Agile/Scrum workshop June 4th 2012

Page to put suggestions and have preliminary discussions for the Agile/Scrum workshop on June 4th 2012.

Intro

This email was sent on May 28th:

```
Dag lui,

aangezien hier wat vragen over kwamen heb ik bijgesloten het 1e voorstel dat
we besproken hebben met de coach/trainer.
Wat we maandag a.s. gaan doen valt onder de genoemde secties Scrum
workshop/Agile requirements workshop als genoemd
onder aanpak.

We hebben gekozen voor Agilix omdat zij met een programma kwamen dat
enerzijds flexibel is qua inhoud en anderszijds een
aantal momenten bevat van ''coaching on the job'', dwz het bij en met ons
meedoen in de meetings die we hebben, zodat het
meer een totaalprogramma is en geen ''cursus'' (dat is wat de andere
contacten ons aanboden) waarna we het zelf maar
moeten uitzoeken.

Als gezegd: dit is GEEN voorgekauwde theorieles, noch een ''SCRUM adoptie
cursus''; we gaan maandag beginnen met
inventariseren welke onderwerpen JULLIE vinden dat we moeten bespreken, en
Cesario (de coach) zal dat dan verder
oppakken. We hebben duidelijk aangegeven dat het doel van de workshop is een
manier van werken te bespreken en vinden die
goed past bij ons team. We vinden het wel belangrijk dat een aantal Agile
componenten daarin terugkomen. Na maandag zijn
er nog een aantal sessie waarbij Cesario bij onze meetings aanwezig zal zijn
en ''hands on'' discussie kan leiden en
verdere adviezen en tips kan geven om te zorgen dat de afgesproken manier
van werken ook in de praktijk kan worden
geborgd.

Als jullie vooraf alvast ideeen willen uitwisselen is dat natuurlijk welkom;
ik hoor het graag, en de rest van het team
waarschijnlijk ook.
```

```
---Door Nico Vermaas:---
Software ontwikkeling werkt normaal gesproken met een bepaalde lifecycle,
bepaalde fasen.
```

In vogelvlucht zijn dat:
Analyse Fase (levert gebruikers specificaties en een functioneel ontwerp) =>
Ontwerp Fase (levert een technisch ontwerp op),
Implementatie (levert functionaliteit & documentatie op).

Soms wordt het iets anders genoemd, of soms is de indeling net iets anders, maar het komt er altijd op neer
dat er verschillende perspectieven zijn, waar verschillende betrokkenen zijn, waarmee verschillend wordt
gecommuniceerd.
Ik heb het nu niet over de lifecycle van het hele project, maar over deelsystemen van de software.

Een groot probleem is dat dat bij ons allemaal dwars door elkaar loopt, dat er daarom geen duidelijke producten
worden opgeleverd. Daarom verloopt de communicatie chaotisch en ad-hoc, veranderen de specificaties gaandeweg en
verliezen we ontzetten veel tijd met het oplossen van problemen die hierdoor veroorzaakt worden.

Wat voorbeelden van waar dit bij ons fout gaat:
- gebruikers vertellen ontwikkelaars hoe ze dingen technisch geimplementeerd willen zien,
  maar hebben daarbij niet het overzicht om te weten wat voor problemen dat in andere delen van
  het systeem veroorzaken.
- ontwikkelaars leggen geen eenduidige specificaties van gebruikers duidelijk en centraal vast,
  waardoor gebruikers/ontwikkelaars/management regelmatig de vrijheid nemen om ze gaandeweg te veranderen.
  (veranderen moet wel mogelijk zijn, maar niet zonder dat dat duidelijk vastgelegd wordt,
  zodat we het overzicht houden).
- specificaties en technische ontwerp worden niet centraal besproken,
  zodat problemen in verschillende delen van het systeem vooraf geidentificeerd kunnen worden.
  Dit deden we voorheen beter (in de OMG), maar met Agile is dit slechter geworden.
- sommige ontwikkelaars praten op eigen houtje met gebruikers, beslissen dan zelfstandig over
  wijzigingen in de specificaties
- de interfaces tussen de deelsystemen liggen niet goed vast en veranderen regelmatig,
  waardoor de software aan de andere kant omvalt.
- Doordat het overzicht mist, is het voor de meeste ontwikkelaars ook niet duidelijk in welk deel van het
  systeem nieuwe functionaliteit dient te worden opgenomen.
- Gemaakte afspraken worden niet onthouden.
  En zelfs als ze wel worden vastgelegd, op de wiki, dan wordt er nog steeds

```
niet nageleefd
  (het veranderende componentenverhaal is daar een voorbeeld van).

  Advies:
  Het opleggen van een strakke ontwikkelmethodiek lijkt me niet handig,
omdat we ook flexibel moeten blijven,
  Maar ik zou wel graag zien dat we wat basale technieken en methodieken
zouden gaan gebruiken, die eigenlijk
  voor elke IT-er gesneden koek zouden moeten zijn.
Antwoord Arno:

Een deel van de problemen die je noemt is terecht, maar zou eigenlijk niet
mogen optreden als iedereen meer moeite doet te communiceren
en overleggen. Hier spelen een aantal problemen:
- Bijna iedereen heeft zijn eigen taakgebied (specialisme), en heeft sterk
de neiging
  - dat tegen elke inmenging van buitenaf af te schermen. Daardoor is er
wellicht te snel een reflex reactie: niet aankomen, dat is van mij!
  - alles in het werk te stellen om zijn ding werkend te krijgen, ook als
dat niet matcht met hoe anderen het eromheen designed hebben.
  - individueel op te treden; dat zijn we immers gewend in ons eigen
taakgebied.
- De OMG was een periodieke meeting en als zodanig niet altijd even zinvol;
professioneel gezien moet het team zelf organiseren dat
als het nodig is, de mensen die nodig zijn voor een probleem bij elkaar
komen (in principe niet periodiek, maar wanneer het juiste
moment daar is; het TEAM kan besluiten dat als men toch periodiek wil
overleggen, dat natuurlijk mag!); die verantwoordelijkheid en
bevoegdheid heeft ELK teamlid!
- Veel teamleden hebben een ingebouwde allergie tegen meetings; maar zolang
een meeting een duidelijk doel heeft en dient om 1
of meer teamleden verder te helpen bij het oplossen van een probleem, is het
altijd zinvol.
- Wegens gebrekkig teamoverleg is er veel communicatie ''op de wandelgang''
en weinig borging van gemaakte afspraken.

Wellicht is er te weinig focus op het goed oplossen van 1 probleem,  maar
proberen we altijd 3-4 problemen tegelijk op te lossen
en lukt dat vaak maar ten dele?

Wat kan je hier nog aan toevoegen?
```

Door Adriaan Renting

Ik wil hierbij ook even mijn mail van 14 oktober in herinnering brengen “Commentaar op de “nieuwe werkwijze” en het algemene software ontwikkeling proces.”, maar die ga ik hier niet herhalen. Daarnaast wil ik ook op de site over software anti-patterns wijzen: http://sourcemaking.com/antipatterns, hoewel dat zeker niet de enige site is waarop we wat zouden

kunnen leren.

Ik wil mijn reactie in twee delen splitsen: Een deel dat ook of vooral over het management gaat, en een deel voor het PWT.

Management

Mijn idee is dat er goed, met de verschillende management lagen, gekeken moet worden naar hoe besluitvorming binnen Astron plaats vind over software. Deze visie is geschreven vanuit mijn prespectief, waar ik normaal gesproken niet bij de besluitvorming aanwezig ben geweest en dus niet weet wat daadwerkelijk besproken is. Ik beoordeel alleen het effect dat ik vanuit mijn positie zie.

Tijd-Kwaliteit-Features afweging

Ik heb wat overlegd met vrienden van mij die bij andere software bedrijven werkzaam zijn, en één punt wat daaruit duidelijk naar voren kwam is dat er een driehoek bestaat met in de punten Tijd-Kwaliteit-Features. Het is niet mogelijk om in een developement cycle tegelijk aan alle drie deze punten te werken. Bij Astron heeft de focus heel lang gelegen op Kwaliteit/Features met het uit het oog verliezen van de component Tijd. De laatste maanden heeft de focus meer op Tijd/Features gelegen en is de Kwaliteit meer in de verdrukking gekomen. De ervaring van de genoemde vrienden is dat in de praktijk het goed werkt om een 4-6 maandse cyclus te hebben, waar de ene cyclus Tijd/Features de focus heeft, en de andere cyclus Tijd/Kwaliteit. Dit levert dus bij de meeste bedrijven een totale cyclus op voor een product van ongeveer een jaar.

Kennis van Software Management

Het valt mij op dat zowel bij management maar beperkt kennis is van Software Management. Tot op zekere hoogte geldt dit ook voor (een gedeelte van) de developers. Dit gaat tot het niveau van NWO en de subsidie verstrekkers. Ik stip enkele punten aan:

- De kosten van een herbruikbaar vs. een 'one of' systeem, denk bijv. aan het 'hergebruiken' van LOFAR voor APERTIF. (factor 3)
- Het leven op een papieren werkelijkheid, waarbij weinig met prototypes, demos en stubs gewerkt wordt.
- Sterk “Waterfall” model georiënteerd denken, met een focus op de specificatie.
- De kosten van maintenance van software.
- De kosten van het pas later in de ontwikkeling vinden van bugs en fouten, en daarmee het belang van vroege integratie.
- Het beoordelen en inschatten van de kosten en complexiteit van (scientific) automatisering.
- De tijd die het kost voordat iemand ingewerkt is en de tijd die het kost om iemand in te werken.
- Het efficiëntie verlies dat optreed als een developer meerdere taken toegedeeld krijgt.
- De kosten om van een prototype (en ik zie LOFAR 1.0 gedeeltelijk als een prototype!) te komen naar een functioneel robuust systeem. (factor 3)
- De kosten van het in huren van expertise vs. het in huis hebben ervan.
- De kosten van late en constante wijzigingen in het ontwerp.

- De kosten van flexibiliteit, vooral m.b.t. een “software telescoop”.

Ik snap dat hier ook afwegingen m.b.t. de opgelegde financieringsmodellen een rol spelen, waardoor de ideale keuze niet altijd gemaakt kan worden, maar ik heb het beeld dat er ook binnen die kaders veel niet optimale keuzes gemaakt worden wegens een gebrek aan inzicht in de uiteindelijke kosten.

Prioriteit stelling

Ik heb het gevoel dat door management gekeken moet worden naar het proces van prioriteit stelling.

- Ik meen een sterke neiging te zien om prioriteit te geven aan de meest volwassen onderdelen, waar al resultaten uit komen en dus feedback tot verbetering gegeven wordt, terwijl de minst volwassen onderdelen weinig aandacht krijgen.
- Het op elkaar afstemmen van prioriteiten op verwachtte doorlooptijd zodat niet iets dat af is, daarna nog maanden of jaren moet wachten op iets anders voordat het gebruikt kan worden. Hierbij ook expliciet de balans hardware-software in mee nemen.
- Een overkoepelende visie op in welke volgorde de verschillende functionaliteiten nodig zijn. Het komen van een “alles is prioriteit 1” naar een geordende lijst van taken op prioriteit. Hierin zijn met de invoering van Scrum wel enige stappen gezet maar alleen met een hele korte horizon.
- Science-Engineering interactie. Er wordt veel direct door de wetenschappers al in oplossingen gedacht, en die dan als specificaties naar de ontwikkelaars gecommuniceerd, zonder dat eerst over het op te lossen probleem overleg heeft plaats gevonden tussen science en engineering.
- Een focus om de meest arbeidsintensieve taken eerst te automatiseren.
- Het inramen van tijd en mankracht voor actieve bewaking van het software ontwikkelingsproces en de software architectuur.
- Het vroeg inzetten van die ontwikkelingen die het grootste risico hebben.

Clear management

Het LOFAR project heeft naar mijn mening veel te leiden gehad onder een gebrek aan een duidelijke chain-of-command en lijn van verantwoordelijkheid. Op veel lagen werden/worden groepen en teams als management aangegeven. Dit lijkt binnen Astron meer voor te komen. Ik zie ook een gebrek aan besluitvaardigheid binnen de organisatie, wat volgens mij hier sterk mee verbonden is. Ik mis een duidelijke verantwoordelijkheidsstruktuur, duidelijke, meetbare doelen en het afrekenen daarop.

Daarnaast leeft bij mij het beeld bij de selectie/aanstelling van management verbeteringen mogelijk zijn, maar die kan ik nog niet helemaal helder verwoorden.

Communicatie met developers

Er wordt weinig met de developers gecommuniceerd over de verwachtingen, tijdslijnen en wat met externe partijen afgesproken is. Mogelijk dat dit wel in wollige documenten te vinden is, als deze al beschikbaar zijn. Ik denk dat de waarde van het actief opzoeken van directe communicatie hier onderschat wordt, en dat er ook een verschil in mondigheid is tussen de ontwikkelaars en wetenschappers wat mogelijk te weinig onderkend wordt.

Management is voor een groot gedeelte de communicatie tussen de ontwikkelaars en de eind-gebruikers en de algemene buitenwereld. Ik heb het gevoel dat er vaak een groot verschil zit in het

beeld wat aan beide kanten leeft.

Lessons Learned

Mijn beeld is dat er weinig gekeken wordt naar eerdere software projecten om daar lessen uit te trekken. (TMS, Newstar, AIPS++ bijvoorbeeld, nu ook LOFAR).

Dit heeft ook te maken met mijn eerste PWT punt. Volgens mij is het met het huidige uren schrijf systeem (Primavera) en de manier waarop dat vanuit management aangestuurd wordt, vrijwel onmogelijk om een goed inzicht te hebben waar de knelpunten liggen in de besteding van manuren, in het ieder geval waar het software ontwikkeling betreft. Dit betekent dat men weinig kwantitatief inzicht heeft waar de grootste winst te behalen is en waar de grootste verliezen gemaakt worden. Ik denk dat hierdoor ook een sterke focus is op het binnenhalen van nieuwe projecten, en weinig reflectie op bestaande projecten.

Paviljoen West Team

Uren

Ik heb het idee dat bij een groot gedeelte van het team het aantal uren dat men “gewerkt” heeft niet overeen komt met het aantal uren dat men aanwezig was. Dit niet omdat mensen zitten de lanterfanten, maar omdat veel collega's uren die niet doorgebracht worden met het inkloppen van code niet meegeteld worden. Mijn suggestie is te kijken of we een uren schrijf systeem kunnen vinden, waarbij de uren dat men aanwezis verdeeld moeten worden over de gestelde taken (incl. evt overige uren zoals Colloqium e.d.) Ik denk dat dan een veel realistischer beeld gekregen wordt van waaraan de tijd besteed wordt en waar de initiële schattingen erg afwijken.

Communicatie met management

Ik denk dat er voor zover de kennis wel aanwezig is, er een duidelijk communicatie kanaal met de verschillende management lagen moet komen, zodat de Software Management gerelateerde punten al tijdens de besluitvorming meer meegenomen worden. Dit vergt ook een zekere bereidwilligheid van management om te luisteren naar niet altijd even communicatief vaardige developers. Hoe dit precies vorm te geven zou samen met management besproken moeten worden. Ik denk dat teveel nu “op de gang” blijft hangen, hoewel ik ook gemerkt heb dat management niet altijd de signalen oppikt die wel afgegeven worden.

Bus factor

Ik neem aan dat de “Bus factor” bij de meesten een bekend begrip is. Ook is bekend dat deze bij LOFAR erg laag ligt, waar vaak maar één persoon verantwoordelijk is voor een stuk software. Het voorstel waar Wouter ook al mee kwam, om voor alle software peer-review te gebruiken, zou hierin al een stuk kunnen helpen. Bij mijn eerste werkgeven was het gebruik dat een developer de code

schreef, en dan een andere deze globaal controlleerde en die de code check in deed. Dit kost initieel meer tijd, maar levert uiteindelijk betere code op, wat verderop in het development proces snel terug verdiend wordt.

Software Architectuur

Er worden vanuit het team weinig pogingen gedaan om discussies op architectuur niveau te voeren.

- Interface afspraken zijn gebrekkig en vaak ongedocumenteerd. Mijn voorstel is om hiervoor een taal-onafhankelijke methode af te spreken.
- Architectuur en design beslissingen worden vaak niet vast gelegd. Het 'waarom“ ontbreekt vaak, alleen 'hoe' is opgeschreven.
- Onderdelen die niet direct op het eigen 'eiland' nodig zijn worden snel buiten het verantwoordelijkheidsgebied geplaatst, waarbij de overkoepelende architectuur uit het oog verloren wordt.
- Een probleem wat zich voordoet wordt gauw op het eigen “eiland” opgelost, waarbij de overkoepelende architectuur niet in ogenschouw genomen wordt.
- Samengevat een sterke neiging om zich een klein stukje toe te eigenen, maar voor de rest de verantwoordelijkheid van zich af te schuiven naar andere developers of “het management”. Er is weinig verantwoordelijkheid voor het geheel.

Testen

Wat duidelijk bleek uit de integratie stappen die de laatste tijd gedaan zijn, is dat er stukken software zijn die eigenlijk geen “test modus” hebben. Ook is er in de architectuur, voor zover deze er al is, weinig rekening daarmee gehouden. Ik denk dat een 'test modus' een expliciet onderdeel zou moeten zijn van interface en architectuur afspraken. Hoe deze vorm te geven is denk ik een discussie die we zo snel mogelijk moeten voeren.

Conclusie

Mijn voorlopige conclusie is dat zowel bij management als bij het PWT meer kennis nodig is van Software Management, zodat men elkaars 'sparring partner' kan zijn bij het verbeteren daarvan. Daarnaast is meer transparantie nodig in verantwoording en verantwoordelijkheid en duidelijke lijnen voor de communicatie van doelen en prioriteiten en de terugkoppeling daarop.

Ik zou graag op een hoger niveau naar de processen binnen Astron willen kijken dan alleen de implementatie van Scrum.