Reverse Requirements Engineering Applied.

Matthias Oliveiro
22-12-2004
"1 jarige Master Software Engineering"
Afstudeerdocent: Jan van Eijck

Stagebegeleider: Daan v/d Berg
Opdrachtgever: Interswitch B.V.

Samenvatting

Requirements engineering bestaat nu al een paar jaar. Het is de kunst van het uitvinden van wat de klant precies wil hebben in een nieuwe applicatie. Reverse Requirements Engineering is bijna hetzelfde als gewoon requirements engineering, er is echter een groot verschil. Reverse Requirements Engineering werkt als reverse engineering en heeft dus een andere aanpak dan normaal requirements engineering In dit document zal ik uitleggen wat dit verschil is en waarom ik juist heb gekozen voor Reverse requirements engineering. Bij het document bijgevoegd is een Programme of Requirements voor het bedrijf Interswitch.

Interswitch B.V. is een callcenter, dat een legacysysteem als productiesysteem gebruikt. Om de requirements uit dit legacysysteem te halen moeten we eerst het systeem op hakken in kleine delen en deze stuk voor stuk analyseren. Ook de datastromen tussen de blokken moet geanalyseerd worden. Dit is nodig omdat we willen weten hoe de delen van het systeem zich gedragen in een keten.

Van deze kleine beetjes informatie over het systeem creëren we een Programme of Requirements. Met dit document is het mogelijk om het huidige systeem van Interswitch te vergelijken met hoe het systeem zou moeten zijn, of wel het ideale systeem. Uitgaande van dit document kunnen we Interswitch adviseren over hoe men nu verder moet met hun huidige systeem.

Inhoud

Inleiding

Wat is Reverse Requirements Engineering nu eigenlijk? Reverse Requirements Engineering, of ook wel RRE genoemd, is een manier om requirements op te stellen. Het grote verschil met normale requirements engineering is dat men bij normale requirements engineering uit gaat van een ideaal systeem. Bij RRE gaat men uit van een al bestaand systeem en stellen aan de hand van dit systeem het ideale systeem op.
Nu hebben bijna alle bedrijven een soort van software infrastructuur waar de bedrijven gebruik van maken om hun doelstellingen te halen. Alles in het leven veranderd van dag tot dag, zo ook in het bedrijfleven. Om bij te blijven met de concurrenten moeten bedrijven veranderen en met hen de software infrastructuur. Het is dus ook niet zo verwonderlijk dat bedrijven investeren in nieuwere, snellere en betere IT oplossingen. Een nieuw software systeem ontwikkelen kost veel geld, zeker wanneer het bedrijf afhankelijk is van dit systeem. Daarom is requirements engineering erg belangrijk het zorgt er voor dat het nieuwe systeem ook daadwerkelijk doet wat men wil dat het zou moeten doen. Reverse requirements Engineering heeft hetzelfde doel maar bereikt dit op een betere manier. In plaats van het wiel opnieuw uit te vinden begint Reverse Requirements Engineering met het oude systeem, haalt hier de requirements uit en maakt van het oude systeem het ideale systeem. Het mooie van deze aanpak is dat we instaat zijn om de oude situatie te vergelijken met de nieuwe. Je kan hier goed zien of de business drivers zijn veranderd of niet.

Om Reverse Requirements Engineering te illustreren maken we gebruik van een casestudy van het bedrijf Interswitch, dit bedrijf heeft mij gevraagd om eens een blik te werpen op hun systeem. Interswitch is een inbound callcenter, dit wil zeggen dat Interswitch niet mensen op belt, wat we meestal gewend zijn van callcenters, maar dat ze gebeld wordt. Effectief gezien neemt Interswitch de telefoonwaar voor andere bedrijven. Meer details over Interswitch staan op bladzijde vier van het bijgeleverde Programme of Requirements.

Probleem definitie

In dit hoofdstuk lichten we de doelstellingen en de probleem definitie toe. De bedoeling van dit document is te laten zien dat Reverse Requirements Engineering niet alleen een theorie is maar dat het werkt in de realiteit. We passen de Reverse Requirements Engineering theorie toe de situatie van het bedrijf Interswitch.

Het probleem met het software systeem van Interswitch is vrij simpel. Het software systeem is gebouwd in 1988. Nu is dit op zich niet een probleem zo lang het goed onder houden wordt over de jaren dan gaat het systeem redelijk goed met de tijd mee. Echter heeft het systeem de afgelopen 5 jaar geen onderhoud gezien. Dit komt door dat het bedrijf dat het systeem gebouwd heeft enkele jaren geleden failliet is gegaan. Met andere woorden het systeem is erg verouderd en er is niemand die exact weet hoe ze het systeem "up to date" moeten houden. Het is dan ook niet zo vreemd dat Interswitch een nieuw systeem wil. Echter weten ze niet wat het nieuwe systeem exact moet bevatten. Verder is er nog de vraag of ze een " off the self" oplossing kopen of een softwarehuis het systeem helemaal opnieuw voor hen laten ontwikkelen.

Gebruik makende van Reverse Requirements Engineering en kijkende naar het oude systeem maken we een template voor het nieuwe systeem. Dit gaat een heel ander soort template worden dan wanneer je forward requirements engineering gebruikt. *Voorspelling: Reverse Requirements Engineering zorgt voor een nauwkeuriger resultaat dan Forward Requirements Engineering.*

Al het bovenstaande beschouwende zal het resulteren in een Programme of Requirements van een ideaal systeem voor Interswitch. Uit gaande van mijn belevingen en het Programme of Requirements zal ik Interswitch adviseren over het systeem in de nabije toekomst.

In het kort bestaat mijnopdracht uit drie delen:

- Maak een Programme of Requirements voor het systeem van Interswitch.
- Beoordeel het systeem van Interswitch en schets een toekomstbeeld.
- Hou het systeem draaiende.

Plan van aanpak

Het is nu bekend waar we RRE op toe gaan passen. Er is echter nog een ding niet duidelijk.
Waarom gebruiken we hier RRE en niet FRE?
FRE is kan in deze situatie ook heel goed gebruikt worden. Ik twijfel er zelf niet aan dat wanneer men FRE gebruikt ook een degelijk systeem voor Interswitch in elkaar weten te zetten. RRE heeft zo zijn voordelen. Niet alleen denk ik dat RRE sneller en makkelijker toegepast kan worden maar kijkende naar wat Interswitch wil is het ook de meest logische optie. Interswitch wil niet een nieuw systeem, ze willen het oude systeem vernieuwen. Een veel voorkomende techniek binnen RRE is " de donut" techniek.

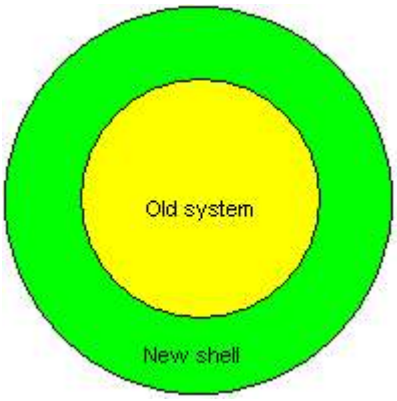

Figure 2 : The Donut effect

Zo als we kunnen zien in figuur 2 bestaat het figuur uit twee delen. Het binnenste gedeelte is het oude systeem; de schil er omheen is een laag van nieuwe code. Met deze laag of schil maakt men het systeem weer up to date. Systemen waar dit systeem mee samen werkt zien niet de oude kern het enige wat ze zien is een geheel. Deze techniek zie je vaak bij systemen die 24 uur per dag moeten draaien. Banken zijn hier een goed voorbeeld van, die systemen bestaan uit een heleboel laagjes.
Een ander voordeel van deze techniek is dat je doet aan 'code reuse' je hoeft niet het hele systeem opnieuw te maken je schrijft eigenlijk een gigantische wrapper voor het oude systeem. Dit is interessant voor Interswitch, want het betekent dat het mogelijk is om de kern van hun systeem te behouden. De enige voorwaarde hier aan is dat de kern van de donut stabiel moet zijn. Is dit niet het geval dan zal de schil gaan scheuren en kunnen we het wel vergeten.

Wat maakt mijn toenadering anders dan andere studies op het gebeid van reverse engineering en requirements engineering? In een publicatie van het DOD (department of defence, 1994 ACM) wordt reverse engineering gebruikt om business rules, business domein informatie, functionele requirements en de architectuur van het systeem te achter halen. Van deze informatie werden een hoop modellen gemaakt om zo het systeem beter te begrijpen. Voor Interswitch hoeft dit natuurlijk allemaal niet aangezien het een veen kleiner bedrijf is. Echter gebeurt er in essentie het zelfde. Het DOD probeert hun verouderd systeem, ook wel een legacy systeem genoemd, te vernieuwen door er een nieuwe schil om heen te bouwen dit kan ook bij Interswitch gebeuren. Een groot verschil is echter dat het DOD niet helemaal opnieuw

kan beginnen er zijn te veel kritieke systemen bij het DOD. Bij Interswitch hebben we deze keus wel om een nieuw systeem te gaan maken.

Breng het systeem in kaart
Eerst moeten we weten wat het systeem allemaal bevat. Dit doen we door niet alleen alle stukken software en hardware te archiveren maar ook alle manual en ander documentatie van het systeem. Dit is nodig want we willen het systeem in kaar brengen. We willen weten wat het systeem nu eigenlijk allemaal omvat. Met deze "kaart" heb je een goed idee wat voor architectuur er is gebruikt toen het systeem werd gebouwd.

Als je het systeem in kaart brengt moet je oppassen dat je niet een kaart maakt die te gedetailleerd is. Een te gedetailleerde kaart kost veel meer tijd en er bestaat de kans dat de kaart biased is.

Markeer de interessante punten
Na dat we het systeem in kaart hebben gebracht moeten we de interessante punten, of ook wel "keypoint" genoemd, markeren op de kaart. Sommige applicaties maken gebruik van delen van het systeem maar zijn zelf geen onderdeel van het systeem zelf. Denk hier aan een loon administratie systeem. Dit systeem maakt gebruik van bijvoorbeeld het werkrooster van het algemene systeem maar is zelf geen onderdeel van het systeem. Wat we willen ontdekken is welke onderdelen essentieel zijn voor het systeem. Hier mee vinden we de "core requirements van het systeem".

Bij het aanduiden van 'ke ypoints" moet men in acht nemen niet te veel 'k eypoints" te identificeren. Men moet bewust blijven van wat nu echt nodig is voor het systeem. Wensen van de klant om extra functionaliteit toe te voegen is nu nog niet belangrijk. Je wil namelijk de kern van het systeem vinden, later hang je hier alles aan op.

Maak use case scenario's
Nu we de kern van het systeem hebben gaan we deze toetsen. Dit kan je heel goed doen door use case scenario's te maken. Van uit deze scenario's kan je zien of er essentiële requirements missen en of er overbodige requirements tussen zitten. De laatste komt voor wanneer de business drivers van een bedrijf veranderen. Bijvoorbeeld een bedrijf gaat een ander product aanbieden waardoor het oude product overbodig wordt. Dit betekend dat er een verandering van de requirements heeft plaats gevonden.

Let er op dat alle stakeholders worden geïdentificeerd. Mis je een stakeholder dan heb je een goede kans dat delen van het systeem voor overbodig worden gezien en dus geschrapt worden.

Het uitspecificeren van het systeem
We hebben de essentiële requirements van de kern van het systeem. Nu moeten we deze gaan uitwerken. Het lezen van de manuals van het systeem zorgt er voor dat een hoop tot nu toe onbelangrijke details boven water komen. We kijken ook naar de hardware en lijden hier constrains van het systeem af. Dit zijn niet alle constrains er komen ook constrains later te voorschijn.

Om ook verborgen requirements te vinden moeten we de code in duiken. Dit kan je doen doormiddel van tools, maar ook het lezen van de code kan je inzicht geven waarom iets op deze manier geprogrammeerd is.

Een voorbeeld van een verborgen requirement is het volgende. Uit de code kon worden opgemaakt dat een faxservice om de vijf minuten werd herstart. Na navraag hebben gedaan bleek dit te kloppen omdat berichten voor klanten binnen 5 minuten moeten verstuurd worden. (zie pagina 5 van het bijgevoegde document)

Hardware constrains zijn tegenwoordig niet het grootste probleem. Hardware is goedkoop geworden en de meeste bedrijven investeren hier veel in. Let wel op dat dit goed wordt overlegt met de stakeholder het kan namelijk heel goed zijn dat er specifieke hardware word gebruikt met een reden.

Wordt de stakeholder

Nu we de requirements deels hebben gevonden moeten we deze verder uitspecificeren. Een goede manier om dit te doen is om de stakeholder te worden. Plaats je zelf niet alleen in zijn schoenen maar probeer ook de dagelijkse taken die deze stakeholder doet met het systeem, ook te doen en te begrijpen. Op deze manier ben je instaat om de use case scenario's, en de hier aanverbonden requirements, die je hebt gevonden te toetsen en nieuwe requirements te vinden.

Let wel op met wat je aan het doen bent. Je bent namelijk wel bezig in de productie omgeving van het bedrijf. Als dit goed wordt uitgevoerd zorgt dit ook voor een hoop goodwill onder het personeel.

De shitlist

In elk systeem zitten fouten. Hoe klein ze ook zijn, deze fouten beïnvloeden de stabiliteit van het programma. Het komt vaak voor dat deze fouten het resultaat zijn van ontbrekende requirements of verkeerde requirements. Om deze requirements boven tafel te krijgen moeten we een "shitlist" maken. Een "shitlist" is een lijst waar alle problemen opstaan van het systeem. Met deze fouten kan je requirements controleren, verkeerde requirements vallen snel door de mand aangezien je ziet wat voor fouten ze op leveren.

De shitlist een goeie methode om verborgen requirements vinden. Deze requirements ontbraken of waren niet correct toegepast in het originele systeem. Let wel op dat je de fouten die door verkeerde requirements er uit filtert en niet te lang bezig bent met fouten die hier niets mee te maken hebben.

Het ideale systeem.

Deze stap staat eigenlijk parallel aan alle andere stappen en moet eigenlijk bij al deze stappen worden uit gevoerd. Bij elke requirement die je vind moeten we gaan na denken of deze requirement nog wel klopt. Hier mee bedoel ik of de requirement niet verouderd is. Je doet dit door met alle requirements een ideaal systeem op te stellen. Dit systeem lijkt waarschijnlijk heel veel op het oude systeem maar er is nu naar gekeken met een oog op wat voor moderne technieken er nu beschikbaar zijn.

Het vinden van de requirements

Na het maken van het plan van aanpak kunnen we gaan beginnen met het opstellen van de requirements.

Breng het systeem in kaart
Allereerst moet het systeem van Interswich in kaart gebracht worden. Hier voor moeten we de hardware, software en de manuals oppervlakkig door kijken. Het probleem was echter dat er geen manuals meer ware of nooit zijn geweest. Dit betekende dat we blind aan de slag moesten met het systeem. De bedoeling was geweest om in de manuals te vinden hoe het systeem ongeveer werkte en het dan op te hakken in interessante delen. Nu moeten we dit uit vinden gebruikmakende van de software en hardware alleen. Dit zorgde voor enige vertraging. De resultaten van deze stap kunt u vinden in het hoofdstuk **"Cont ext oude systeem"** in het bijgevoegde Programme of Requirements.

Markeer de interessante punten
We hebben het systeem nu in kaart gebracht en we weten hoe het systeem werkt in vogel vlucht. Nu is het de bedoeling dat we de interessante punten, ook wel "keypoints" genoemd, in het systeem benoemen. De requirements die we later gaan vinden hangen we onder een van deze keypoints. Op deze manier krijg je niet een hele lange lijst met requirements maar een aantal kleinere meer geordende lijsten. Zie voor de resultaten het hoofdstuk **"Cont ext oude systeem"** in het bijgevoegde Programme of requirements.

Maak use case scenario's
We weten nu wat het systeem doet we hebben echter nog geen idee wat de gebruikers met het systeem doen. Met behulp van Use cases kunnen we uitvinden wat de stakeholders nu eigenlijk met dit systeem doen. Met deze usecases kwamen er een aantal fouten in de huidige requirements naar boven. Dit kwam omdat er een aannamen gedaan was die niet klopte. We hadden gesteld dat de telefonisten, een van de stakeholders, redelijke veel konden met het systeem. Je moet hier denken aan het verwijderen en aanpassen van de boodschappen. Later bleek dit niet het geval te zijn omdat de telefonisten eigenlijk helemaal niets mogen. Dit is toen het systeem werd opgezet werd voor gekozen van uit het management. De telefonisten kunnen effectief alleen maar invoeren van berichten.

Het uitspecificeren van het systeem
We hebben nu het systeem van Interswitch in kaart gebracht en we weten wat de stakholders taken zijn. Eigenlijk was het de bedoeling om tijdens deze stap de manuals van het systeem meer in details in te lezen. Nu deze er niet zijn moeten we dit doen door dieper in de software te gaan graven om zo de requirements te achter halen. De requirements die hier uit kwamen werden onderverdeeld onder de key points die waren gevonden.
Bij het kijken naar de hardware constrains viel me al snel op dat alle hardware sterk verouderd was. Men maakt gebruik van 386 machines zonder hardeschijf maar met een opstart floppy. Van af het netwerk wordt het systeem opgestart. Na een kort gesprek met de directeur werd er toegezegd dat er nieuwe hardware wordt aangeschaft. Wat dit ging worden was nog niet bekent maar mijn advies hier in werd op prijs gesteld.

Wordt de stakeholder
Om te controleren of de requirements die niet alleen gevonden waren bij het maken van de use cases maar ook later bij het verder uitspecificeren van het systeem besloot ik om een stakeholder te worden. Ik heb de twee dagen die daar op volgden gewerkt bij Interswitch als een telefonist. Ik kwam er hier achter dat een telefonist eigenlijk een hele hoop acties moest doen voor dat een bericht daad werkelijk werd verzonden naar de klant. Dit zorgde voor een grote toevoeging van requirements aan het interface gedeelte van het systeem. De meeste requirments waren redelijk voor de hand liggend en dat was waarschijnlijk de reden waarom ik ze niet direct gevonden heb.

The shitlist
De shit list van Interswitch kan gevonden worden in de bijlagen. Iets wat op viel was dat wanneer er een onderdeel van het systeem begaf de symptomen te zien waren op de faxmachines. De faxmachines zorgen voor de terug koppeling naar de klanten maar had schijnbaar ook nog de functie van alarmcentrale voor de mensen van Interswitch.

Het ideale systeem
Tijdens het vinden van de requirements hebben we er ook steeds naar gekeken of het systeem met deze requirements goed zou functioneren. Bij een hoop requirements was dit niet het geval. Een voorbeeld hier van is de Fax/alarmcentrale in de vorige alinea. Een alarm centrale zorgde nu niet echt voor een stabiel werkklimaat. Aangezien als er een alarm af ging men in paniek raakte en zomaar dingen ging doen. Dit is niet iets wat je wil als systeembeheerder. Dus deze requirement werd geschrapt. Van uit de Context van het oude systeem hebben we een ideaal systeem gecreëerd. Zie voor het ideale systeem het Programme of Requirements hoofdstuk: **Context ideale systeem**

De Resultaten

Het Reverse Requirements Engineering project resulteerde in een Programme of Requirements en een evaluatie van het huidige systeem. Het Programme of Requirements kan gevonden worden in de bijlagen, Interswitch gebruikt het om vervangende systemen te vergelijken en te toetsen.

Tevens heb ik Interswitch geadviseerd om het huidige systeem te laten voor wat het is en hier geen ontwikkelingen meer aan te doen. Dit zou namelijk in verhouding tot een nieuw systeem veel te veel gaan kosten. Er zijn een aantal goede systemen op de markt die het huidige systeem goed kunnen vervangen. Deze systemen zullen nooit helemaal aan de requirements van het ideale systeem voldoen maar komen wel een heel eind.

Tijdens het project droeg ik ook de verantwoordlijkheid om het huidige systeem draaiende te houden. Nu aan het einde van het project draait het nog steeds maar het scheelde niet veel of het had niet gewerkt. Interswitch is erg afhankelijk van dit systeem en dat maakt het erg riskant om er onderzoek naar te doen terwijl het draait. Het systeem is een aantal keer erg hard gecrasht maar ik heb het weer aan de praat gekregen.

Heeft Reverse Requirements Engineering gewerkt? Ja, nu aan het einde van het project kunnen we concluderen dat Reverse Requirements Engineering heeft gewerkt. We hebben de requirements gevonden van het originele systeem, deze hebben we up to date gemaakt. dit resulteerde in het Programme of Requirements. Tevens heeft het de mensen van Interswitch in staat gesteld om te kijken hoe hun systeem nu eigenlijk werkte. Dit was niet mogelijk geweest wanneer we conventionele requirements hadden toe gepast. Hadden we dit wel gedaan dan zouden we een heel ander systeem hebben gekregen.

In het begin van dit document heb ik een voorspelling gedaan deze kan ik nu beantwoorden. De voorspelling was:

Reverse Requirements Engineering zorgt voor een nauwkeuriger resultaat dan Forward Requirements Engineering.

Deze voorspelling klopt inderdaad. Onlangs heeft een software bedrijf dat gespecialiseerd is in Callcenters een pakket aangeboden. Dit pakket voldeed niet aan de requirements van het ideale systeem. Nu komt dit niet door dat het Programme of Requirements voor Interswitch was geschreven en het programma niet. Want er ontbraken een aantal elementaire onderdelen zoals de mogelijkheid om telefoongesprekken te loggen. Reverse Requirements Engineering is toe te passen op elk systeem dat je zou willen.

Evaluatie

In dit gedeelte van het document zullen we het project evalueren. Evaluatie is nodig omdat de kwaliteit van toekomstige projecten gelijk of beter moet zijn dan dit project. Fouten die gemaakt zijn bij dit project hoeven niet nog eens gemaakt te worden.

Accomplishments
Wat hebben we bereikt met dit project? We hebben bewezen dat Reverse Requirements Engineering werkt. Het is een methode die toegepast kan worden op systemen die vervangen moeten worden. Reverse Requirements Engineering scheelt tijd en fouten en het leid je naar de antwoorden, in dit geval zijn dit de requirements.

Tevens heeft Interswitch een Programme of Requirements van een ideaal systeem. Dit kan gebruikt worden om te kijken of een nieuw systeem aan de eisen voldoet aan de eisen die zij stellen aan het system.

Het systeem werkt nog steeds, dit is eigenlijk een beetje een dubieus punt. Om mijn project te kunnen doen had ik een werkend systeem nodig. Omdat ik de enige was die ook maar een beetje wist hoe het systeem werkte was het mijn taak om het systeem draaiende te houden. Ik had daar eigenlijk twee taken een van requirements engineer en van systeembeheerder. Dit zorgde voor een aanzienlijke vertraging.

Reflectie
Nu aan het einde van het project vraag ik mij zelf af wat ik anders zou hebben gedaan als ik het nog eens kon doen. Ten eerst zou ik nooit hebben toegestemd dat ik verantwoordelijk zou zijn voor het draaiende houden van het systeem. Dit heeft me echt veel te veel tijd gekost en zorgde niet echt voor een onbevooroordeelde blik op het systeem.

Het project zou ook sneller klaar zijn geweest als ik niet te veel was in gegaan op de details. Ik heb nu geleerd dat het beter is om een wat algemener plaatje te hebben van het hele systeem dan een heel gedetailleerd plaatje van een kleinstukje van het systeem.

Zelf beschouwing

In dit gedeelte van het document beoordeel ik mij zelf en het project wat ik gedaan heb. Ik heb de afgelopen maanden erg veel geleerd niet alleen qua requirements engineering maar ook over hoe de bedrijfsvoering wordt gedaan binnen een callcenter. Tevens heb ik nu ook ervaring opgedaan met echter stakeholders en hoe lastig deze kunnen zijn.

De kwaliteit van dit project is goed aangezien Interswitch dit document echt goed kunnen gebruiken voor de evaluatie van andere systemen. Dit heeft er ook voor gezorgd dat Interswitch geen investeringen hoefde te doen van duizenden euro's . Echter kon ik dit niet zonder de hulp van mijn begeleiders aan de UvA. Een cijfer die ik zou geven voor dit project is dan ook een 7

De kwaliteit van dit document is redelijk tot goed. Het echte document is het Programme of Requirements. Dit document houd zich aan de voor opgestelde eisen. Eerst had ik dit document in het engels geschreven echter beviel dit niet zo erg dus is het herschreven in het Nederlands. Een cijfer voor dit document zou een 6,5 zijn voldoende maar niet perfect.

De probleemstelling die ik heb gekozen was erg moeilijk voor mij omdat mijn jaar nooit heft geleerd om een Programme of Requirements te schrijven. Gelukkig is dit met hulp van mijn begeleiders goed gelukt. Reverse Requirements Engineering is een tak van requirements engineering waar wel veel over gepraat is maar weinig is toegepast. Ik had ook niet veel voorbeelden en moest het wiel zelf uit vinden. Voor de probleem stelling zou ik me zelf een 7 willen geven.

De raakvlakken van dit project en de lessen die ik heb gehad verschillen. Kennis van een hoop vakken komen samen in dit project. Echter steekt het Requirements Engineering er met kop en schouder boven uit. Op de tweede plaat staat software evolution sinds dit project eigenlijk ging over de requirements van een legacy system. Sommige vakken werden niet echt gebruikt vakken als testing en software process hadden hier geen plaats. Qua relevantie zou ik dit project een 7 willen geven.

Literatuur

"DOD Legacy Systems: Reverse Engineering Data Requirements" by Peter Aiken, Alice Muntz, and Russ Richards appeared in the Communications of the ACM in May 1994 volume 37(5):26-41.

Rayson, P., Garside, R., and Sawyer, P. (1999). Recovering Legacy Requirements. In Proceedings of REFSQ'99. Fifth International Workshop on Requirements Engineering: Foundations of Software Quality, June 14-15 1999, Heidelberg, Germany. Published by University of Namur, pp. 49-54.

Rayson, P., Garside, R., and Sawyer, P. (1999). Language engineering for the recovery of requirements from legacy documents. REVERE project report, Lancaster University, May 1999.

Reverse Requirements Engineering with Multiple Models, Richards D., Proceedings of the First Asia Pacific Workshop on Intelligent Software Engineering , 23 November 1998, National University of Singapore, Singapore

Flexible Reverse Engineering of Web Pages with VAQUISTA, Jean Vanderdonckt, Laurent Bouillon, and Nathalie Souchon

Boeken

Het complete foxpro 2.5 boek, Charles Siegel, Sybex Inc; 2nd edition (July 1, 1991)
Murach's Structured COBOL, Mike Murach, Anne Prince, and Raul Menendez, Bk&CD-Rom edition (October 1, 2000)

Gebruikte tools

1ReverseC.....used for reverse engineering on C programs
http://www.plus-one.com/+1ReverseC_fact_sheet.html

CaliberRM .....Requirements management
http://www.borland.com/caliber/

CaliberRBT.....Test case design, Requirements based testing
http://www.bitspi.com/products/caliberrbt.htm

Analyst Pro..... Requirements engineering, Requirements tracing
http://www.analysttool.com/

Appendix I: Use cases

Use case 1 General service of Interswitch

Nog in te voeren plaatje

Use case 2 The desk operator

Appendices II: The shit list

Cobis

Probleem:.....Cobis is ontzettend traag met opstarten
Oplossing:.Er moet geïndexeerd worden. Dit gaat in de volgende stappen:

1. Iedereen uit Cobis
2. De faxapparaten uitzetten
3. Cobis e-mail uitzetten
4. FoxPro starten
5. Toets in: **_reindex** (dit zal ongeveer 25 minuten duren)
6. FoxPro verlaten door **quit** in te toetsen
7. Iedereen kan weer in Cobis
8. Faxapparaten weer aanzetten
9. Cobis e-mail weer aanzetten

Probleem: .....Factureren gaat niet
Oplossing: Is op scherm zichtbaar is dat de uitbeldatabase een bepaald aantal records bevat?

Ja Verwijder dan de volgende bestanden:

In de directory w:\cobis :	tempbood.cdx
	tempbood.dbf
In de directory w:\cobis\data\fact :	bel_*.dbf
	bel_*.cdx
	tot_*.dbf

Probeer de facturering nu opnieuw. Lukt dit niet, neem dan contact op met Officia.

Nee Waarschijnlijk kan het factureringsprogramma de directory w:\data\timdata**uitbel.dbf** niet vinden.
Re-boot de computer en probeer het opnieuw. Lukt dit niet neem dan contact op met Officia.
Let op! Klanten die gefactureerd worden, moeten wel servicerecht hebben.

Faxen

Probleem: Een van de faxen blijft in back-up stand en komt hier niet meer uit
Oplossing: Dit probleem is tijdelijk te verhelpen, totdat de systeembeheerder terug is.
Handel als volgt:

1. Zet de desbetreffende fax uit
2. Start FoxPro en toets in: **use fax_cfg**
3. Toets in: **edit**
4. Loop met pijltjestoets naar het veld: **Back dat**
5. Zet de datum zover vooruit tot de systeembeheerder terug is
6. Zet de fax weer aan

Probleem: *Op het logboek staat veelvuldig: unexpected char in modem result string*
Oplossing: Dit komt doordat de modem blijft hangen.
Handel als volgt:
1. Zet de desbetreffende fax uit
2. Laat deze ongeveer 1 uur uit staan en probeer het dan opnieuw

Probleem: *Oppiepen lukt niet*
Oplossing: schakel Fax-3 uit en weer aan.

E-mail

Probleem: *E-mail wordt niet meer verzonden*
Oplossing: Waarschijnlijk 'hangt' de Cobis e-mail server. Handel als volgt:
1. Re-boot de Cobis e-mail server. Dus helemaal uitzetten en vervolgens weer aanzetten. De e-mail server staat in de computerruimte.
2. Bij het login scherm kun je op Enter drukken om door te gaan.
3. Alles wordt nu verder vanzelf opgestart.

Printers

Probleem: *er kan niet meer geprint worden*
Oplossing: Janneke Re-boot de linker computer van Janneke, dus helemaal uit- en weer aanzetten.

Probleem: *Er kan niets meer worden opgezocht*
Oplossing: Waarschijnlijk is de netwerkverbinding verbroken. Start de desbetreffende computer opnieuw op (re-booten), dus helemaal uit- en weer aanzetten

Toegang controle systeem

Probleem: *Tijdregistratie van de telefonistes wordt niet meer gelogd*
Oplossing: Waarschijnlijk 'hangt' de computer van de tijdregistratie. Start de computer van
de tijdregistratie opnieuw op (re-booten), dus helemaal uit- en weer aanzetten.
Deze computer staat in de computerruimte, er staat een printer bovenop.
Het wachtwoord is: 1111.

Lopac maken diskette.

Frequentie.
Dagelijks gedurende circa 4 dagen tijdens invoeren lonen.
Doel.
Tijdens het maken van de loonadministratie elke dag een back-up maken van de bedrijfsgegevens. Het betreft bedrijf 20 en 30.
Hoe
Typ in het menu [F7]
Password: [geen]
Optie 4 en 5 is maken back-up van respectievelijk systeembestanden en bedrijfsgegevens.

Lopac teruglezen back-up diskette.

Doel.
Restore van de bedrijfsgegevens indien er een probleem met de loonadministratie is. Dit werkt niet als het programma zelf (Lopac) op de een of andere wijze corrupt geworden is. Deze restore is bedoeld voor regels en bedrijfsgegevens.

Hoe

Typ In het menu [F7]
Password [geen]
Optie 12 en 13 restore.

Technische achtergrond.
Lopac heeft geen echte back-up programma om de backup te realiseren. Er wordt eenvoudigweg gebruik gemaakt van de functie pkzip om bestanden te compressen en op diskette te zetten. Deze methode blijkt uitstekend te werken.