het factoryproces dat is ingevoerd is uiteindelijk gericht op het behalen van winst, of dit nu kwaliteitswinst betreft of financiële winst. Er zijn tijdens het onderzoek factoren zichtbaar geworden die winst op deze twee vlakken beïnvloeden.

Bij het opzetten en onderhouden van de standaarden in de softwarefactory, is investering door de exceptionele professionals een vereiste. Deze professionals moeten namelijk veralgemeniseren en vastleggen hoe ze het werk aanpakken en wat ze doen om succes te behalen. In de tijd die deze exceptionele professionals besteden aan het onderhouden van de standaarden, kunnen ze zelf niet aan projecten werken. Dit vergt dus een zekere investering, die moet worden terugverdiend. Het terugverdienen van de investering kan gebeuren op twee manieren:

- de projecten in de factory zijn door de investering niet meer afhankelijk van de exceptionele professionals in de teams, en kunnen met minder kostbare krachten hetzelfde resultaat behalen.
- de investering zorgt voor winst puur door een efficiëntere werkwijze. Hierdoor kan een team, (met exceptionele professional) extra winst behalen.

De eerstgenoemde manier, het beperkter nodig zijn van de exceptionele professional, is bij het Product Centre nog niet zichtbaar. De exceptionele professional speelt nog een grote rol in de teams binnen de factory, en op dit moment lijkt deze rol nog niet te worden vervangen door de mate van standaardisatie. Dit betekent dus dat de professionals nog steeds nodig zijn, maar dat deze professional een deel van de tijd investeert in het onderhouden van de factory. Deze tijd zou, zonder de factoryaanpak, kunnen worden besteed aan het werken aan projecten.

De tweede manier, waarbij de standaard werkwijze voor meerdere projecten winst oplevert, is de vorm die in het Product Centre zichtbaar is. Of deze manier doelmatig is, is afhankelijk van de potentiële winst (PWinst) die uit de standaard wordt gehaald, en het aantal projecten dat met de standaard kan worden uitgevoerd.

$$Winst\ door\ factorystandaard\ =\ AantalProj\ \times\ PWinst$$

Het *aantal projecten* dat in de factory kan worden uitgevoerd wordt echter beïnvloed door de scope van de factory. Hoe groter de scope, hoe meer projecten er in kunnen worden uitgevoerd. De scope beïnvloedt tevens de winst per project. Een bredere scope zorgt voor meer projecten, maar ook voor minder winst per project door gebruik van de standaard (omdat de standaard niet meer specifiek is).
Daarnaast is de positie in de evolutie van belang. Tijdens het onderzoek werd duidelijk dat een standaard niet altijd dezelfde waarde houdt. De standaard is onder invloed van *evolutie*, en wordt in de tijd steeds minder bruikbaar en passend. Dit komt omdat de standaard hetzelfde blijft en de omgeving verandert.
Een andere factor is de *inconsistentie* van het gebruik van de standaard. Indien een standaard niet consistent wordt toegepast, of niet juist wordt toegepast, kan dit de potentiële winst verminderen.
Ten slotte moet niet vergeten worden dat er is geïnvesteerd in de standaard. Er kan door de exceptionele professional tijd in geïnvesteerd zijn, of er kan een kostbare tool zijn aangeschaft. Deze *investering* wordt daarom ten laste gebracht van de winst.

Deze inzichten kunnen als volgt worden weergegeven:

$$Winst\ door\ factorystandaard\ =$$
$$(\ AantalProj\ \times\ Scope\ )\ \times \left(\frac{PWinst}{Scope}\right) - \text{Evol} - \text{Incons} - \text{Invest}$$

Wanneer de softwarefactory er in slaagt de scope zodanig te kiezen dat de potentiële winst en het aantal projecten dat in de factory kan worden uitgevoerd in balans is, dan is de mogelijke financiële winst en kwaliteitswinst naar verwachting optimaal. De onderzoeksresultaten wijzen er echter wel op dat factoryonderdelen onderhevig zijn aan *evolutie*, dat ze niet per definitie *consistent* gebruikt worden en dat ze *investering* vragen.

10.3 De ultieme factory

Softwaredevelopment is, zoals alle nieuwe engineering disciplines ontstaan als een ambacht. Een relatief kleine groep kundige mensen wist hoe men hoge kwaliteit software moest maken. Toen de vraag naar software toe nam, ontstond er de wens binnen organisaties om op te schalen. Opschalen op twee manieren: het realiseren van grotere en complexere softwaresystemen, en het realiseren van een grotere hoeveelheid softwaresystemen. Ambachtelijk werk schaalt echter slecht op, ondermeer doordat de groep kundige en ervaren mensen eindig is.

Eén van de mogelijkheden die organisaties hebben om hier mee om te gaan is het verdelen van de ervaren en kundige mensen over verschillende teams. Deze teams bestaan uit meerdere minder ervaren mensen, met één kundige en ervaren professional. Hiermee kan er meer geproduceerd worden, en dragen de kundige professionals tijdens projecten hun kennis over aan de onervaren professionals. Dit lijkt, gebaseerd op de waarnemingen die zijn gedaan, de positie van het Product Centre van BedrijfX te zijn.

In een verder doorgevoerde softwarefactory kan men nog verder opschalen. De ervaren professional legt zijn werkwijze vast, en men kiest en maakt handige hulpmiddelen. Het resultaat hiervan wordt vastgelegd en wordt een standaard. Wanneer de minder ervaren professionals nu een project willen doen, kan men op basis van de standaard aanpak het project volbrengen op het performance en kwaliteitsniveau van de ervaren professional. Daarnaast is het voordeel hiervan dat iedereen op dezelfde manier werkt, en daarmee worden medewerkers vervangbaar, werk overdraagbaar en resultaten voorspelbaar.
Het nastreven van deze ultieme softwarefactory, met daarin een ver doorgevoerde standaard en een hoge voorspelbaarheid, hoeft niet per definitie het gewenste positieve resultaat te brengen.

In het ergste geval zorgt de combinatie van een beperkte scope, en de snelheid waarmee de evolutie van het softwarelandschap op dit moment verloopt ervoor dat de winsten die behaald worden met de standaard werkwijze zeer beperkt blijven. Daarnaast kunnen de exceptionele professionals minder worden ingezet op projecten, omdat deze de standaard moeten onderhouden.

In de ultieme factory zijn de verschillende onderdelen naadloos op elkaar afgestemd. De tools, standaarden en werkwijze uit elke fase van de ontwikkelcyclus sluiten op elkaar aan en vormen een zogenaamde *tool chain* (Neema, Scott, & Karsai, 2005). De meest ultiem afgestemde factory kan echter ook de minst houdbare zijn. Juist door het feit dat alles zo goed op elkaar is afgestemd, krijgen kleine veranderingen een grotere impact. Wanneer één proces of tool moet worden vervangen (b.v. door de evolutie), heeft dit grote gevolgen voor de aansluiting op de andere onderdelen van de factory.

Het Product Centre van BedrijfX bevindt zich nog niet in het hierboven beschreven scenario. De teams hebben een zekere vrijheid in het bepalen van hun werkwijze en de standaard wordt nog niet strikt gehandhaafd. De geschetste effecten zijn echter niet ondenkbaar wanneer ambities als standaardisatie en voorspelbaarheid in de softwarefactory worden gepresenteerd, zonder de bijbehorende keerzijde en nuances.

11 Visie

In het voorgaande hoofdstuk zijn de onderzoeksresultaten beschreven, gebaseerd op de waarnemingen die in het onderzoek zijn gedaan. Dit onderdeel van de scriptie is een aanvulling op de onderzoeksresultaten, en schetst de inzichten en ideeën van de onderzoeker. Deze inzichten zijn in dit hoofdstuk vertaald naar een SWOT-analyse (Strengths, Weaknesses, Opportunities, Threats) voor het Product Centre.

In de SWOT worden delen van de onderzoeksresultaten kort opnieuw beschreven. Vervolgens worden er aanbevelingen gedaan op basis van de SWOT analyse.

11.1 SWOT

Sterkten	Zwakten
▪ Hergebruik is aanwezig en in ontwikkeling (E-platform en Oracle2Go) ▪ Er zijn vrijheden in de factory voor de juiste developers (exceptionele professionals) ▪ Helpende bureaucratie is waargenomen ▪ Laag aantal fouten in software is waargenomen bij meting fouten per functiepunt	▪ Voorspelbaarheid nog niet optimaal ▪ Performance niet altijd optimaal ▪ Tooling sluit niet altijd aan op projecten ▪ Gebruik van de standaarden is nog zeer verschillend tussen teams
Kansen	**Bedreigingen**
▪ Bereidheid van developers voor een verdere standaardisatie ▪ Meer aandacht voor evolutie van tooling ▪ Dynamiek als verrijking zien in plaats van als bedreiging ▪ Metrieken toepassen	▪ Bureaucratieverandering (b.v. voor CMM certificering) kan de exceptionele professionals afschrikken ▪ Evolutie van factorycomponenten ▪ Vergaande standaardisatie beperkt de complexiteit en flexibiliteit die men in de factory aankan.

Tabel 12: Kansen en bedreigingen

De verschillende punten die zijn genoemd in deze SWOT analyse worden in de volgende paragrafen gespecificeerd.

11.2 Sterkten

Hergebruik is aanwezig en in ontwikkeling (E-platform en Oracle2Go)

Het hergebruik dat in de factory aanwezig is, lijkt in de praktijk een positief effect te hebben op de factoryresultaten. Oracle2Go en E-platform zorgen ervoor dat projecten met een hogere kwaliteit en binnen kortere tijd kunnen worden gerealiseerd, en worden ook door ontwikkeld.

Er zijn vrijheden in de factory voor de juiste developers (exceptionele professionals)

In de factory hebben de exceptionele professionals meer vrijheden in het bepalen van de werkwijze dan de niet-exceptionele professionals. Dit komt doordat de exceptionele professionals vaak de rol invullen van architect. Professionals met deze rol hebben meer vrijheid in het bepalen van de aanpak van het project. Ook geven ze richting aan de niet-exceptionele professionals. Deze werkwijze zorgt ervoor dat per project door een ervaren developer de werkwijze wordt bepaald en draagt bij aan de flexibiliteit van de softwarefactory. Daarnaast lijkt deze extra vrijheid voor de exceptionele professional ervoor te zorgen dat deze groep, anders dan verwacht, niet meer geconfronteerd wordt met de standaarden dan de niet-exceptionele professionals. Dit lijkt bij te dragen aan de tevredenheid van de professionals over hun werksituatie.

Helpende bureaucratie waargenomen

In de factory is een helpende bureaucratie waargenomen. Ook dit draagt bij aan de personeels-tevredenheid en effectiviteit.

Laag aantal fouten in software waargenomen bij meting fouten per functiepunt

Bij de bugs per functiepunten meting heeft de factory een positieve score behaald. Deze meting (hoewel een momentopname binnen een project) geeft aan dat er projecten binnen de factory zijn met een hoge kwaliteit software als resultaat.

11.3 Zwakten

Voorspelbaarheid nog niet optimaal

Bij de performancemeting die is uitgevoerd kwam naar voren dat de voorspelbaarheid van een factoryproject niet in alle gevallen optimaal is. In de gemeten situatie was een overschrijding van de geplande uren zichtbaar. Dit kwam door externe invloeden die niet door de factory gecontroleerd konden worden. Deze externe invloeden maken de factory minder voorspelbaar en kunnen de kwaliteit van het eindproduct negatief beïnvloeden.

Performance niet altijd optimaal

Bij de performancemeting scoorde de factoryaanpak niet optimaal. In de meting die is uitgevoerd was het aantal functiepunten dat per uur werd gehaald minder dan bij de meting buiten de factory. De factoryopzet zorgt er dus nog niet voor dat in alle gevallen een optimale performance gehaald wordt.

Tooling sluit niet altijd aan op projecten

Het gebruik van de standaarden binnen de factory is nog zeer verschillend tussen teams. Zowel het gebruik van bepaalde factorytools als het gebruik van vaste templates of tools is heel verschillend tussen teams.

Gebruik van de standaarden is nog zeer verschillend tussen teams

Het komt voor dat de tooling uit de factory niet volledig aansluit op het project dat wordt uitgevoerd. In dat geval zou men een andere tool willen selecteren, maar is dit niet altijd mogelijk vanwege het standaardisatiebeleid. Dit type situaties kan zorgen voor performancebeperking.

11.4 Bedreigingen

Bureaucratieverandering (b.v. voor CMM certificering) kan de exceptionele professionals afschrikken

Een positief punt van de huidige softwarefactory is het feit dat er een helpende bureaucratie is waargenomen. Er zijn echter ontwikkelingen (zoals de CMM certificering) waarbij processen worden gestroomlijnd en vastgelegd. Dit soort ontwikkelingen heeft mogelijk een verandering in bureaucratie tot gevolg en kan een meer dwingende bureaucratie als resultaat hebben. Een dwingende bureaucratie kan de factory minder flexibel maken en kan de professionals beperken.

Evolutie van factorycomponenten

Onderdelen van de factory, zoals een standaard proces of een standaard tool, zijn onderhevig aan evolutie. Met name op het gebied van tooling gaan de ontwikkelingen in de IT-branche erg snel en zijn tools op het moment van invoeren al bijna verouderd. De complexiteit hierin zit in het moment waarop van tool moet worden gewisseld en op de aansluiting tussen de tools onderling. Bij een set van tools die volledig op elkaar is afgestemd, is de impact van verandering groter. Op het gebied van tooling zijn tevens kansen, deze zijn opgenomen in de volgende paragraaf.

Vergaande standaardisatie beperkt de complexiteit en flexibiliteit die men in de factory aankan

Een ideaalbeeld van de softwarefactory als volledig gestandaardiseerd systeem beperkt de complexiteit die men in de factory aankan. Hoewel de minder ervaren professionals mogelijk het kwaliteits- en performanceniveau van de ervaren professionals kunnen benaderen door het volgen van de ver doorgevoerde standaard werkwijze, zijn deze professionals een stuk minder flexibel. Ze hebben namelijk geleerd de standaard werkwijze te volgen, maar doen niet alle kennis en vaardigheden op van de ervaren professional. Daardoor kunnen ze minder goed inspelen op afwijkende situaties.

Een risico van het invoeren van een vergaande standaardisatie in de softwarefactory, is dat de minder ervaren mensen in de factory minder snel zo vaardig worden als de ervaren professional. Het is denkbaar dat de ervaren professionals het werken in de factory minder aantrekkelijk gaan vinden en daardoor wegstromen. Vanuit het oogpunt van de articulatie (zie paragraaf 4.2) blijven deze vaardige en ervaren professional essentieel in de factory. Er zijn punten in het proces waar de standaard werkwijze niet past, en er een nieuwe aanpak bedacht moet worden. Dit is met name het geval in de complexere projecten. Dit type projecten is in de westerse softwarefactory's geconcentreerd. Het beleid van BedrijfX is namelijk om met name projecten met hoge complexiteit en risico's in de

westerse software factory uit te voeren. De minder risicovolle en minder complexe projecten worden offshore uitgevoerd. De strategie van de softwarefactory kan hierop worden afgestemd.

11.5 Kansen en Aanbevelingen

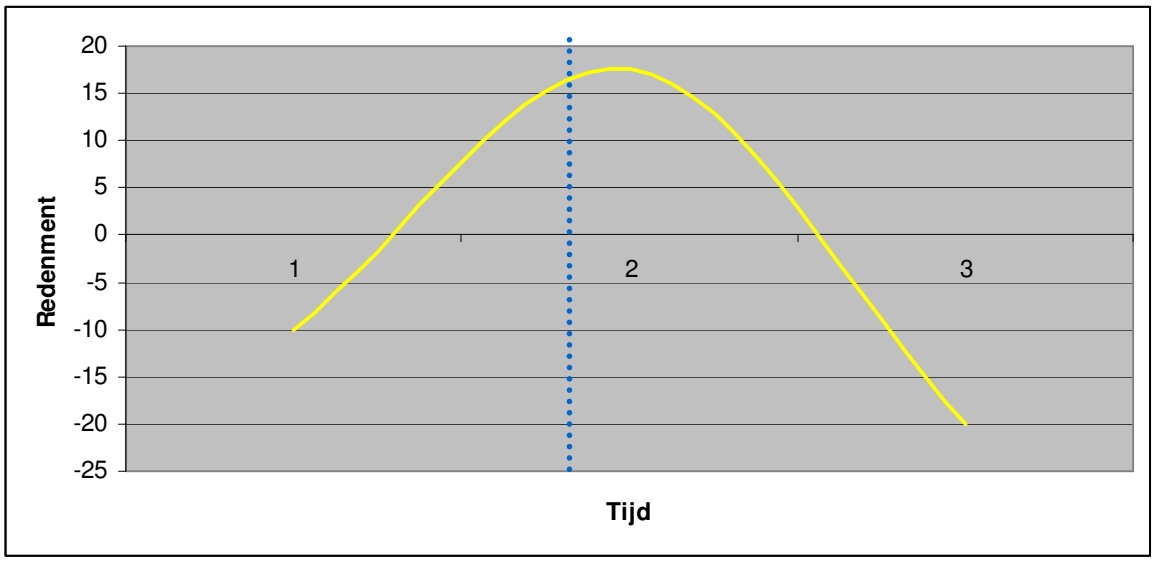
De kansen uit de SWOT analyse zijn in deze paragraaf tevens vertaald naar aanbevelingen.

Bereidheid van developers voor een verder gebruik van standaardisatie

Een van de zwakten van de factory zoals deze nu is, is dat het gebruik van de standaarden tussen de teams erg verschilt. Uit het onderzoek blijkt dat developers wel bereid zijn om meer te standaardiseren en daarvoor vrijheid op te geven. Hier liggen dus kansen voor de organisatie om de verschillende teams meer volgens de aanwezige standaard te laten werken.

Evolutie van het factoryproces

Bij het inrichten van de factory wordt het proces, de bijbehorende tooling en templates en rolverdeling afgestemd op de situatie en de projecten die op dat moment worden uitgevoerd. Je kunt stellen dat een factory, net als bij software die opgeleverd is, na ingebruikname een evolutie doormaakt. Scherper gezegd betekent dit dat de factory, om te kunnen overleven, zich zal moeten aanpassen aan veranderende omstandigheden. Hierbij zal er telkens een afweging gemaakt moeten worden gemaakt of de werkwijze en tools van de factory de resultaten van de factory nog steeds optimaliseren. Een tool of werkwijze die verouderd is, kan het proces ook negatief beïnvloeden, wanneer deze niet meer aan de eisen voldoet die de ontwikkelorganisatie er aan stelt. Dit inzicht is gebaseerd op de waarnemingen die zijn beschreven in hoofdstuk 7, en paragraaf 0 in bijzonder. Ook geven exceptionele professionals in de vragenlijst aan dat er sommige dingen gestandaardiseerd zijn in het Product Centre, die geschikt waren voor projecten in het verleden, maar tegenwoordig niet goed meer aansluiten.



Figuur 13

Als we deze waarnemingen verder interpreteren, dan kan gezegd worden dat een factoryonderdeel (een tool, template of werkwijze) een ontwikkeling doormaakt (zie Figuur 13). Op het moment dat de tool of template geïntroduceerd wordt, is er in geïnvesteerd, en is er een periode waarin de investering terugverdiend wordt (1). Nadat de initiële investering is terugverdiend kan een periode winst gemaakt worden door de voordelen van het gebruik van het factoryonderdeel. Deze winst kan tijdswinst of kwaliteitswinst zijn.

Na verloop van tijd kan het factoryonderdeel minder goed gaan aansluiten op de projecten die in de factory worden uitgevoerd, en vlakt de winst die wordt gehaald door gebruik ervan af (2). Daarna zal de tool de prestaties van de factory negatief beïnvloeden (3). De winst die de standaard werkwijze en standaard onderdelen van de factory brengen, kan negatief worden beïnvloed door tools die te lang in gebruik blijven. De factorydoelstelling *´software sneller en goedkoper produceren´* wordt hierdoor beïnvloed. Om een factoryproces optimaal te krijgen en te houden, zal de evolutie van factoryonderdelen bewaakt moeten worden. Onderdelen waarvan de voordelen teruglopen (het kritisch punt in de rendementsgrafiek) zullen aandacht moeten krijgen en zullen vervangen of aangepast moeten worden.

Hier ligt een kans voor het Product Centre. Op het gebied van tooling zijn er verbetermogelijkheden. Wellicht kan er een aparte verantwoordelijke voor tooling worden aangewezen bij verbeterinitiatieven. Een andere optie is om teams zelf in discussie te laten gaan over nieuwe tooling en op welke punten het meeste winst te behalen is.

De dynamiek van projecten als verrijking in plaats van als bedreiging

Vanuit de softwarefactory is standaardisatie en voorspelbaarheid ten doel gesteld. Een denkwijze kan zijn dat deze doelen onder druk staan door de projecten die in de factory worden uitgevoerd. Deze projecten vragen vaak om net een andere aanpak dan waarin het factoryproces voorziet. Het is daardoor denkbaar dat er een sfeer ontstaat waarin afwijking van het factoryproces als negatief wordt ervaren.

Een andere manier om naar deze ontwikkeling te kijken, is dat de dynamiek van de projecten helpt bij het voeden van de factory met nieuwe elementen. Templates, tools of werkwijzen die in projecten uitgeprobeerd worden en succesvol blijken kunnen dan worden gepromoveerd tot factory standaard.
Bij het volgen van het proces van een projectteam tijdens het onderzoek, werden nieuwe processen en handleidingen voor tools zichtbaar die binnen projecten gemaakt zijn. Deze nieuw gemaakte handleidingen en processen kunnen aanwijzingen zijn voor de punten waarop de factory kan worden verrijkt.

Metrieken toepassen

Op dit moment is het gebruik van metrieken binnen de developmentteams nog beperkt. Gegevens over performance en kwaliteit kunnen zeer waardevol zijn voor het zichtbaar maken van de effecten van veranderingen. Het effect van nieuwe tooling of andere verbeterinitiatieven kan dan meetbaar worden gemaakt. Het is voorstelbaar dat een betrouwbare functiepunten telling voor elk project nog te veel investering vraagt. Het tellen van het aantal fouten per 1000 regels code kan echter vrijwel geautomatiseerd worden gedaan, en kan toch een beeld geven van de opgeleverde kwaliteit. Ook is dit type metrieken erg interessant voor vervolgonderzoek naar het softwareproces.

12 Evaluatie

Het belangrijkste doel van dit onderzoek was het verkrijgen van inzicht in het softwareproces in een softwarefactory, en welke factoren de doelmatigheid van een dergelijk softwareproces beïnvloeden. Het onderzoek heeft een goed beeld opgeleverd van het softwareproces in het Product Centre en de belangrijkste thema's die hierin een rol spelen.

De onderzoekscomplexiteit was hoger dan aanvankelijk was verwacht. Veel mogelijkheden die vooraf waren bedacht bleken in de praktijk niet mogelijk. Oorzaken hiervan zaten in het feit dat er zeer beperkte bestaande input beschikbaar was voor het onderzoek, en dat het zichtbaar maken van het proces van de ontwikkelaars moeilijker bleek dan verwacht. Hierdoor was improvisatie nodig, en is het onderzoek bijgesteld.

Aan het begin van het onderzoek was één van de doelen het verzamelen van prestatiecijfers van projecten die in de factory werden uitgevoerd. Bestaande gegevens waren echter niet beschikbaar, of konden niet gebruikt worden voor doeleinden van het onderzoeksproject. Er zijn momenteel wel initiatieven binnen het Product Centre om meer te doen met metrieken, alleen waren de resultaten hiervan tijdens het onderzoek nog niet beschikbaar. Dit maakte dat alle gegevens tijdens het onderzoek zelf ingewonnen moesten worden. Hiervoor zijn verschillende activiteiten ontplooid. Tijdens het onderzoek zijn gesprekken gevoerd met softwareprofessionals en projectmanagers, is er een functiepunten analyse uitgevoerd door een ervaren FP-analyst bij twee projecten en is de code van twee projecten geanalyseerd. Twee teams zijn opgedeeld in de groepen exceptioneel en niet-exceptioneel en er is een vragenlijst opgesteld die bij deze groepen is afgenomen. De cijfermatige onderbouwing was beperkt tot de tijdens het onderzoek verzamelde gegevens; door de beperking in beschikbare gegevens was dit het maximaal haalbare.

De gedetailleerde registratie van de werkwijze van professionals was moeilijker dan verwacht. Het was moeilijk voor de professionals om te werken met de dagboekmethode. Dit was een activiteit die tijdens het werk aandacht vroeg, en zonder aanwezigheid van de onderzoeker. Het was lastig om gedetailleerde gegevens te bemachtigen op deze wijze. Professionals hadden, ook na herinneringen geen of weinig gegevens geregistreerd. Daarom zijn andere onderzoeksmanieren gekozen, waarbij gesprekken en observatie meer opleverde.

Het instrument dat gekozen was voor het maken van de groepen exceptioneel en niet-exceptioneel bleek geschikt. De resultaten van dit meetinstrument correspondeerde exact met de indeling die de projectmanagers hadden gemaakt. De exploratieve manier van onderzoeken was geschikt voor het doel. Er is een breed inzicht opgebouwd van de complexe materie rond het proces van de softwarefactory.

Literatuurlijst

Acuña, S. T., Juristo, N., & Moreno, A. M. (2004). Assigning people to roles in Software Projects. *Software, Practice and Experience* .

Acuña, S. T., Juristo, N., & Moreno, A. M. (2006). Emphasizing Human Capabilities in Software Development. *IEEE Software* , 94-101.

Acuña, S. T., Juristo, N., & Moreno, A. M. (2003). Modelling Human Competencies in the Software Process. *ProSim* .

Adler, P. S., & Broys, B. (1996). Two Types of Bureaucracy: Enabling and Coercive. *Administrative Science Quarterly* .

Bach, J. (1995). Enough about process: what we need are heroes. *IEEE Software* , 96-98.

Bach, J. (1994). The Immaturity of CMM. *American Programmer* .

Badddoo, N., Hall, T., & Wilson, D. (2000). Implementing a people focused SPI programme.

Bandinelli, S., & Fuggetta, A. (1995). Modelling and Improving an Industrial Software Process. *IEEE Transactions on Software Engineering* , 440-454 .

Börjesson, A., & Mathiassen, L. (2003). Making SPI Happen: The IDEAL Distribution of Effort. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)* (p. 328). IEEE Computer Society.

Brooks, F. P. (1995). *The Mythical Man Month.* Addison Wesley Longman.

Capretz, L. F. (2002). Personality types in software engineering. *Elsevier Science* .

Caputo, K. (1998). *CMM implementation guide.* Boston, USA: Addison-Wesley.

Chroust, G. (2002). Soft Factors impeding the Adoption of Process Models. *28 th Euromicro Conference (EUROMICRO'02)* (p. 388). IEEE.

Conradi, R., & Fugetta, A. (2002). Improving Software Improvement. *IEEE Software* , 92--99.

Cusumano, M. A. (1991). Factory Concepts and Practices in Software Development. *IEEE Annals of the History of Computing* , 3-32.

Davenport, T. (1993). *Process Innovation: Reengineering work through information technology.* Boston: Harvard Business School Press.

Diaz, M., & Sligo, J. (1997). How Software Process Improvement Helped Motorola. *IEEE Software* , 75-81.

Erez, M., Earley, P. C., & Hulin, C. L. (1986). The impact of participation on goal acceptance and performance: A two-step model. *Acadamy of Management Journal* , 50.

Fitzgerald, B. (1997). The Use of Systems Development Methodologies in Practice: a Field Study. *Information Systems Journal* .

Flanagan, J. C. (1954). The Critical Incident Technique. *Psychological Bulletin* .

Fowler, F. J. (1992). How Unclear Terms Affect Survey Data. *The Public Opinion Quarterly* .

Frankel, D. S. (2004). Business Process Platforms and Software Factories. *SAP Labs* .

Fuggetta, A. (2000). Software Process: A roadmap. *Proceedings of the Conference on The Future of Software Engineering* (pp. 25 - 34). Limerick, Ireland: ACM Press.

Gallivan, M. J. (2002). The influence of software developers' creative style on their attitudes to and assimilation of a software process innovation. *Elsevier Science* .

Gendall, P., & Hoek, J. (1990). A Question Of Wording. *Marketing Bulletin* .

Greenfield, J., & Short, K. (2003). Software Factories-Assembling Applications with Patterns, Models, Frameworks and Tools. *Microsoft* .

Heemstra, F. J., Kusters, R. J., & Trienekens, J. J. (2001). *Softwarekwaliteit; Op weg naar betere software.* Den Haag: Hagen & Stam uitgevers.

Herbsleb, J. D., & Goldenson, D. R. (1996). A systematic survey of CMM experience and results. *18th International Conference on Software Engineering (ICSE'96)* (p. 323). IEEE.

Humprey, W. S., Snyder, T. R., & Willis, R. R. (1991). Software Process Improvement at Hughes Aircraft. *IEEE Software* , 11-23.

IEEE Computer Society Professionals Practices Committee. (2004). *Guide to the Software Engineering Body of Knowledge.* Los Alamitos, California: IEEE Computer Society.

Johnson, R. B. (1997). Examining the validity structure of qualitative research. *University South Alabama* .

Kalton, G., & Schuman, H. (1980). The effect of the question on survey responses: a review. *University of Michigan* .

Kerth, N. L. (1998). Call for the rational use of personality indicators. *Computer* .

Kitchenham, B. A., & Pfleeger, S. L. (2002). Principles of Survey Research. Part 3: Constructing a Survey Instrument. *ACM SIGSOFT Software Engineering Notes* , 20 - 24.

Krosnichk, J. A. (1999). Survey Research.

Kruchten, P. (2002). Tutorial: introduction to the rational unified process. *Proceedings of the 24th International Conference on Software Engineering* (p. 703). Orlando, Florida: ACM Press.

Lehman. (1988). Some reservations on software process programming. *Proceedings of the 4th international software process workshop on Representing and enacting the software process* (pp. 111 - 112). Devon, United Kingdom: ACM Press.

Lehman, M. (1987). Process Models, Process Programs, Programming Support. *Proceedings of the 9th international conference on Software Engineering* (pp. 14-16). Monterey, California, United States: IEEE Computer Society Press.

Lonchamp, J. (1993). A Structured Conceptual and Terminological Framework for Software Process Engineering. *Second International Conference on the Software Process* (pp. 41-53). Berlin, Germany: IEEE.

Maravelias, C. (2003). Post-bureaucracy – control through professional freedom. *Journal of Organisational change / Emerald Insight* .

McLean, E. (1973). Empirical studies of management information systems. *DataBase* , 172-180.

Neema, S., Scott, J., & Karsai, G. (2005). Architecture Analysis in Software Factories. *Vanderbilt university/Nashville* .

NESMA. (1996). *Definities en telrichtlijnen voor de toepassing van functiepuntanalyse v2.0.* Amsterdam.

Niazi, M., Wilson, D., & Zowghi, D. (2003). A maturity model for the implementation of software process improvement: an empirical study. *Elsevier Science* .

O' Hara, F. (2000). European Experiences with Software Process Improvement. *Proceedings of the 22nd international conference on Software engineering* (pp. 635 - 640). Limerick, Ireland: ACM Press.

Osterweil, L. J. (2003). Understanding process and the Quest for Deeper Questions in Software Engineering Research. *ACM SIGSOFT Software Engineering Notes* , 6 - 14 .

Osterweil, L. (1987). Software Processes are Software Too. *Proceedings of the 9th international conference on Software Engineering* (pp. 2 - 13). Monterey, California, United States: IEEE Computer Society Press.

Pruijt, H. (2000). Repainting, modifying, smashing Taylorism. *Erasmus Univerity, Journal of Organisational Change* .

Rada, R. C. (2000). Sharing standards: Standardizing Software Projects. *Communications of the ACM* , 21 - 25.

Rainer, A., & Hall, T. (2001). *An analysis of some core studies of Software Process Improvement.* Hatfield, Hertfordshire, United Kingdom: John Wiley & Sons.

Regio, M., & Greenfield, J. (2005). Designing and Implementing an HL7 Software Factory. *International Workshop on Software Factories.* Montréal, Canada.

Rutherfoord, R. H. (2001). Using Personality Inventories to Help Form Teams for Software Engineering Class Projects. *Proceedings of the 6th annual conference on Innovation and technology in computer science education* (pp. 73 - 76). Canterbury, United Kingdom: ACM Press.

Scacchi, W. (2001). Process Models in Software Engineering. *Encyclopedia of software engineering* .

Singh, R. (1995). International standard ISO/IEC 12207. *Computer* , 89-90.

Sommerville, I., & Rodden, T. (1996). Human, Social and Organisational Influences on the Software Process. *Lancaster University* .

Sommerville, I., Sawyer, P., & Viller, S. (1999). Managing Process Inconsistency Using Viewpoints. *IEEE Transactions on Software Engineering* , 784-799.

Staudenmayer, N. (1996). Moving away from a Hacker vs Disciplined-Based Organisation Legacy – An Organisation Theory Perspective on Software Process. *10th International Software Process Workshop (ISPW '96)* (p. 104). IEEE.

Stitt-Gohdes, W. L., Lambrecht, J. J., & Redmann, D. H. (2000). The critical-incident technique in job behavior research. *Journal of Vocational Educational Research* , 63-89.

Turley, R. T., & Bieman, J. M. (1995). Competencies of Exceptional and Nonexceptional Software Engineers. *Elsevier Science* .

Van Dale. (2005). *Van Dale Hedendaags Nederlands.*

Weinberg, G. M. (1998). *The psychology of computer programming.* Dorset House Publishing.

Yamamura, G. (1999). Process Improvement Satisfies Employees. *IEEE Software* , 83-85.

Index

Bijlage A: De Q-Sort (Turley & Bieman, 1995)

DEFINITION I value the synergy of group efforts and invest the effort required to create group solutions, even at the expense of my individual results. **KEY BEHAVIORS** ✓ I balance the strengths and weaknesses of other team members. ✓ I promote constant communication among team members using techniques such as brainstorming sessions, travel, phone calls, e-mail, or just being physically close to the rest of the team. ✓ I recognize synergy of group efforts and invest personal time and energy to leverage it. *Item#1*	**DEFINITION** I use a prototyping method to assess key system parameters before designing the final product. I avoid using my prototypes as final implementations. **KEY BEHAVIORS** ✓ I use prototypes as a mechanism for incremental development of a product. ✓ I attempt to not allow my prototypes to become the final product. ✓ I use prototypes to assess critical areas like performance and time estimates. ✓ I prototype in parallel with the detailed design phase. *Item #4*
DEFINITION I proactively seek the assistance of others in learning, researching, designing, understanding, debugging, or checking results. **KEY BEHAVIORS** ✓ I ask previous implementers to explain their designs. ✓ I ask other engineers to critique or evaluate my designs. ✓ I survey others to create lists of alternatives. *Item #2*	**DEFINITION** I apply incremental testing techniques during code development such that a given module achieves a high degree of reliability by the tune it's completed. **KEY BEHAVIORS** ✓ I create tests in parallel with the creation of code. ✓ I automate tests so they can be run frequently throughout design. ✓ I test frequently during development to ensure reliability at the module level. *Item #5*
DEFINITION I spend a significant amount of time assisting others in the completion of their tasks or influencing broad organizational direction. **KEY BEHAVIORS** ✓ I act as a lab-wide consultant for process or product issues. ✓ I review, direct, or influence the work of other engineers. ✓ I assist other engineers with their tasks in an effort to complete the project. ✓ I teach engineering skills to other engineers. *Item #3*	**DEFINITION** At the time of assignment. I possess the unique skills or knowledge required to accomplish the task at hand. **KEY BEHAVIORS** ✓ I possess the necessary domain knowledge and skills required for the job. ✓ Prior to an assignment, and on my own initiative, I become an expert in a given area. *Item #6*

DEFINITION I actively seek the training needed to complete the assigned task. **KEY BEHAVIORS** ✓ I seek out documentation required to understand my current assignment. ✓ I take classes which will directly help in the completion of the current assignment. ✓ I keep current by reading trade or technical journals. ✓ I improve my skills and awareness by attending conferences. *Item #7*	**DEFINITION** I use a methodical approach in understanding and solving problems. **KEY BEHAVIORS** ✓ I build mental or physical system models to enhance my understanding and visualization of the problem. ✓ I design well controlled experiments to efficiently resolve problems. ✓ I invest in the development of test tools to solve problems. *Item #10*
DEFINITION I proactively attempt to leverage other engineers' effort by using their code or designs. I attempt to leverage my own effort by making my code reusable. **KEY BEHAVIORS** ✓ I look for code or code fragments which can be reused, ol design and code so that my effort can be reused. *Item #8*	**DEFINITION** I seek to improve performance or results through the use of new tools or methods. **KEY BEHAVIORS** ✓ I proactively seek out new tools or methods to solve problems. ✓ I use my work assignment as a way to learn new tools or methods. ✓ I recognize value in new tools or techniques. *Item #11*
DEFINITION I take advantage of the tools and techniques of structured design in order to understand and communicate designs, but do not necessarily follow the complete formalism of the approach. **KEY BEHAVIORS** ✓ I use structured techniques (such as Structured Analysis and Design, Hierarchy Charts,…) as a means of joint development and communication. ✓ I use structured techniques as a mechanism for passing off a design or part of a design to another engineer for implementation. ✓ I view structured techniques as just another tool which can be applied to certain problems, rather than as a panacea. *Item #9*	**DEFINITION** I show a strong concern for schedules and I estimate schedules well. **KEY BEHAVIORS** ✓ I maintain personal "rules of thumb" for schedule estimation. ✓ I refine schedule estimates based on my measured progress. ✓ I schedule via task breakdown and successive refinement. ✓ I meet schedules. *Item #12*

DEFINITION I use code reading and other group development techniques to ensure final code quality. **KEY BEHAVIORS** ✓ I participate in the code reading of other engineers´ work. ✓ I ask for others to code read my work. ✓ I participate in brainstorming and other group development techniques. *Kern #13*	**DEFINITION** In response to schedule pressure. I am forced to sacrifice important parts of the design process. **KEY BEHAVIORS** ✓ In response to schedule pressure I am forced to provide incomplete documentation. ✓ When schedules slip, I do not have time to adequately inspect or test the product. ✓ When pushed to pull up a schedule, I will not prototype or adequately design risky parts of product. *Item #16*
DEFINITION I use decomposition design techniques relying on visual representation of designs. I create structured designs, generally without using formal techniques. **KEY BEHAVIORS** ✓ I keep design specification in sync with the actual implementation. ✓ I follow a *top-down* design method using decomposition to successively refine the design. ✓ I take a modular approach in order to reuse as much of the design as possible. ✓ I use pictures to communicate and understand designs. *Item #14*	**DEFINITION** I create solutions which are elegant and simple and allow for easy extension to future needs. **KEY BEHAVIORS** ✓ I value simple solutions. ✓ I design general solutions which will be easily extended, even if it's not currently needed. ✓ I apply structure to ill defined problems and problem domains. *Item #17*
DEFINITION I consider customer or user input and feedback to be an essential ingredient in the design of products. **KEY BEHAVIORS** ✓ I proactively attempt to obtain customer and user input and feedback for products. ✓ I measure project success in terms of customer satisfaction. *Item #15*	**DEFINITION** I take pride in producing defect free products on schedule in minimum time. **KEY BEHAVIORS** ✓ I take pride in meeting or beating schedules. ✓ I take pride in achieving low defect counts. ✓ I take pride in achieving high productivity and accomplishing significant amounts of work over short periods of time. *Item #18*

DEFINITION I take the initiative to identify ways of completing important tasks. I influence others to consider alternative approaches. **KEY BEHAVIORS** ✓ I proactively complete projects or tasks that I consider important. ✓ I influence others in design, organizational structure, … ✓ I identify ways to surmount barriers and remove obstacles *Item #19*	**DEFINITION** I enjoy the challenge of the assignment and the sense of accomplishment from completing it. I just plain have fun at work. **KEY BEHAVIORS** ✓ I look forward to going to work. ✓ I derive a sense of accomplishment from work. ✓ I enjoy the challenge of a tough assignment. ✓ I am driven by the reward of doing something new and different. *Item #22*
DEFINITION I proactively attempt to influence project direction by influencing management. **KEY BEHAVIORS** ✓ I discuss issues concerning other engineers with my manager. ✓ I attempt to set project direction and make project decisions by influencing my manager. ✓ I make specific resource or assignment recom-mendations to management. ✓ I promote product ideas through demos or selling of ideas to management. *Item #20*	**DEFINITION** I am driven by a sense of mission and clearly articulated goals to achieve a specific result. **KEY BEHAVIORS** ✓ I create and articulate clear and specific goal statements. ✓ I drive the project to achieve specific goals. *Item #23*
DEFINITION I value the sense of accomplishment which comes from making a direct contribution. **KEY BEHAVIORS** ✓ I seek assignments where I can contribute. ✓ I feel rewarded by the chance to contribute. *Item #21*	**DEFINITION** I stress the solution over the source of the solution. I don't care where a good idea comes from and don't feel the need to promote my own ideas. **KEY BEHAVIORS** ✓ I focus on the end result, regardless of who creates the solution. ✓ I allow use of a *discovery process* for others to come to see the value of my ideas. ✓ I discuss ideas, not positions. ✓ I allow others to re-write an idea I've created. *Item#24*

DEFINITION I exhibit and articulate strong beliefs and convictions. I act in accordance with these beliefs, even when it is counter to specific management direction. **KEY BEHAVIORS** ✓ I act in accordance with my beliefs rather than solely based on my assignment. ✓ I risk my performance ranking in an effort to secure the best solution. ✓ I argue forcefully for a specific point of view. *Item #25*	**DEFINITION** I am thorough in my assignment and persevere until it's completed. **KEY BEHAVIORS** ✓ I drive hard to complete even the tedious parts of my assignment. ✓ I like to see things done cleanly and correctly. ✓ I stick with assignments even when it's not clear that they're going anywhere. At least I'm eliminating things and making the problem smaller. *Item #28*
DEFINITION I mix my personal and work goals by seeking or tailoring assignments to my professional interests. **KEY BEHAVIORS** ✓ I identify positions I would like to have and lobby to receive them. ✓ I seek assignments which will further my professional development. ✓ I identify technical areas which I'd like to develop and find ways to apply them to the project at hand. *Item #26*	**DEFINITION** I have mastered the skills and techniques necessary for good software design and implementation. **KEY BEHAVIORS** ✓ I have a strong technical and software development background. ✓ I am comfortable with multiple software design and implementation techniques. ✓ I have very strong software development skills. *Item #29*

DEFINITION I confront others when necessary to ensure a good design or product solution. **KEY BEHAVIORS** ✓ Rather than letting a conflict simmer, I will openly confront another person in an effort to resolve it. ✓ I will raise a tough issue of conflict with another engineer to my manager in an effort to have it resolved. *Item #27*	**DEFINITION** I posses strong analytic skills that allow me to visualize a complex problem and create alternative solutions **KEY BEHAVIORS** ✓ I am able to see basic theory and basic structure in a problem. ✓ I am able to visualize what's going on inside a complex system. ✓ I am able to break a large, complex problem into smaller, more manageable chunks. *Item #30*
DEFINITION I am driven by a strong bias for action and sense of urgency in completing my assignments. **KEY BEHAVIORS** ✓ When faced with a tough problem, I don't hesitate to get started. I develop the required capability as I go. ✓ I am results oriented and want to make progress on a regular basis. ✓ I push myself to achieve results quickly. *Item #31*	**DEFINITION** I am innovative in my solutions to problems. **KEY BEHAVIORS** ✓ I like to create alternatives which are both creative and practical. ✓ I have creative ideas and solutions to problems. *Item #34*

DEFINITION I am detail oriented and able to deal with very complex problems. **KEY BEHAVIORS** ✓ I´m good at taking details from a lot of different sources and determining a good solution to a problem. ✓ I keep track of lots of detail, either in my head or on paper. ✓ I concern myself with "corner cases" and other seemingly insignificant data, since this is often where the breakthrough comes from. *Item #32*	**DEFINITION** My prior experience with similar projects leads to my high performance on current projects. **KEY BEHAVIORS** ✓ I use the experience gained on one project to improve my solutions on subsequent projects. ✓ I find that the skills I learn on one project are directly applicable to my next project. *Item #35*
DEFINITION I am very methodical, organized, and cautious in my work. **KEY BEHAVIORS** ✓ I make sure that all paths are covered in my design and problem solving. ✓ I work slowly and carefully to avoid making mistakes. *Item #33*	**DEFINITION** I set high personal expectations and goals. I am not satisfied with the status quo. **KEY BEHAVIORS** ✓ I am constantly looking for better tools, better technologies, or better problem solving approaches. ✓ I give myself time to improve. *Item #36*
DEFINITION I have a high concern for reliability and a strong commitment to quality. **KEY BEHAVIORS** ✓ I make sure that my products have few, if any, defects. ✓ I work hard to be sure that my results are clear and readable to others. *Item #37*	**DEFINITION** I maintain a broad "big picture" view of my projects in an attempt to influence the project direction. **KEY BEHAVIORS** ✓ I remain aware of what other engineers are doing and suggest ways to better achieve project objectives. ✓ I try to be sure that project goals make sense, and work to change them if they don't. ✓ I try to fit my project into the broader scheme of division programs. *Item #38*

Bijlage B: Data uit de Q-Sort

Professional #1

Tabel 13: Gegevens uit de Q-Sort (de nummers corresponderen met de Q-Cards)

SELF												
Most												
6	2	7										
5	31	9	10	37								
4	30	27	36	28	34	18	22					
3	6	35	15	24	23	17	14	13	4	12	20	29
2	8	5	1	19	11	38	26					
1	3	21	33	32								
0	25	16										
Least												
TEAMMEMBER												
Most												
4	2											
3	21	38										
2	16	20	25									
1	29	27										
0	3											
Least												
MANAGER												
Most												
4	16											
3	21	2										
2	20	27	25									
1	3	29										
0	38											
Least												

Tabel 14: Vertaling van de Q-Sort naar conclusie exceptioneel of niet-exceptioneel

Item	Competency	Self	Team	Manager	VAR	E score	NE score
1	Team oriented	-1					
2	Seeks help from others	3	2	1	1		3
3	Helps others	-2	-2	-1	0,333333	-2	
4	Uses prototyping to asses de	0					
5	Write automated tests with cc	-1					
6	Possesses Unique Knowledge	0					
7	Obtains Necessary Training	3					
8	Leverages/Reuses Code	-1					
9	Uses structured techniques fo	2					
10	Uses methodical problem solv	2					
11	Uses new tools or methods	-1					
12	Schedules and estimates well	0					
13	Uses code reading to ensure t	0					
14	Uses decomposition design te	0					
15	Focuses on user or customer	0					
16	Responds to schedule pressu	-3	0	2	6,333333		x
17	Emphasizes Elegant and Sim	0					
18	Takes Pride in Quality and Pr	1					
19	Takes Initiative to Identify Way	-1					
20	Proactively attempts to influer	0	0	0	0	0	
21	Driven by desire to contribute	-2	1	1	3		x
22	Enjoys Challenge of Assignm	1					
23	Driven by a sense of mission	0					
24	Stresses Soltuion over Source	0					
25	Exhibits and articulates strong	-3	0	0	3		
26	Mixes personal and work goal	-1					
27	Willingness to Confront Other	1	-1	0	1		1
28	Perseverance	1					
29	Mastery of skills and techniqu	0	-1	-1	0,333333	0	
30	Strong analytic skills, ability t	1					
31	Driven by bias for action and s	2					
32	Pays close attention to detail	-2					
33	Methodical, Organized and Ca	-2					
34	Innovative	1					
35	Prior experience	0					
36	High personal expectations ar	1					
37	Concern for reliability and qua	2					
38	Maintains ´big picture´ view	-1	1	-2	2,333333	-1	
		0	0	0		-3	4
						conclusie:	NE

Professional #2

Tabel 15: Gegevens uit de Q-Sort (de nummers corresponderen met de Q-Cards)

SELF												
Most												
6	3	25										
5	29	30	35	38								
4	36	17	31	9	6	8	22					
3	19	34	20	33	16	37	28	32	21	11	26	15
2	5	1	18	27	4	13	23					
1	24	14	2	10								
0	12	7										
Least												
TEAMMEMBER												
Most												
4	29											
3	25	38										
2	20	3	27									
1	21	16										
0	2											
Least												
MANAGER												
Most												
4	38											
3	29	3										
2	27	20	16									
1	25	2										
0	21											
Least												

Tabel 16: Vertaling van de Q-Sort naar conclusie exceptioneel of niet-exceptioneel

Item	Competency	Self	Team	Manager	VARIANTIE	E score	NE score
1	Team oriented	-1					
2	Seeks help from others	-2	-2	-1	0,333333		-2
3	Helps others	3	0	1	2,333333	3	
4	Uses prototyping to asses	-1					
5	Write automated tests with	-1					
6	Possesses Unique Knowle	1					
7	Obtains Necessary Training	-3					
8	Leverages/Reuses Code	1					
9	Uses structured techniques	1					
10	Uses methodical problem s	-2					
11	Uses new tools or methods	0					
12	Schedules and estimates w	-3					
13	Uses code reading to ensu	-1					
14	Uses decomposition desigr	-2					
15	Focuses on user or custom	0					
16	Responds to schedule pres	0	-1	0	0,333333		0
17	Emphasizes Elegant and S	1					
18	Takes Pride in Quality and	-1					
19	Takes Initiative to Identify W	0					
20	Proactively attempts to influ	0	0	0	0	0	
21	Driven by desire to contribu	0	-1	-2	1		0
22	Enjoys Challenge of Assigr	1					
23	Driven by a sense of missic	-1					
24	Stresses Soltuion over Sou	-2					
25	Exhibits and articulates str	3	1	-1	4	x	x
26	Mixes personal and work g	0					
27	Willingness to Confront Oth	-1	0	0	0,333333		-1
28	Perseverance	0					
29	Mastery of skills and techn	2	2	1	0,333333	2	
30	Strong analytic skills, abilit	2					
31	Driven by bias for action an	1					
32	Pays close attention to det	0					
33	Methodical, Organized and	0					
34	Innovative	0					
35	Prior experience	2					
36	High personal expectations	1					
37	Concern for reliability and q	0					
38	Maintains ´big picture´ view	2	1	2	0,333333	2	
		0	0	0		7	-3
						conclusie: E	

Bijlage C: De functiepunten analyse

Functiepunten analyse van project in Product Centre

Functiepunten telling

Onderstaande analyse is op enkele punten onherkenbaar gemaakt, zoals is afgesproken met het deelnemende team.

		Product FP				
Ref. ontwerp	Voeg FPA toe / Voeg kopje in / Naam	Type	DET	RET	Comple-xiteit	FP
Logische gegevensverzamelingen						
	[redacted]	KGV	50	14	M	10
	Relatie (o.a. Relatie, Adres, Contactpersoon)	KGV	15	6	G	7
Functies						
	Importeren [redacted]					
	Importeren [redacted] (import xml bestand - inserts en updates)	IF	70	14	M	6
	Selecteren importbestand	IF	3	1	E	3
	Overzicht financieel administrateur	UF	50	9	M	7
	Opvragen					
	[redacted] webservice	UF	27	9	M	7
	FPA Tabellen					
	Totaal bruto functiepunten					40
	Correctiefactor					0,81
	Totaal netto functiepunten					32

Aannames

Referentie	Omschrijving
	De gebruikte entiteiten zijn als KGV's geteld. Ze bevinden zich in hetzelfde systeem maar worden in het getelde onderdeel alleen gebruikt en niet onderhouden.
7.2.n & 7.2.o & 7.2.p	Het dynamisch weergeven van de vragen vanuit het door de manager gedefinieerde model is niet geteld als een afzonderlijke opvraagfunctie voor het betreffende elementtype, maar wordt gezien als onderdeel van de definitie en beantwoording invoerfuncties.

Systeemkenmerken

1	Datacommunicatie	3
2	Gedistribueerde gegevensverwerking	1
3	Prestatiedoeleinden	1
4	Produktiesysteem	1
5	Transactievolume	2
6	On-line data-entry	0
7	Gebruikersdoelmatigheid	2
8	On-line update	2
9	Complexe interne verwerking	3
10	Produceren van herbruikbare code	1
11	Installatiegemak	0
12	Bedieningsgemak	0
13	Meerdere lokaties/organisaties	0
14	Flexibiliteit	0
		16

Functiepunten analyse van project buiten Product Centre

FP telling

Ref. ontwerp	Voeg FPA toe / Voeg kopje in Naam	Type	DET	RET	Comple-xiteit	FP
Logische gegevensverzamelingen						
	Project (o.a. Project, Publication, Element, Game, MultipleChoiceQuestion)	KGV	30	10	M	10
	Player (o.a. Player, PlayerProfileData)	KGV	15	3	E	5
Functies						
	Speler					
	Beantwoorden enquete vraag	IF	4	2	E	3
	Bestand uploaden	IF	2	1	E	3
	Beantwoorden foto-vraag	IF	4	1	E	3
	Invoeren klant editie	IF	5	1	E	3
	Samenvatting beantwoorde vragen	UF	18	7	M	7
	Rapport opvragen	OF	2	1	E	3
	Manager (onderhouden meta model)					
	Definieren enquete vraag	IF	6	1	E	3
	Definieren upload pagina	IF	10	1	E	3
	Definieren foto-vraag	IF	15	2	G	4
	Definieren samenvatting pagina	IF	4	1	E	3
	Spelvolgorde instellen	IF	4	1	E	3
	Tracking spelers	UF	6	3	G	5
	Overnemen sessie andere klant	OF	2	1	E	3
	Totaal bruto functiepunten					61
	Correctiefactor					0,88
	Totaal netto functiepunten					54

Aannames

Referentie	Omschrijving
	De gebruikte entiteiten zijn als KGV's geteld. Ze bevinden zich in hetzelfde systeem maar worden in het getelde onderdeel alleen gebruikt en niet onderhouden.

Systeemkenmerken

1	Datacommunicatie	3
2	Gedistribueerde gegevensverwerking	1
3	Prestatiedoeleinden	1
4	Produktiesysteem	1
5	Transactievolume	2
6	On-line data-entry	5
7	Gebruikersdoelmatigheid	3
8	On-line update	2
9	Complexe interne verwerking	2
10	Produceren van herbruikbare code	1
11	Installatiegemak	0
12	Bedieningsgemak	0
13	Meerdere lokaties/organisaties	0
14	Flexibiliteit	2
		23

Bijlage D: De vragenlijst

1. Bij het maken van een datamodel maak ik gebruik van een checklist uit de factory.
Volledig juist Juist Onjuist Volledig onjuist

2. Binnen mijn projectteam is voor de start van het project afgesproken welke onderdelen uit het factory model gebruikt worden binnen het project.
Volledig juist Juist Onjuist Volledig onjuist

3. Ik gebruik zelf gekozen tools omdat ik daarmee tot een beter resultaat kom dan met de standaard factory tools.
Volledig juist Juist Onjuist Volledig onjuist

4. Bij het gebruik van tooling heb ik gebruik gemaakt van tool mentors van de factory.
Volledig juist Juist Onjuist Volledig onjuist

5. Binnen de gestandaardiseerde aanpak heb ik voldoende vrijheid om zelf problemen op de best mogelijke manier op te lossen.
Volledig juist Juist Onjuist Volledig onjuist

6. Ik maak geen gebruik van de checklists van de factory.
Volledig juist Juist Onjuist Volledig onjuist

7. Ik zou sommige dingen graag anders aanpakken, maar dit kan niet vanwege de standaard werkwijze.
Volledig juist Juist Onjuist Volledig onjuist

8. In de praktijk ben ik vrij om mijn eigen werkwijze te bepalen.
Volledig juist Juist Onjuist Volledig onjuist

9. Deze vraag gaat over wat jij doet om goede kwaliteit programmacode te produceren, die op tijd klaar is. We stellen deze vraag aan alle deelnemers aan deze enquête. Hoe zorg je ervoor dat jij op tijd goede code oplevert? (open vraag)

10. Als ik een goed onderbouwd idee heb dan wordt de gestandaardiseerde werkwijze binnen de factory daarop aangepast.
Volledig juist Juist Onjuist Volledig onjuist

11. Ik zie verbeterpunten voor de standaard werkwijze binnen de factory.
Volledig juist Juist Onjuist Volledig onjuist

12. Op welke drie gebieden valt qua werkwijze het meeste te verbeteren?

13. In de praktijk heb ik invloed op de werkwijze van mijn projectteam.
Volledig juist Juist Onjuist Volledig onjuist

14. Ik zou graag meer invloed hebben op mijn eigen werkwijze dan ik op dit moment heb door de gestandaardiseerde werkwijze.
Volledig juist Juist Onjuist Volledig onjuist

15. Ik gebruik tools of templates uit de factory bij mijn werkzaamheden.
Volledig juist Juist Onjuist Volledig onjuist

16. Indien je factory tools of templates gebruikt, welke tools of templates zijn dit?

17. Ik vind het belangrijk mijn eigen hoofdstukindeling te kunnen gebruiken in mijn documenten.
Volledig juist Juist Onjuist Volledig onjuist

18. Bij het maken van een architectuurdocument maak ik gebruik van de templates uit de factory.
Volledig juist Juist Onjuist Volledig onjuist

19. Ik gebruik zelf gebouwde of zelf gekozen tools of templates bij mijn werkzaamheden.
Volledig juist Juist Onjuist Volledig onjuist

20. Indien je zelf gebouwde of zelf gekozen tools of templates gebruikt, welke tools of templates zijn dit?

21. Hoe waardevol zijn de factory tools of templates voor het ontwikkelproces?
Op een schaal van 1 tot 5 (5=zeer bruikbaar, 1= niet bruikbaar)

22. Ik zou graag een eigen tool gebruiken, maar dit kan niet vanwege standaard tools die gebruikt worden.
Volledig juist Juist Onjuist Volledig onjuist

23. Als ik een eigen tool wil gebruiken in plaats van een standaard factory tool, dan is dit geen probleem.
Volledig juist Juist Onjuist Volledig onjuist

24. Ik gebruik eigen templates omdat ik daarmee tot een beter resultaat kom dan met de factory templates
Volledig juist Juist Onjuist Volledig onjuist

25. Hoeveel tijd besteed je in verhouding /verwacht je in verhouding te besteden aan verschillende activiteiten, binnen het huidige project? (in totaal 100%) Een schatting is voldoende.

1) Identify design elements : %
2) Use-cases analysis : %
3) Describe user interface : %
4) Prototype user interface : %
5) Review design : %
6) Class and operation design : %
7) Design database : %
8) Analyse runtime behaviour : %
9) Developer test : %
10) Review code : %
11) Fix bugs : %
12) Change requests : %
13) Documentatie : %
14) Overig: _____ : %

26. Voor hoeveel procent van de projecten sluit de standaard werkwijze van de factory niet goed aan?
%

27. Willen we meer projecten kunnen uitvoeren dan moet de standaard werkwijze veel flexibeler worden / moet er minder worden gestandaardiseerd.
Volledig juist Juist Onjuist Volledig onjuist

28. Ik raadpleeg de factory werkwijze (het ontwikkelmodel) bij 3 van de 5 projecten.
Volledig juist Juist Onjuist Volledig onjuist

29. Elk project heeft een eigen unieke werkwijze nodig.
Volledig juist Juist Onjuist Volledig onjuist

30. Ik ben bereid om vrijheid in het bepalen van mijn werkwijze in te leveren, indien de factory hierdoor efficiënter kan werken.
Volledig juist Juist Onjuist Volledig onjuist

31. Er zijn dingen gestandaardiseerd die goed waren voor projecten uit het verleden, maar deze zijn niet geschikt voor huidige projecten.
Volledig juist Juist Onjuist Volledig onjuist

32. Door het werken binnen deze gestandaardiseerde omgeving ben ik een betere software engineer geworden.
Volledig juist Juist Onjuist Volledig onjuist

33. De standaard omgeving beperkt mij in mijn ontwikkeling als software engineer.
Volledig juist Juist Onjuist Volledig onjuist

34. Ik ken problemen bij projecten die veroorzaakt zijn door onze standaard werkwijze.
Volledig juist Juist Onjuist Volledig onjuist

35. Op basis van mijn ideeën / aanbevelingen is de gestandaardiseerde werkwijze binnen de factory aangepast.
Volledig juist Juist Onjuist Volledig onjuist

36. Ik zou meer ruimte willen om zelf mijn werk in te richten.
Volledig juist Juist Onjuist Volledig onjuist

37. Ik hoef geen zaken uit te voeren als het nut daarvan voor mij niet duidelijk is.
Volledig juist Juist Onjuist Volledig onjuist

38. Er is onvoldoende ruimte voor creativiteit binnen de factory.
Volledig juist Juist Onjuist Volledig onjuist

39. Productiviteit zal toenemen als iedereen zich meer aan de standaard houdt.
Volledig juist Juist Onjuist Volledig onjuist

40. Als iedereen zich aan de standaard houdt dan gaat dit ten koste van het werkplezier.
Volledig juist Juist Onjuist Volledig onjuist

41. Hoe zou je zelf je tijd verdelen, om tot een zo goed mogelijk resultaat te komen bij het huidige project? Je hebt geen eisen vanuit de klant of vanuit het management. (in totaal 100%)

1) Identify design elements : %
2) Use-cases analysis : %
3) Describe user interface : %
4) Prototype user interface : %
5) Review design : %
6) Class and operation design : %
7) Design database : %
8) Analyse runtime behaviour : %
9) Developer test : %
10) Review code : %
11) Fix bugs : %
12) Change requests : %
13) Documentatie : %
14) Overig: _____ : %

42. Als ik afwijk van de standaard werkwijze dan word ik hierop aangesproken door collega's of mijn baas.
Volledig juist Juist Onjuist Volledig onjuist

43. Ik ben projecten anders gaan uitvoeren sinds ik binnen de factory werk.
Volledig juist Juist Onjuist Volledig onjuist

44. Welke gebieden van standaardisatie ervaar je als knellend?

45. Ik ben actief bezig met verbeteren van mijn werkwijze of aanpak.
Volledig juist Juist Onjuist Volledig onjuist

46. Een standaard werkwijze is essentieel om goede producten op een concurrerende wijze te ontwikkelen.
Volledig juist Juist Onjuist Volledig onjuist