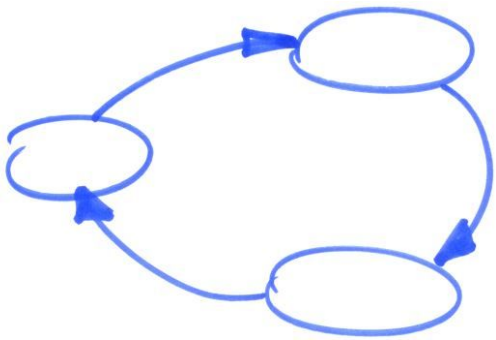
Belofte maakt schuld



Marc Evers & Willem van den Ende

17-1-2008

In dit artikel beschrijven we het "Belofte maakt Schuld" patroon aan de hand van een casus over een team dat in een vicieuze cirkel terecht komt van beloftes maken aan haar klanten, hard werkt om die beloftes waar proberen te maken en als dat niet helemaal lukt, meer beloftes maakt.

Met behulp van systeemdenken en oorzaak-gevolg diagrammen maken we de dynamiek van Belofte maakt Schuld inzichtelijk. Op deze manier kunnen we achterliggende oorzaken vinden en vicieuze cirkels doorbreken.

De casus komt uit onze eigen ervaring in organisaties die software ontwikkelen, en uit de praktijk van deelnemers aan onze workshops.

Inleiding

Heeft u wel eens...

...het gevoel geen grip op de situatie te krijgen?

...dat u probeert problemen op te lossen maar het team lijkt in een vicieuze cirkel te zitten?

...dat u eindeloos brandjes aan het blussen bent, waarbij het blussen van de ene het volgende brandje weer aan lijkt te steken?

In dit artikel schetsen we een dergelijk herkenbaar verhaal uit onze praktijk – een team dat beloftes aan haar klanten doet, op een manier waarop het bijna onmogelijk maakt ze in te lossen... Het gevolg is een neerwaartse spiraal van beloftes doen, niet nakomen, en nog meer beloftes doen om de teleurstelling bij klanten te compenseren. Uiteindelijk resulteert dit in een compleet verlies van vertrouwen en geloofwaardigheid.

We laten zien wat de dieperliggende oorzaken zijn en hoe je het probleem echt kan oplossen in plaats van alleen symptomen te bestrijden. We doen dit door te systeemdenken met behulp van oorzaak-gevolg diagrammen

Systeemdenken is een benadering waarbij een (deel van een) organisatie of een project beschouwd worden in termen van een systeem van variabelen die invloed uitoefenen op elkaar. De focus ligt op hoe de delen onderling samenhangen in plaats van lineaire oorzaak-gevolg relaties. Het gaat hierbij om dynamiek en verandering, waarbij feedback een essentiële rol speelt.

Systeemdenken is een middel om mentale modellen van verschillende betrokkenen expliciet te maken en om niet voor de hand liggende effecten en zelfversterkende, destructieve lussen te zien. Oorzaak-gevolg diagrammen maken zulke lussen zichtbaar, en vereenvoudigen het vinden van effectieve interventies.

Inhoud

Het verhaal

Er was eens... een kleine IT organisatie, bestaande uit enkele softwareontwikkelaars (Paul, Vera, Gert) en Klaas, de manager van de groep. Klaas doet ook de sales van hun product. Ze werken nu ruim een jaar aan 'hun' product, een Vernieuwend Web Systeem (VWS). Ze hebben inmiddels drie klanten (Joop, Fred & Joris). Het VWS product is deels generiek, om de onderhoudskosten te drukken. Voor elk van de klanten leveren ze ook een stuk maatwerk, omdat het team hecht aan 'customer intimacy' en graag de input en feedback van klanten gebruikt om het Vernieuwend Web Systeem verder te verbeteren en voor meer klanten aantrekkelijk te maken..

De drie klanten zijn enthousiast: ze zien de mogelijkheden van het systeem; het is voor hun context nog niet compleet. Elke klant heeft nog specifieke functionaliteit nodig om de komende jaren mee vooruit te kunnen. Ze hebben er alle drie echter vertrouwen in dat de club de functionaliteit op gaat leveren.

Omdat ze met korte releases van één maand werken zien de klanten continu voortgang, de ene keer iets meer dan de andere keer, maar het product vordert gestaag.

Wat is er nu aan de hand?

Het team heeft net een release achter de rug. Vrijwel alles wat gepland was is opgeleverd. Een nieuwe planning voor de volgende oplevering is opgesteld. Hierbij maakt het team gebruik van de ontwikkelsnelheid (velocity) die ze in de afgelopen release gemeten hebben. Ze hebben als proces afspraak om voor een volgende release niet meer features (die ze *stories* noemen) te beloven dan ze in de afgelopen release daadwerkelijk hebben kunnen bouwen. Elke feature wordt ingeschat met een aantal *storypunten*. Het totaal aantal behaalde storypunten in de afgelopen release is wat er maximaal kan worden ingepland in de volgende.

Tijdens de laatste planningsessie heeft Klaas nog geprobeerd of het team toch niet een extra feature op zich wilde nemen, maar het team hield voet bij stuk. “Laten we gewoon doen wat realistisch is, als we sneller gaan dan verwacht kunnen we altijd nog extra stories bouwen. Dat is beter dan teveel beloven en er later op terugkomen,” aldus Gert.

Ze zijn nog maar net twee dagen bezig met de geplande stories, of Klaas komt 's ochtends de ontwikkelruimte binnen. “Ik heb gisteren opnieuw met Ronald gesproken, en ik heb hem eindelijk over de streep getrokken! Klant nummer 4! Ik heb hem wel toe moeten zeggen dat we deze release al feature FR53i speciaal voor zijn organisatie bouwen, dat is zijn voorwaarde.”

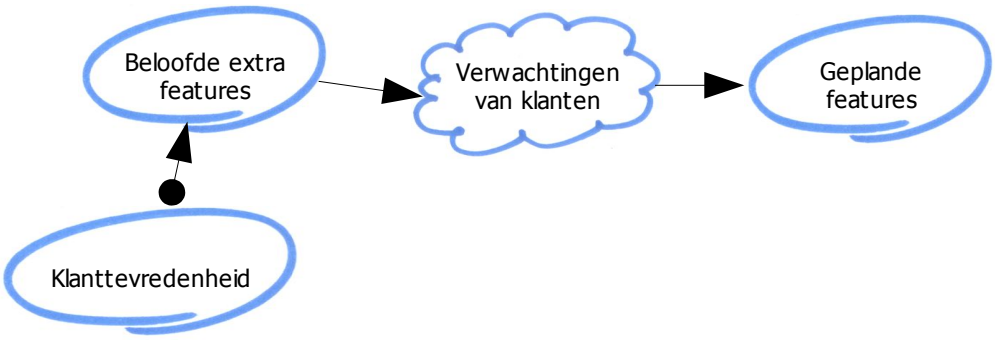
“Maar we hebben deze release al volgepland en die feature is best wat werk, toch minimaal drie extra stories erbij! Dat is...acht extra storypunten” werpt Vera tegen.

“Deze klant is van strategisch belang, Ronald is zo iemand die als hij eenmaal overtuigd is, uit zichzelf allerlei anderen enthousiast gaat maken. Dat gaat ons extra klanten en omzet opleveren. We moeten gewoon met zijn allen even stevig doorwerken deze release, dan gaat het wel lukken!”

“Dan wordt het haastwerk, ik weet niet of de kwaliteit van de code dan wel goed genoeg blijft.” zegt Paul vertwijfelt.

"Geen probleem, dan refactoren jullie gewoon wat extra de volgende release en dan komt het allemaal goed. Ik zie dat jullie het belang ervan inzien om allemaal even de schouders eronder te zetten. Afgesproken!", en Klaas loopt terug naar zijn kantoor.

Het team heeft zijn twijfels, maar is ook wel enthousiast geraakt door het vooruitzicht op de vierde klant. Ronald is inderdaad niet iemand die zich makkelijk laat overhalen. Het is al een hele prestatie dat ze zover zijn. Dus iedereen trekt er hard aan om de drie extra stories te bouwen. De druk en de werktijden lopen wel op tegen het eind van de release.



Als de klant minder tevreden wordt, besluit Klaas extra features te beloven. Die beloftes wekken hoge verwachtingen bij de klanten. Om aan die verwachtingen te voldoen, worden extra features ingepland. Bovenstaand oorzaak-gevolg diagram geeft deze dynamiek weer: de rondjes bevatten de variabelen, de pijltjes ertussen geven de causaliteit aan. Variabelen zijn dingen die we kunnen observeren of meten. Causaliteit kan in dezelfde of in de tegengestelde richting werken. Wanneer bijvoorbeeld het aantal beloofde features toeneemt, nemen ook de verwachtingen van de klant toe, of tegengesteld, als de klanttevredenheid afneemt, neemt het aantal beloofde features juist toe.

De pizza's die Klaas 's avonds laat brengt maken veel goed. Het gevoel om belangrijk werk te doen, een klant blij te maken en onderdeel van een hecht team te zijn geeft best wel een kick

Het is dan wel wat haastwerk, ze zijn niet echt blij met de kwaliteit van het werk. Gelukkig kunnen ze daar de volgende release wat aan doen, bij voorbeeld een aantal ontbrekende unittests alsnog schrijven.

Operatie geslaagd...

De release is geslaagd, met alle geplande stories en feature FR53i. Na de oplevering is iedereen in het team moe en gestrest. Klaas komt nog even de ontwikkelruimte binnen: "Goed gedaan, zie je nu wel dat jullie meer kunnen! Ik ben trots op jullie."

Bij de volgende planningssessie heeft het team moeite om het aantal geplande stories te beperken. Klaas wil graag net zoveel inplannen als ze nu opgeleverd hebben, namelijk 29 storypunten. Gert houdt voet bij stuk: "die extra hoge ontwikkelsnelheid is vertekend: we hebben wel meer opgeleverd, maar dat is niet verantwoord gebeurd - we hebben overgewerkt, unittests en refactorings laten liggen en elkaars code niet gereviewed. Ik weet niet hoe lang we dit vol kunnen houden. En Klaas, bovendien had je beloofd dat we de volgende release extra tijd zouden krijgen om het achterstallige werk

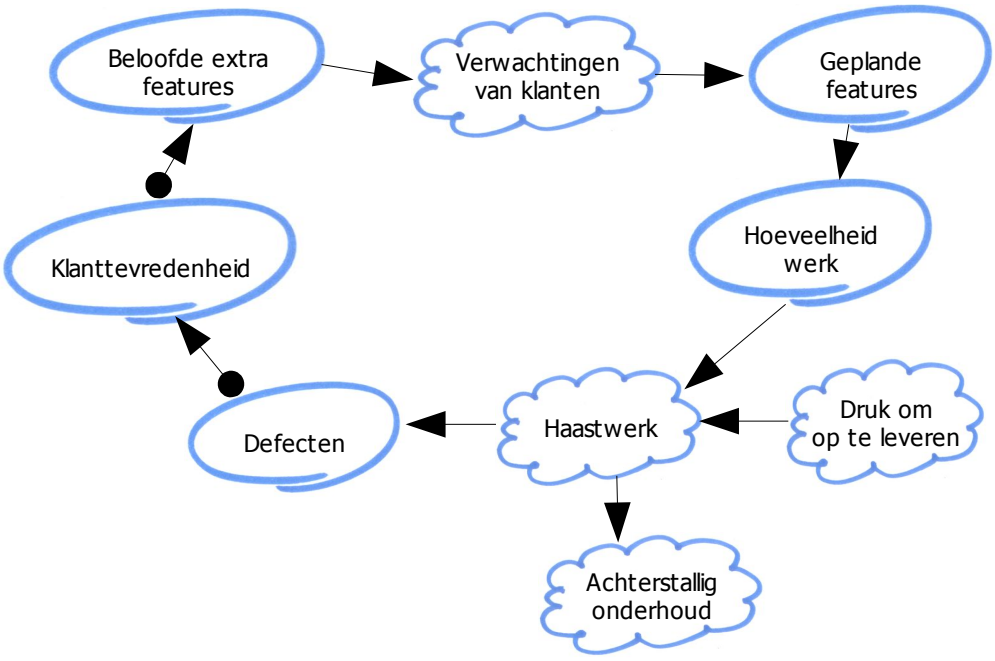
in te halen. De vorige release hebben we 21 punten gehaald. Deze keer hebben we er 6 punten bijgedaan. Die moeten we de volgende keer inhalen, dus we moeten 21 min 6 is 15 punten beloven om het duurzaam te houden. Waarschijnlijk kunnen we iets meer doen, maar laten we verstandig zijn en duurzaam ontwikkelen"

Klaas geeft toe, maar met tegenzin.

De volgende release

Het werk start traag. Het team is nog bezig bij te komen en is de eerste week weinig productief. Ze proberen wat van het achterstallig onderhoud weg te werken, maar dat lukt maar met mate. Ze zijn te moe om hun hoofd goed helder te krijgen. Vanaf de 2e week gaan ze, mede onder druk van Klaas, aan de slag met de stories voor deze release. Langzaamaan komen ze op snelheid.

Dan beginnen de verschillende klanten echter defecten te rapporteren. Het afgelopen jaar hadden ze 2 tot 3 defecten per release, nu kwamen er binnen een week opeens 4 binnen. Ze kregen ook een boze e-mail van Ronald die een vervelende bug had gevonden en zijn teleurstelling daarover niet onder stoelen of banken stak. Meteen ging het team aan de slag met de defecten – snel oplossen, anders raken ze Ronald meteen weer als klant kwijt



Een grote hoeveelheid werk in combinatie met de druk die Klaas op het team uitoefent zorgen ervoor dat ontwikkelaars eerder geneigd zijn om voor quick & dirty oplossingen te gaan, met het idee dat dat later wel in te halen is. Dit zorgt enerzijds voor steeds meer achterstallig onderhoud, anderzijds worden meer defecten geïntroduceerd door deze manier van werken. Meer defecten heeft meestal een lagere klanttevredenheid tot gevolg.

Tegen het eind van de release lopen ze flink achter. Ze proberen nog zoveel mogelijk op te leveren, maar ze leveren uiteindelijk maar de helft op van wat

ze gepland hadden: 11 storypunten. De verschillende klanten zijn lichtelijk teleurgesteld.

Volgende keer beter

Ronald begint openlijk zijn twijfels te uiten over het systeem. Klaas zoekt hem snel op en probeert het te sussen. "Het zat ons deze keer gewoon wat tegen, de ontwikkelaars waren wat grieperig aan het begin van de periode, daarna schoot het gewoon niet zo op; ik heb ze hierover stevig toegesproken, neem maar van mij aan dat ze de komende tijd beter hun best doen. Wij zullen zorgen dat je de volgende release ook feature 8RTv91x krijgt, met de HH05 extensie."

Ronald besluit ze nog een kans te geven. Klaas maakt een rondje langs de andere klanten en benadrukt dat de defecten en het niet halen van de planning slechts een incident was. De volgende keer gaat alles weer zoals het hoort.

Klaas drukt het team op het hart dat 8RTv91x af moet de komende release, falen is geen optie. Hij plant een reeks stories in die allemaal "essentieel" zijn voor de komende release. De teamleden zien dat dit veel meer is dan de velocity van de afgelopen release (26 punten); het is zelfs meer dan hun velocity van voor de problemen. De teamleden stemmen gelaten in: ze weten dat het niet gaat lukken, maar ze hebben geen keus denken ze.

De weg omhoog?

Het team merkt dat ze met de nieuwe stories langer bezig zijn. De quick & dirty oplossingen die ze voor de vorige twee releases hebben gebruikt zitten in de weg. Het kost ze meer tijd om hun code te begrijpen, om extra unittests toe te voegen en om de oorzaak van fouten te vinden. Ook blijven er nieuwe defecten binnenkomen, de meeste in de features die de afgelopen twee releases zijn opgeleverd. Na de preek van Klaas richten ze vooral op de 8RTv91x feature. Die krijgen ze af, andere stories blijven liggen. Ze realiseren deze release slechts 13 storypunten aan werk.

Ronald is deels tevreden deze keer: "Ik ben blij dat jullie 8RTv91x af hebben, maar ik had verwacht dat de x80y8 ook al klaar zou zijn, daar had Klaas het toen over." De andere klanten beginnen zich te roeren. "Deze keer hebben jullie alweer veel minder opgeleverd. Het lijkt wel alsof jullie alleen nog maar bugs opleveren!" zucht Fred gefrustreerd.

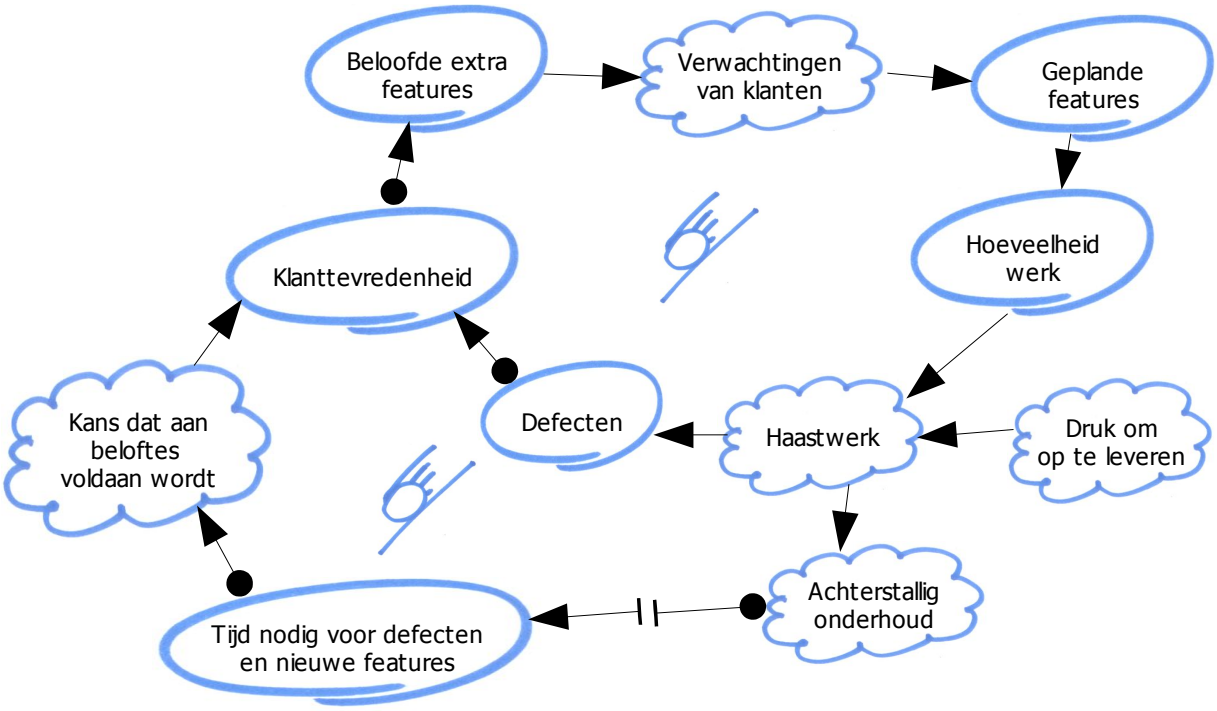
"x80y8 plannen we nu in!" belooft Klaas aan Ronald. Tegen de andere klanten zegt hij: "Ik ga een goed gesprek aan met het team, ik ben het helemaal met jullie eens, dit kan niet zo. De volgende release gaat het goed komen, dan leveren we ook Xnrg-4.5.4 op, waar jullie alle drie om zitten te springen!".

Klaas roept het team bijeen in een vergader zaal: "We moeten nu even hard werken om het vertrouwen van onze klanten weer terug te winnen. Ik weet dat jullie het kunnen, stel me niet teleur. Laat dat refactoring voorlopig maar even zitten, daar is nu geen tijd voor. Ik krijg het gevoel dat testen ook niet bijdraagt aan de productiviteit, als iedereen nu gewoon stories gaat bouwen, dan komt alles weer goed.". Hij gaat verder: "Als dat duidelijk is, dan is dit de planning voor de komende release, inclusief x80y8 en Xnrg-4.5.4. Alles bij elkaar 24 storypunten, dat moet geen probleem zijn want jullie hebben wel eens 29 gehaald. Volgens mij zijn jullie punten ook aan inflatie onderhevig, dus dan kan er best wat bij. Zo, genoeg vergaderd, ga maar weer snel programmeren."

Meer defecten

Ook deze release weer hetzelfde verhaal: stories realiseren kost meer en meer tijd vanwege achterstallig onderhoud; er blijven defecten binnenkomen, ook steeds vaker veroorzaakt door het oplossen van eerdere defecten. Het team begint zijn motivatie kwijt te raken. Ze proberen nog gauw wat stories af te raffelen, om verwijten van Klaas te voorkomen. Aan het eind van de release blijkt de velocity 11 te zijn, of 10, want van 1 story is er nog discussie of deze nu af is of niet.

Als Ronald zijn x80y8 uitprobeert, waar hij zo lang naar uitgekeken heeft, blijkt deze maar half te werken. En dan ook nog de minst belangrijke helft! Wat hem betreft is de maat vol, dit is het einde van VWS voor hem. Hij laat het niet na om deze beslissing luid en duidelijk aan te kondigen.



Na verloop van tijd heeft achterstallig onderhoud merkbaar invloed op de tijd die het oplossen van een defect of het bouwen van een nieuwe feature kost. De kans dat beloftes binnen de geschatte tijd worden gehaald, neemt hierdoor af. Minder vaak opleveren wat je belooft zorgt voor een dalende klanttevredenheid. Er zitten twee zelfversterkende lussen in dit systeem: extra features beloven leidt er indirect toe dat dat klanten nog minder tevreden worden. Het systeem is niet stabiel: klanten zullen weglopen en ontwikkelaars vertrekken.

Bij de koffieautomaat komt Vera Joop tegen. Ze ziet er vermoeid uit. "Het gaat niet best met VWS volgens mij," zegt Joop. "Nee" zegt Vera, "ik ben niet blij met de gang van zaken, sorry."

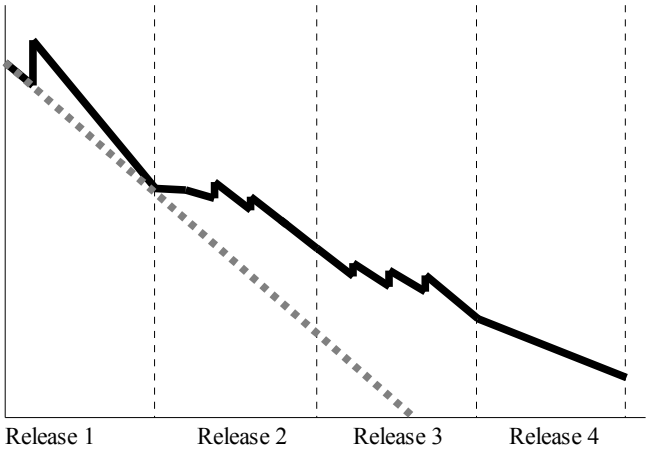
"Maakt niet uit, we zullen er niet lang meer last van hebben. We zijn aan het rondkijken naar een vervangend pakket en hebben nu twee geschikte kandidaten. Ik vind het sneu voor jullie, de samenwerking met de ontwikkelaars heb ik altijd positief gevonden."

"Nou, ik ben er ook bijna vanaf," zegt Vera, "ik heb gisteren een gesprek gehad bij QXD. Het klikte, ik ga er volgende maand beginnen."

"Gefeliciteerd! Jammer van VWS, maar ik ben in elk geval blij voor jou!"

Gedoemd te falen?

Als we het werk dat opgeleverd wordt uitzetten tegen de tijd, krijgen we onderstaande *burndown grafiek*. De zwarte lijn geeft de hoeveelheid resterend werk aan, de helling van deze lijn is een indicatie voor de ontwikkelsnelheid. De stippellijn geeft het verwachte verloop weer op basis van de beginsnelheid. Als de zwarte lijn afwijkt van de stippellijn is dat aanleiding voor verder onderzoek en mogelijk interventies.



In dit geval zien we dat er na release 1 minder resultaat wordt geleverd dan verwacht. Hoe komt dat? Wat kunnen we doen om binnen afzienbare termijn tot een oplevering te komen?

Onze oorzaak-gevolg diagrammen geven inzicht in de achterliggende dynamiek. Wat kunnen we hier nu mee? Of specifieker, wat zegt ons dit over mogelijke ingrepen? We kunnen bijvoorbeeld meer of minder features beloven, de hoeveelheid gepland werk variëren of meer of minder druk op het team uitoefenen. In het model betekent dit dat we de waarde van variabelen veranderen.

Als we alleen waarden van variabelen veranderen, maar de lussen in stand houden, blijft het systeem instabiel. We zien in onze praktijk dat een dergelijk systeem een behoorlijke tijd kan voortbestaan:

- ✗ klanten hebben geen andere keuze (dat denken ze tenminste) en blijven dan toch maar
- ✗ klanten durven niet goed voor zichzelf op te komen en laten het langere tijd over zich heen komen
- ✗ het product is relatief onbelangrijk voor de klant; de impact van de problemen is derhalve vrij klein voor de klant
- ✗ ondanks alles proberen de ontwikkelaars toch het beste ervan te maken; het zou misschien verstandig zijn om vroegtijdig dingen te laten escaleren, maar dat gaat tegen je gevoel van professionaliteit en vakmanschap in

Maar vroeg of laat gaat het mis en stort het systeem ineen: klanten lopen weg, mensen raken opgebrand, ontwikkelaars vertrekken.

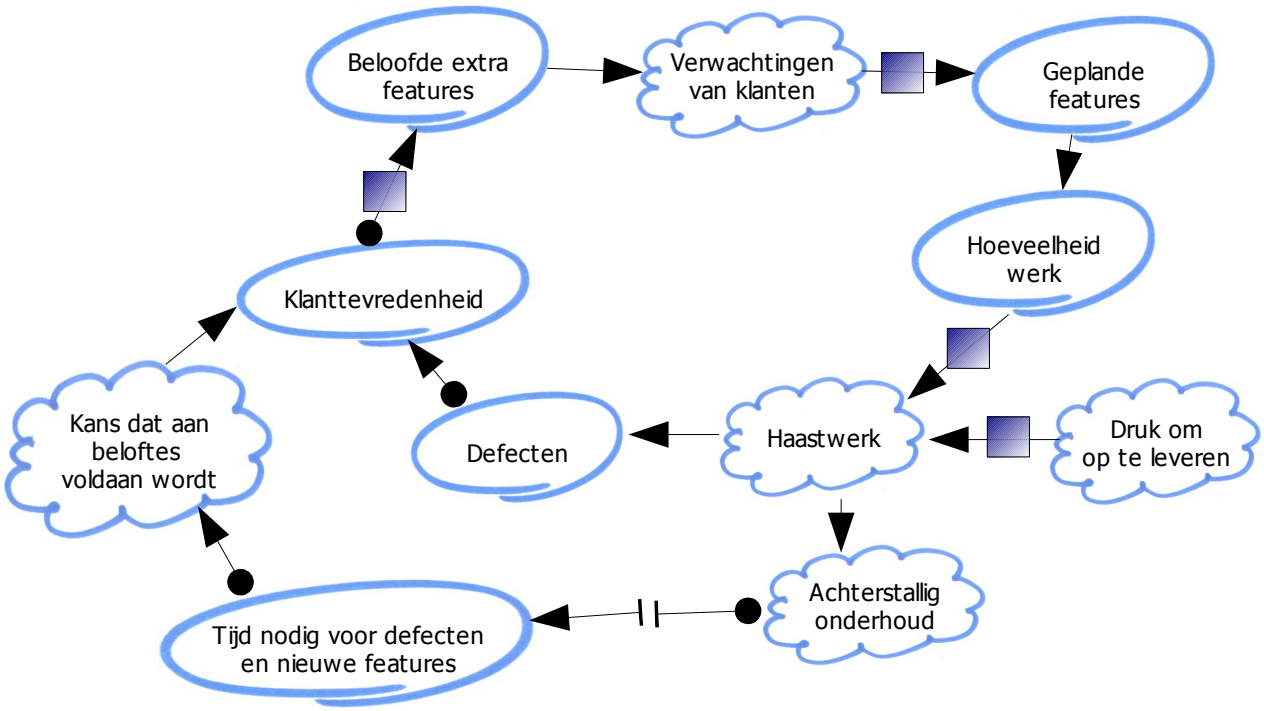
Het kan ook anders

Er zijn interventiemogelijkheden met meer effect. De relaties die we tussen de verschillende variabelen gevonden hebben zijn niet allemaal "natuurwetten" of in steen gehouwen: achter sommige zit een impliciete of expliciete keus (een *managementbeslissing*). Bijvoorbeeld de relatie tussen klanttevredenheid en beloofde extra features – we kunnen hier beslissen of we extra features beloven of niet als de klanttevredenheid daalt.

Er zijn nog meer punten in het model waar we een keuze hebben:

- ✔ de hoeveelheid druk die Klaas uitoefent op het systeem; het is niet afdoende om alleen op dit punt te interveniëren, de vicieuze cirkel blijft namelijk in stand
- ✔ de hoeveelheid extra features die je inplant om aan de verwachtingen van de klanten te voldoen
- ✔ de mate waarin de ontwikkelaars quick & dirty werken, als de werkdruk toeneemt

Deze managementbeslissingen hebben we in onderstaand oorzaak-gevolg diagram aangeduid met blokjes.



Door je mentale modellen en aannames op deze manier expliciet te maken en te bespreken, zie je welke keuzemogelijkheden je hebt en welke keuzes de dynamiek van het geheel structureel veranderen.

Het is zaak om het team niet te overbelasten. Op het moment dat de vicieuze cirkel al in werking is getreden kan dit lastig zijn: het team moet pas op de plaats maken, de verwachtingen naar beneden bijstellen. Je weet dat je een of meer klanten teleur gaat stellen. Alleen is het beter om dat bewust te doen, in plaats van het te laten gebeuren. Je zal een keer door de zure appel heen moeten bijten, daarna kun je zorgen dat de verwachtingen van de klanten realistisch blijven. Misschien zal het je een klant kosten, maar het alternatief is nog minder aantrekkelijk.

In onze ervaring is het ook uitkijken met weinig beloven en vervolgens verwachtingen ver overtreffen: het risico is dat klanten gaan verwachten dat je elke keer veel meer oplevert dan beloofd...

Conclusies

Het "Belofte maakt schuld"-patroon, waarbij men meer belooft om huidige problemen op te lossen en aannemen dat je het later wel inhaalt, werkt in het algemeen niet. Stukjes afsnijden lijkt daarbij aantrekkelijk, maar dat is het niet. Het probleem is dat de effecten niet direct zichtbaar zijn. Oorzaak en gevolg zijn indirect, liggen uiteen in de tijd en beïnvloeden elkaar wederzijds.

De combinatie van teveel beloven en stukjes afsnijden zorgt voor een vicieuze cirkel. Het team raakt in een destructieve spiraal.

We hebben deze spiraal in een aantal verschillende omgevingen gezien. Als deze optreedt, ligt het aan het systeem als geheel en niet aan individuele personen. Het zoeken van de persoon die 'schuld' is aan de problemen heeft dus geen zin en versterkt de vicieuze cirkel alleen maar.

Manipulatie, in dit geval door Klaas, leidt er toe dat mensen het moeilijk vinden nee te zeggen of zich er niet eens van bewust zijn dat nee zeggen een optie is. Nee zeggen moet voor iedere betrokkene altijd een optie zijn.

Goed leren observeren als team en het gebruik van oorzaak-gevolg diagrammen helpen om deze indirecte effecten en vicieuze cirkels zichtbaar en oplosbaar te maken.

Literatuur

Gerald M. Weinberg, *Quality Software Management*, volumes 1-4 (*Systems Thinking, First Order Measurement, Congruent Action, Anticipating Change*)

> *Diepgaande toepassing van onder meer systeemdenken op allerlei problematiek binnen softwareorganisaties.*

Peter M. Senge, *The Fifth Discipline: The Art & Practice of the Learning Organization*, 1994

> *Senge past systeemdenken toe op lerende organisaties. Hij behandelt onder meer effectdiagrammen en een aantal* archetypen *- terugkerende systemische patronen.*

Bill Bryan, Michael Goodman, Jaap Schaveling, *Systeemdenken – Ontdekken van organisatiepatronen*, 2006

> *Nederlandstalig boek over systeemdenken toegepast binnen de context van organisaties en management.*

Donella H. Meadows, *Places to intervene in a system*,in: Whole Earth Magazine Winter 1997 -

www.developerdotstar.com/mag/articles/places_intervene_system.html

> *Essay met een overzicht van verschillende interventiemogelijkheden in een systeem.*

systemsthinking.net

> *Portal met wiki en blog-aggregator, met teksten van verschillende systeemdenkers binnen de IT.*

Auteurs

Marc Evers (Piecemeal Growth)

Marc Evers werkt als zelfstandig coach, trainer en adviseur op het gebied van (agile) softwareontwikkeling en softwareontwikkelprocessen. Hij traint en begeleidt mensen op het vlak van technische vaardigheden, agile processen, projectmanagementvaardigheden en persoonlijke vaardigheden. Op deze manier helpt hij IT-organisaties om lerende organisaties te worden met focus op continue reflectie en verbetering. Marc richt zich op agile en lean softwareontwikkeling, systeemdenken, retrospectives en open space processen.

Marc is binnen Nederland een van de pioniers op het gebied van agile en Extreme Programming. Hij presenteert regelmatig bij verschillende internationale congressen en heeft de Agile Open en XP Day Benelux conferenties opgericht.

Telefoon: 06 44 55 000 3

Website: www.piecemealgrowth.nl

E-mail: marc@piecemealgrowth.nl

Willem van den Ende (Living Software BV)

Willem van den Ende is een Nederlandse eXtreme Programming pionier. Sinds 1999 begeleidt hij als trainer, coach, ontwikkelaar en facilitator organisaties bij het effectiever ontwikkelen van software. Hij is een oprichter van de xp-nl gebruikersgroep, systemsthinking.net en de conferenties XP Days Benelux en Agile Open. Verder is hij een gewaardeerd workshop facilitator op conferenties in binnen- en buitenland, zoals XP(Day), Software Practice Advancement en Agile200*. Willem is bestuurslid van de Agile Alliance, de internationale vereniging ter bevordering van Agile Software Development.

Willems scherpe blik, zijn enorme kennis en ervaring stellen hem in staat zich zeer flexibel en improviserend in workshops op te stellen. Hij heeft het vermogen mensen anders te laten kijken.

Telefoon: 06 413 06 965

Website: www.livingsoftware.nl

E-mail: willem@livingsoftware.nl

Bent u benieuwd wat systeemdenken (en -doen!) voor uw organisatie kan betekenen? Wij helpen u graag op weg, bijvoorbeeld met advies, een workshop of begeleiding. Neem gerust vrijblijvend contact met ons op.