Afstudeerverslag

Het ontwerpen, bouwen en testen van een generieke engine die de koppeling tussen BOXwise en een willekeurig ERP systeem mogelijk maakt.

Student : *Ray Wijshake*
Studentnummer : *20050335*

Afstudeerverslag TranCon

Student	:	*Ray Wijshake*
Studentnummer	:	*20050335*
Opleiding	:	*Informatica*
Onderwijsinstelling	:	*Haagse Hogeschool*
Locatie	:	*Zoetermeer*
Examinatoren	:	*Ellen Wesselingh, Tim Cocx*
Afstudeerperiode	:	*9 februari 2010 – 4 juni 2010*
Bedrijf	:	*TranCon bv*
Adres	:	*Tarwestraat 25, Nieuw Vennep*
Begeleider	:	*Frank van den Hoek*
Datum	:	*3 juni 2010*
Versie	:	*1*

Voorwoord

Na 17 weken is deze opdracht ook weer afgerond. Ik weet nog goed toen ik voor het eerst bij TranCon kwam. Een nieuw bedrijf, nieuwe mensen, nieuwe namen, en vooral veel neerslag. Inmiddels is het alweer bijna zomer en is het werken bij TranCon een ritme geworden. De tijd vliegt voorbij.

Tijdens deze opdracht heb ik weer veel geleerd. Door te werken met een bestaand software pakket heb ik inzicht gekregen wat er bij komt kijken om deze aan te passen. Ik heb veel gewerkt aan de nieuwbouw van software, maar deze opdracht heeft bewezen dat het bouwen op bestaande software ook een leuke uitdaging kan zijn.

Het werken bij TranCon was ook een ervaring op zich. In tegenstelling tot mijn eerste stage werkte ik nu bij een veel kleiner bedrijf. Hierbij heb ik vooral ondervonden dat het werken in een klein team anders is dan het werken op hele ontwikkel afdeling.

Bij deze wil ik ook de gelegenheid gebruiken om Frank te bedanken. Frank was mijn bedrijfsmentor en ik heb veel van hem geleerd. Ook wil ik Jeff, Jurgen en Vincent bedanken. Hoewel zij minder bij mijn opdracht betrokken waren heb ik toch van hun kunnen leren. Tot slot wil ik ook Annemieke bedanken. Annemieke heeft ook haar afstudeeropdracht bij TranCon gedaan en zo hebben wij af en toe met elkaar kunnen praten over de opdrachten.

Al met al is de tijd voorbij gevlogen, en ik ga hierna met volle enthousiasme aan de slag bij TranCon.

Ray Wijshake
Zoetermeer, 3 juni 2010

Inhoudsopgave

BIJLAGE A: PLAN VAN AANPAK

BIJLAGE B: INCEPTION RAPPORT

BIJLAGE C: ELABORATION RAPPORT

BIJLAGE D: TESTPLAN

BIJLAGE E: CD MET ALLE PRODUCTEN

1. Inleiding

Het verslag dat hier voor u ligt is geschreven als eindverslag van mijn afstudeeropdracht bij het bedrijf TranCon. In het verslag zal ik beschrijven wat mijn opdracht was en hoe ik het heb uitgevoerd. Daarbij zal ik ook mijn keuzes toelichten.

Allereerst zal ik in hoofdstuk 2 achtergrondinformatie geven over TranCon en de producten die zij ontwikkelen en verkopen (CARTwise en BOXwise). Hierbij zal ik dieper op het product BOXwise ingaan omdat dit het product is waaraan ik heb gewerkt.

In hoofdstuk 3 zal ik eerst mijn afstudeeropdracht bespreken.

In hoofdstuk 4 zal ik mijn aanpak van dit project toelichten. Hierbij licht ik toe welke methodieken en technieken ik heb gebruikt en waarom ik deze heb toegepast.

In hoofdstuk 5 ga dieper in op de opdracht zelf. Hierbij beschrijf ik hoe ik het project ben gestart en de basis heb gelegd voor de verdere ontwikkeling van de software.

In hoofdstuk 6 licht ik het ontwikkelproces van het eerste deelproject toe: de interface. Dit vormt de basis voor BOXwise om op een generieke manier met ERP systemen te kunnen communiceren.

In hoofdstuk 7 ga ik verder met het tweede deelproject: de koppeling. Dit biedt BOXwise de mogelijkheid om weer met Exact te communiceren, gebruikmakend van de interface.

In hoofdstuk 8 evalueer ik tot slot het project waarbij ik terug kijk naar mijn aanpak en opgeleverde producten.

2. TranCon en zijn producten

Voordat de afstudeeropdracht wordt besproken, zal ik eerst informatie geven over het bedrijf TranCon en de producten die zij ontwikkelen: CARTwise en BOXwise. Hierbij zal ik vooral dieper op het product BOXwise ingaan. Dit is namelijk het product waarmee ik tijdens mijn opdracht heb gewerkt.

2.1 TranCon bv

Het bedrijf TranCon bv bestaat sinds 2006. TranCon ontwikkelt software voor de logistieke markt waarbij gebruiksvriendelijkheid hoog in het vaandel staat. Het is een klein bedrijf met zeven medewerkers. Hiervan zijn er twee leidinggevenden en twee vaste programmeurs in dienst.

TranCon heeft twee producten: CARTwise en BOXwise. Deze zullen in de volgende paragrafen worden toegelicht.

2.2 CARTwise

CARTwise is een webwinkel-oplossing waarmee bedrijven hun producten en/of diensten eenvoudig op een aantrekkelijke manier op het internet kunnen aanbieden. Dit is bereikt door o.a. een eenvoudige user interface en het kunnen toepassen van een eigen huisstijl.
CARTwise heeft een standaard koppeling met het ERP systeem Exact Globe 2003. Dit heeft als voordeel dat gegevens zoals artikel en prijs direct uit het ERP systeem worden gehaald waardoor dubbele invoer wordt vermeden.

2.3 BOXwise

BOXwise is een magazijnbeheer oplossing voor het mkb. Door technieken als touchscreen en barcodescanning te combineren met een gebruiksvriendelijke userinterface is het voor magazijnmedewerkers eenvoudig om magazijntaken uit te voeren. De software heeft een lage leercurve waardoor nieuwe medewerkers er snel mee kunnen werken.
BOXwise werkt naadloos samen met Exact Globe 2003 en is daarvoor ook gecertificeerd door Exact Software. Alle gegevens worden rechtstreeks uit Exact Globe gehaald waardoor het altijd up-to-date is.

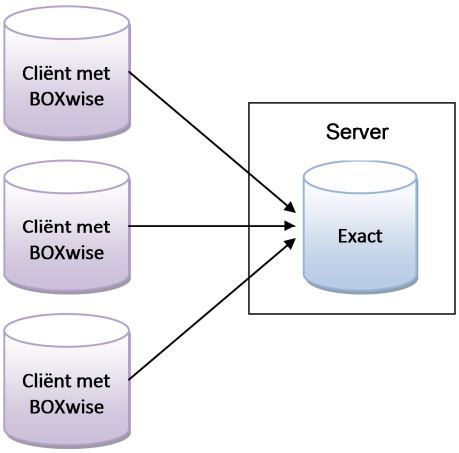
BOXwise is in twee versies verkrijgbaar:

- BOXwise Pick & Pack
- BOXwise Pro.

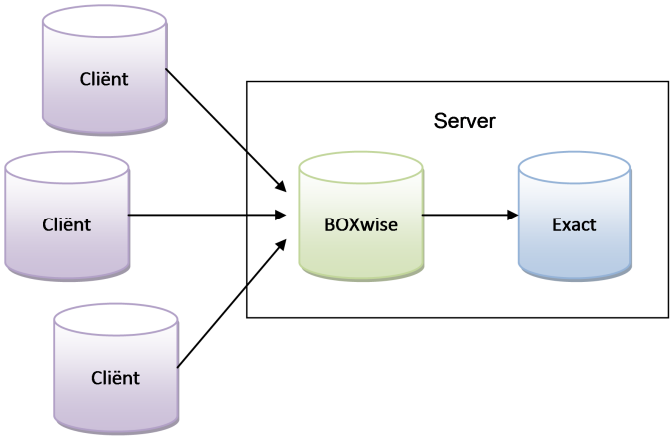
BOXwise P&P (hierna te noemen P&P) en Pro werken beiden op een verschillende wijze:

BOXwise P&P	BOXwise Pro
- Lokaal geïnstalleerd	- Geïnstalleerd op een server
- Elke cliënt (touchscreen pc) heeft een eigen BOXwise installatie	- Verschillende cliënts, zoals wireless scanners, die slechts een gebruikers interface hebben.
- Alle bewerkingen worden op de cliënt uitgevoerd	- Alle bewerkingen worden op de server uitgevoerd.
- Elke cliënt heeft een verbinding met Exact	- Elke cliënt heeft verbinding met BOXwise. BOXwise hanteert de verbinding met Exact.
- Kan snapshots creëren van openstaande verkooporders.	

Schematisch zien BOXwise P&P en Pro er als volgt uit:



Afbeelding 1: Schematische weergave BOXwise P&P



Afbeelding 2: Schematische weergave BOXwise Pro

Het grootste verschil tussen de twee versies is dat BOXwise P&P werkt met een snapshotsysteem voor het verwerken van de orders. Dit houdt in dat er een momentopname wordt gemaakt van de verkooporders. Het wordt gebruikt om op een intelligente manier de voorraad (aanbod) tegen de verkooporders (vraag) af te bouwen.

Functionaliteiten

BOXwise heeft een aantal functionaliteiten die globaal te verdelen zijn in drie groepen:

- Ontvangsten
- Leveringen
- Inventarisatie

Elke groep ondersteunt het desbetreffende proces in een magazijn. Per groep zal er aan de hand van een situatieschets worden toegelicht wat de functionaliteit precies inhoudt.

Ontvangsten

Voorbeeld: een leverancier levert nieuwe producten aan, die vervolgens moeten worden geregistreerd en opgeslagen.

In een magazijn worden producten op voorraad gehouden, zodat deze kunnen worden uitgeleverd aan klanten zodra zij het nodig hebben. Deze producten kunnen een aantal standaard producten zijn, maar ook producten die specifiek voor een klant zijn.

Als een product wordt aangeleverd, moet het worden geregistreerd in het ERP systeem. Deze administratieve handeling is nodig omdat deze informatie financieel (wat is de waarde van de voorraad) en commercieel (wat is de voorraad) van groot belang is. Bij het registreren kan ook worden opgeslagen op welke locatie de goederen zijn geplaatst.

BOXwise ondersteunt dit proces in een vijftal stappen:

- Alle inkooporders worden gesorteerd op naam van de leverancier.
- Per leverancier kan worden geselecteerd welke orders geopend moeten worden.
- De inhoud van de order wordt weergeven.
- De magazijnmedewerker scant de producten waardoor de producten worden geregistreerd.
- Afronding van het proces.

Leveringen

Voorbeeld: een klant plaatst een order, de producten moeten vervolgens worden verzameld en verstuurd.

Producten in een magazijn worden uitgeleverd aan een klant zodra deze een order plaatst. De producten van de order moeten worden verzameld zodat het naar de klant kan worden gestuurd. Met BOXwise kan dit in vier stappen:

- Alle orders van dezelfde klant worden samengevoegd tot één levering. Een klant kan namelijk meerdere orders plaatsen.
- Per klant kan worden bekeken welke producten geleverd moeten worden. Hierbij kan een looplijst worden uitgeprint waarin staat welke producten er moeten worden verzameld, wat de hoeveelheid is en wat de locatie is. Hierbij is de meeste optimale route berekend waardoor er minder tijd nodig is om alle producten te verzamelen.
- Nadat de producten zijn verzameld, worden de barcodes gescand. Hierdoor wordt er geregistreerd welke producten zijn verzameld, en of de juiste hoeveelheid is verzameld. Als dit gereed is, worden de orders verwerkt in het ERP systeem.
- De laatste stap is het printen van de labels voor het transport. Op de labels staat het afleveradres plus eventuele andere relevante informatie. Veelal zijn de labels voorzien van een barcode dat wordt gebruikt door de transporteur.
  BOXwise ondersteunt ongeveer 250 transporteurs (waaronder UPS, DHL, TNT en GLS) waardoor er geen aparte software moet worden geïnstalleerd.

Inventarisatie

Voorbeeld: in verband met de jaarlijkse controle van de voorraad moet er een voorraad telling worden gedaan in het magazijn.

Het is mogelijk dat de fysieke voorraad van artikelen niet overeen komt zoals deze in het ERP systeem is geregistreerd. Dit kan verschillende onderzaken hebben zoals het niet juist verwerken van product afnamen. Daarom moeten er tellingen worden uitgevoerd zodat de voorraad die in het ERP systeem is geregistreerd, ook overeen komt met de plank voorraad.

Met BOXwise kan een magazijnmedewerker de tellingen in een paar stappen uitvoeren:

- Eerst wordt het magazijn geselecteerd. Het is mogelijk dat een bedrijf meerdere magazijnen heeft.
- Vervolgens wordt er een lijst met producten die in het desbetreffende magazijn zijn opgeslagen, opgehaald.
- Met de deze lijst worden de producten handmatig geteld, waarna de resultaten worden geregistreerd.
- Na afloop worden de resultaten verwerkt in BOXwise, die het op zijn beurt weer verstuurt naar het ERP systeem. Nadat de gegevens in het ERP systeem zijn verwerkt, komt de voorraad in het ERP systeem overeen met de werkelijke fysieke voorraad.

2.4 Exact Globe 2003

Exact Holding nv is een bedrijf dat software maakt voor onder andere de boekhouding, Customer Relationship Management en Enterprise Resource Planning. Dit alles heeft als doel om de bedrijfsprocessen beter te structureren. Één van hun producten is Exact Globe 2003, een ERP oplossing. Hiermee kan een bedrijf de gehele administratie automatiseren. [1]

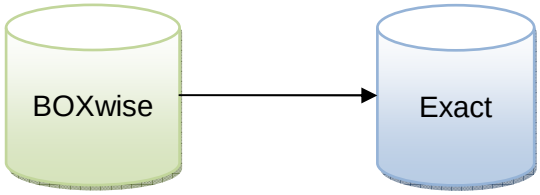
[1] Bron: http://exact.com/nl/

3. De afstudeeropdracht

Voordat ik ga beschrijven welke activiteiten ik heb uitgevoerd , zal ik eerst de afstudeeropdracht toelichten.

3.1 Aanleiding

BOXwise is naadloos geïntegreerd met het ERP systeem Exact Globe 2003. Alle gegevens worden real-time weergeven, en de voorraad wordt direct in Exact verwerkt.
De intelligentie die de koppeling tussen BOXwise en Exact mogelijk maakt is een onderdeel van BOXwise zelf. BOXwise kan op deze manier prima functioneren, maar het is niet flexibel. Dit is schematisch weergeven in de onderstaande afbeelding.



Afbeelding 3: Schematische weergave tussen BOXwise en Exact

3.2 Probleemstelling

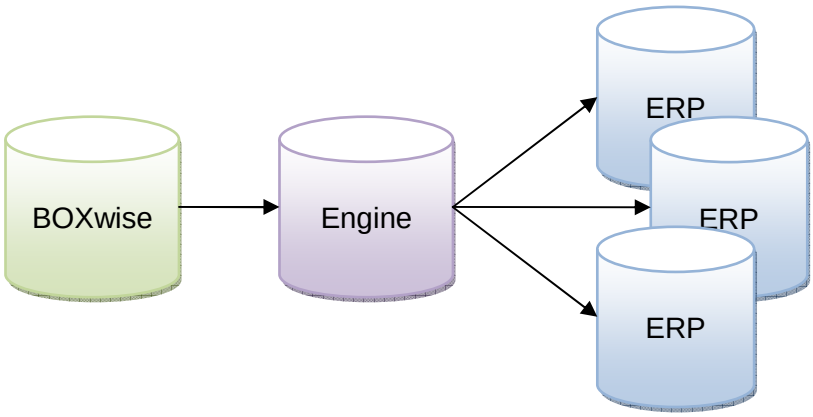
BOXwise moet worden uitgebreid met de mogelijkheid om met andere ERP systemen te kunnen communiceren dan alleen Exact Globe 2003. Het moet dermate flexibel zijn dat het in een later stadium eenvoudig met een ERP systeem kan communiceren.

3.3 De opdracht

Het ontwerpen, bouwen en testen van een generieke engine die de koppeling tussen BOXwise en een willekeurig ERP systeem mogelijk maakt.

3.4 Doelstelling

Een uitbreiding van BOXwise waardoor het op een generieke wijze kan communiceren met elk willekeurig ERP systeem.



Afbeelding 4: Schematische weergave tussen BOXwise en meerdere ERP systemen

4. Aanpak

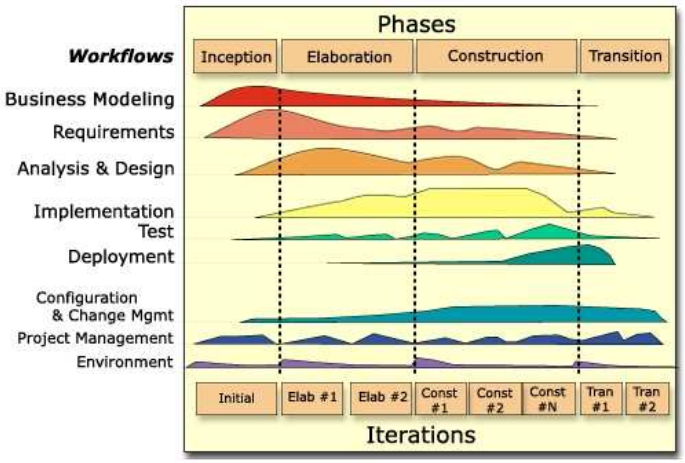
In dit hoofdstuk zal ik toelichten welke methodieken ik tijdens het project heb toegepast. Daarnaast zal ik beschrijven welke technieken ik heb gebruikt om de software te ontwikkelen.

4.1 Projectmethodiek

Ik heb ervoor gekozen om met RUP te werken (Rational Unified Process). Het is een ontwikkelmethodiek voor softwareontwikkeling. Ik heb deze methodiek gekozen omdat er ik al ervaring mee heb. Hierdoor kon ik RUP meteen efficiënt toepassen. Daarnaast gaf het mij de gelegenheid om TranCon kennis te laten maken met RUP.

RUP deelt een ontwikkelproject in een aantal fases:

1. *Inception*
   Verkennen van het project
2. *Elaboration*
   Het ontwerp
3. *Construction*
   De bouw
4. *Transition*
   Afronding



Afbeelding 5: Fases en disciplines van RUP

Een belangrijk aspect van RUP is het iteratief werken. Bij iteratief werken wordt het te ontwikkelen programma opgesplitst in kleinere delen, meestal bepaald aan de hand van de functionaliteiten. Een dergelijke opsplitsing, een iteratie genaamd, wordt vervolgens uitgewerkt (ontwerp, bouw, test), waarna er wordt gecontroleerd of nog aan alle eisen is voldaan, en of het goed werkt. Zo niet, dan wordt de iteratie nogmaals doorlopen om het wel in orde te krijgen. Vervolgens start de volgende iteratie die weer dezelfde cyclus doorloopt. Zo wordt een programma in delen opgeleverd, wat als voordeel heeft dat het desbetreffende deel werkt zoals het in de eisen is vastgelegd. Indien aan alles tegelijk wordt gewerkt zonder terug te kijken, dan is een programma minder robuust.

RUP is echter nog veel uitgebreider, waarbij het ook mogelijk is om o.a. rollen en verantwoordelijkheden toe te wijzen, en gebruik te maken van workflows. Gezien het feit dat ik het project alleen heb uitgevoerd, heb ik er voor gekozen om slechts een beperkt deel van RUP te gebruiken. De door RUP gedefinieerde fases en het gebruik maken van iteraties leek mij gezien de omvang van het project voldoende om mee te werken.

Aan de hand van de RUP fases heb ik mijn afstudeerproject gepland, waarbij er een aantal iteraties zijn gedefinieerd. De iteraties heb ik gebaseerd op de drie groepen functionaliteiten van BOXwise (ontvangsten, leveringen en inventarisatie). Op deze manier heb ik BOXwise stapsgewijs kunnen aanpassen.

Ik ben begonnen met de Inception fase. In deze fase ben ik het project begonnen door de werking van BOXwise te verkennen. Aangezien ik een uitbreiding moest ontwikkelen was het noodzakelijk dat ik begreep hoe BOXwise werkt en wat het doet. Daarnaast bekeek ik wat de eisen waren zodat ik wist wat ik precies moet ontwikkelen, en keek ik alvast vooruit naar eventuele ideeën voor het ontwerp. De Inception fase heb ik afgerond met een Inception rapport waarin ik de resultaten van de voorgaande werkzaamheden heb beschreven.

Vervolgens ben ik gestart met de Elaboration fase. Hier heb ik mij meer verdiept in de toekomstige situatie door het ontwerp op te stellen. In de Construction fase heb ik de software daadwerkelijk gebouwd waarbij ik ook begon met de Transition fase. De laatst genoemde fase omvatte het testen en documenteren van de software. De Elaboration, Construction en Transition fase heb ik meerdere malen doorlopen. Dit zal verder worden toegelicht in hoofdstuk 6 en 7.

4.2 Ontwerpen van de software

Het opstellen van het ontwerp heb ik gedaan tijdens de Elaboration fase. Ik had een duidelijk beeld hoe BOXwise in elkaar zat door het uitvoeren van de Inception fase. Door me in deze fase te richten op het ontwerp heb ik kunnen na gaan of er met alle eisen rekening was gehouden en hoe de software zou gaan werken. Ook stelde het mij in staat eventuele problemen te signaleren. Uiteindelijk had dit als doel om ervoor te zorgen dat ik de software kon bouwen waarbij van tevoren al was nagedacht over de werking. Zo kon er worden voorkomen dat de software niet voldeed aan de eisen.

Om de software te ontwerpen heb ik gebruik gemaakt van *UML*[2] diagrammen. UML staat voor Unified Modeling Language en is een modelleertaal om ontwerpen van informatiesystemen te kunnen maken. Dit wordt vaak gebruikt met RUP en zelf heb ik er al enige ervaring mee. Daarnaast is UML bekend in het bedrijfsleven, dus daarom heb ik ervoor gekozen om hiervan gebruik te maken.
Ik heb ervoor gekozen om twee diagrammen van UML te gebruiken: klassen- en sequentie diagrammen. Deze leken mij geschikt voor mijn opdracht en om het ontwerp weer te geven. Klassen diagrammen heb ik gebruikt om te bepalen welke specifieke gegevens nodig waren voor de informatiestromen. Sequentie diagrammen heb ik gebruikt om de informatiestromen in kaart te brengen, en welke objecten daarbij waren betrokken. De informatiestromen en objecten en het gebruik van de diagrammen zullen in hoofdstuk 5 en 6 worden toegelicht.

In eerste instantie wilde meer gebruik maken van UML door meerdere soorten diagrammen toe te passen. Maar ik kwam er achter dat deze geen toegevoegde waarde boden voor het ontwerp. Daarom heb ik het bij de klassen- en sequentie diagrammen gehouden.

4.3 Bouwen van de software

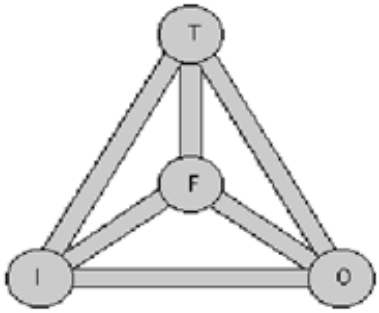
Voor het ontwikkelen van de software heb ik gebruik gemaakt van het .Net framework van Microsoft. Hierbij gebruikte ik de programmeertaal C#. BOXwise is ook met deze technieken gebouwd, daarom was het logisch om daarmee verder te gaan.

Het gebruik van .Net en C# werd daarmee ook meteen een vereiste voor het ontwikkelen van de koppelingen. Een ontwikkelaar kan er zelf voor kiezen om de interne werking van een koppeling te schrijven met een techniek naar keuze, maar uiteindelijk moet hij toch gebruik maken van .Net en C#. Anders kan BOXwise geen gebruik maken van de koppeling.

[2] Bron: http://nl.wikipedia.org/wiki/Unified_Modeling_Language

4.4 Testmethodiek

Om te garanderen dat de software goed werkt, heb ik ook een testtraject opgesteld. Dit is gedaan met behulp van TMap.



Afbeelding 6: De 4 pijlers van TMap

TMap staat voor Team Management Approach en is een gestructureerde testmethodiek. Met behulp van TMap is het mogelijk om vanaf het begin van een software ontwikkeltraject actief bezig te zijn met testen.

Om gestructureerd te testen is het testproces onder te verdelen in vier aspecten, ook wel de vier pijlers van TMap genoemd (afbeelding 6):

- *Fasering*
  Een testtraject bestaat uit meerdere activiteiten die allemaal gepland moeten worden. Deze worden onderverdeeld in verschillende fases. Hiervoor is een TMap faseringsmodel aanwezig die parallel loopt aan het ontwikkelproces (afbeelding X).

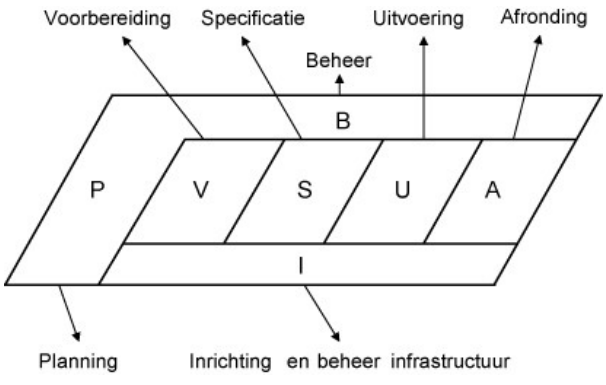
- *Organisatie*
  Bij het een testtraject zijn er ook mensen bij betrokken die allemaal hun eigen taken hebben. Dit vereist een goede organisatie om alles in goede banen te leiden.

- *Infrastructuur*
  Om tests te kunnen uitvoeren, is er een testomgeving nodig met de hulpmiddelen om tests goed te kunnen uitvoeren.

- *Techniek*
  Er is een verscheidenheid aan technieken beschikbaar om tests uit te voeren. TMap biedt een goedgevulde “gereedschapskist” met testtechnieken die gebruikt kunnen worden.

Ik heb voor TMap gekozen zodat ik gestructureerd een testproces kon opzetten. Daarnaast had ik er al enige ervaring mee. Ik heb slechts een beperkt deel van TMap gebruikt aangezien ik dit project alleen heb uitgevoerd. Ik heb de nadruk gelegd op de fasering en de technieken omdat deze twee pijlers het meeste van toepassing waren op dit project.

Voor de fasering is afbeelding 7 te gebruiken. Ik ben begonnen met een testplan waarin ik heb beschreven wat ik zou gaan testen en wat de planning daarbij zou zijn (Planning en Voorbereiding). De planning voor het testen heb ik tegelijk met het ontwikkelproces laten lopen. Zo kon ik direct de nieuwe broncode testen en eventuele fouten snel ontdekken en oplossen. Vervolgens bekeek wat ik wilde testen (Specificatie). Dit deed ik aan de hand van de simpele vraag: “*wat wil ik testen?*”. Daarbij bekeek ik ook welke testtechnieken geschikt waren om die tests uit te voeren. Vervolgens heb ik in de fase Uitvoering de tests opgesteld en uitgevoerd. Uiteindelijk heb ik het afgerond door in een document de resultaten te vermelden. In hoofdstuk 6 en 7 zullen deze activiteiten verder worden toegelicht.



Afbeelding 7: Het TMap-faseringsmodel

4.5 Bewaken van de voortgang

Voor dit afstudeerproject stond een vast aantal weken gepland (17 weken). Daarbij was het de bedoeling dat aan het einde van het project een werkend product werd opgeleverd. Daarom vond ik het belangrijk dat er op de voortgang werd gelet. Mochten er signalen zijn dat er te weinig tijd was om de opdracht af te ronden, dan moesten er meteen maatregelen genomen worden. Ik heb de volgende werkwijzen toegepast om de voortgang in de gaten te houden:

- *Planning*
  Van tevoren heb ik een planning opgesteld waarin ik aangaf welke activiteiten uitgevoerd moesten worden, en hoeveel tijd ik dacht dat het kostte. Echter, de tijd die ik dacht nodig te hebben voor bepaalde activiteiten, nam ik niet zeer strikt. De deadlines die ik voor bepaalde producten opstelde, waren strikter. Het was geen probleem als een product eerder werd opgeleverd, maar wel als het later zou worden opgeleverd.

- *Voortgangsgesprekken*
  De voortgangsgesprekken hield ik met mijn begeleider. Deze waren eens per week of twee weken, afhankelijk hoe druk hij het had. Tijdens de gesprekken vertelde ik wat ik had gedaan, wat er nog ging gebeuren, en stelde ik vragen over bepaalde zaken helder te krijgen. Teven bood dit de mogelijkheid om te bepalen of ik nog op schema lag, of dat er maatregelen moesten worden genomen omdat een deadline in gevaar lag.

4.6 Competenties

Bij deze afstudeeropdracht heb ik ook een aantal competenties uitgezocht. Door deze competenties uit te voeren tijdens mijn afstudeerproject heb ik kunnen aantonen dat ik in staat ben om het daadwerkelijk uit te voeren volgens met een daarbij behorende aanpak. De onderstaande competenties heb ik uitgekozen omdat deze mij relevant leken voor de opdracht:

Uitvoeren analyse door definitie van requirements
Deze competentie vond ik geschikt omdat het belangrijk is om alle eisen helder te hebben. Anders bestond de kans dat er software werd opgeleverd waarbij bepaalde functionaliteiten niet waren geïmplementeerd, of dat het simpelweg niet overeen kwam met het gewenste resultaat.

Ontwerpen systeemdeel
Er moest een nieuwe software component worden gebouwd zodat BOXwise met andere ERP systemen kan communiceren. Deze competentie heb ik gekozen om te demonstreren omdat het in feite de kern van de opdracht is. Er moest iets worden gebouwd om BOXwise meer generiek te maken.

Bouwen systeemdeel
Het doel was dat er een systeemdeel zou worden gebouwd om een generieke koppeling tussen BOXwise en ERP systemen mogelijk te maken. Deze competentie is daar geschikt voor om aan te tonen dat ik op een gestructureerde manier de software kan bouwen.

Initiëren en plannen van het testproces
Om een goede werking van de koppeling te garanderen, is het van belang dat deze wordt getest. Vooral ook gezien het feit dat BOXwise al door klanten wordt gebruikt. Daarom past deze competentie ook goed bij deze afstudeeropdracht. Met deze competentie wilde het ik testproces starten.

Uitvoeren van en het rapporteren over het testproces
Deze competentie is een logische stap na het initiëren en plannen van het testproces. Hierbij heb ik de tests opgezet en uitgevoerd.

5. De start van het project

Het project ben ik gestart met de Inception fase. In deze fase wilde ik vooral helder krijgen hoe BOXwise werkt en wat er moet gebeuren. Door dit in een vroeg stadium uit te voeren kan ik snel een overzicht creëren zodat ik weet wat ik precies moet doen. In de volgende paragrafen zal ik toelichten wat ik precies heb gedaan.

5.1 Plan van aanpak

Ik heb mijn project gestart met het opstellen van een Plan van Aanpak (PvA). Hiermee wilde ik de betrokken personen op de hoogte brengen van mijn aanpak. Dit heb ik gedaan door eerst aan te geven hoe ik mijn opdracht zou gaan plannen. Door aan te geven dat ik met RUP ga werken en de RUP fases gebruik voor de planning, kon ik meteen globaal een indruk geven hoe het ontwikkelproces er uit zou komen te zien. Daarbij vermeldde ik ook het gebruik van UML voor het ontwerp. Zo wist men wat ik zou gaan gebruiken om het ontwerp vorm te geven. Vervolgens gaf ik aan dat ik mijn software zou gaan testen om te garanderen dat het goed werkt. Daarbij zou ik gebruik gaan maken van TMap zodat gestructureerd kan gaan testen. Omdat ik me realiseerde dat er ook een overdracht plaats zou vinden, heb ik aangegeven dat ik documentatie zou gaan opstellen. In die documentatie zou ik beschrijven hoe wat mijn ontwerp was en hoe het werkt. Hierdoor zou het voor de andere ontwikkelaars gemakkelijker zijn om het product te begrijpen zonder de gehele broncode door te nemen. Daarnaast kon ik ook daar eventuele keuzes in vastleggen zodat het achterliggende idee van ontwerp werd toegelicht.

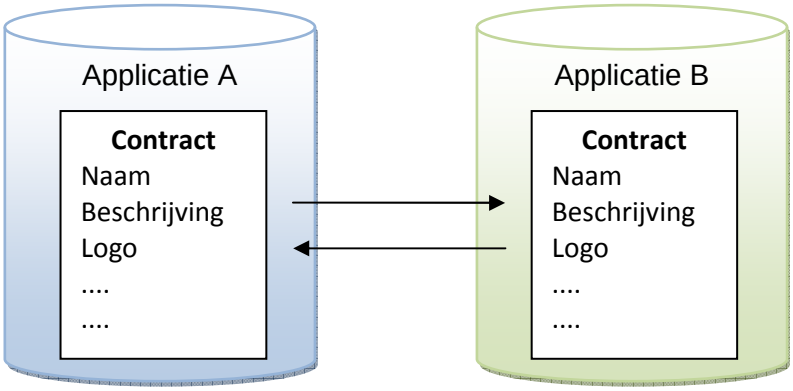
Tot slot voegde ik een planning toe waarin ik aangaf hoe lang ik dacht bezig te zijn met een bepaalde fase. In paragraaf 3 zal ik dieper ingaan op de planning waarbij ik toelicht waarom ik voor al deze activiteiten heb gekozen. Voor inzage is het Plan van Aanpak toegevoegd als Bijlage A.

5.2 Inventariseren van ideeën

In de eerste week begon ik al met het verkennen van BOXwise Pick & Pack. Er bleken twee versies van BOXwise te zijn (P&P en Pro) waarbij mijn werkzaamheden bij P&P zouden plaats vinden. Ik kreeg een uitleg van de werking waarbij ook bestaande ideeën ter sprake kwamen. De wens om BOXwise met meerdere ERP systemen te kunnen laten communiceren bestond namelijk al langer. Daarbij had men ook al nagedacht over mogelijke ideeën. Één daarvan was gebaseerd op een bestaande functionaliteit in BOXwise.

BOXwise heeft al een generiek systeem om labels uit te kunnen printen. Die labels worden gebruikt als een bedrijf goederen naar een klant wil verzenden. Voor het verzenden wordt er gebruik gemaakt van de diensten van een transporteur. Elke transporteur heeft zijn eigen labels die worden gebruikt om goederen te kunnen versturen. Gebruikers van BOXwise kunnen die labels met BOXwise uitprinten. BOXwise ondersteunt dit met een generieke manier genaamd *shipping layers*. Shipping layers bieden BOXwise extra functionaliteit aan om labels voor transporteurs uit te kunnen printen. Een enkele shipping layer geeft BOXwise de mogelijkheid om een label van een specifieke transporteur uit te printen. Zo is er dus een shipping layer voor DHL, maar ook één voor UPS. Het principe werkt door het gebruik van een *interface*[3]. Een interface definieert een zogenaamd contract tussen twee applicaties waardoor zij op een gestandaardiseerde manier met elkaar kunnen communiceren. Dit is schematisch weergeven in afbeelding:

[3] Interface: http://nl.wikipedia.org/wiki/Interface



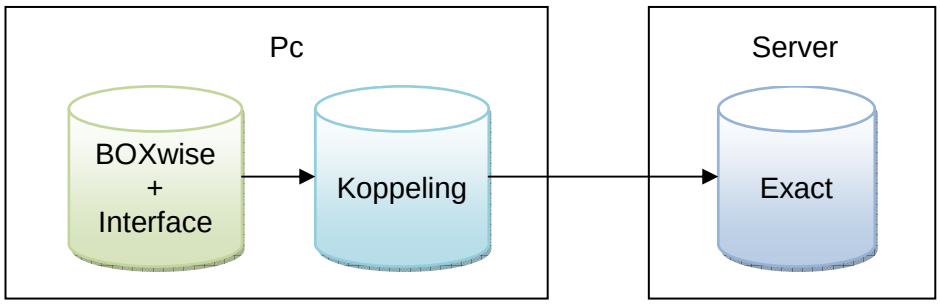
Afbeelding 8: Schematische weergave van een interface

Applicatie A definieert het contract met acties waarna applicatie B het contract moet overnemen. Applicatie A kan vervolgens via een aantal vastgestelde acties communiceren met applicatie B. Applicatie B weet op zijn beurt op welke manieren van hem gebruikt kan worden doordat de acties van het contract zijn overgenomen. Het gebruik van een interface maakt dus gestandaardiseerde communicatie mogelijk.

Dit idee was ook wenselijk voor de communicatie met een ERP systeem. BOXwise bevatte zelf de broncode om verbinding met Exact te maken. Dit wordt ook wel de koppeling genoemd: het zorgt voor een koppeling tussen BOXwise en Exact. Door de koppeling van BOXwise te scheiden en apart aan te leveren, zou het mogelijk zijn om met elk willekeurig ERP systeem te kunnen communiceren. Door het gebruik van een interface kon BOXwise op een gestandaardiseerde manier communiceren met een koppeling. Daarom heb ik besloten om dit idee te gaan gebruiken voor mijn opdracht: de werking is eenvoudig maar doeltreffend. Door het gebruik van een interface de term *engine* komen te vervallen. Die term was gebruikt omdat het niet geheel helder was hoe BOXwise op een generieke manier zou gaan communiceren met andere ERP systemen.

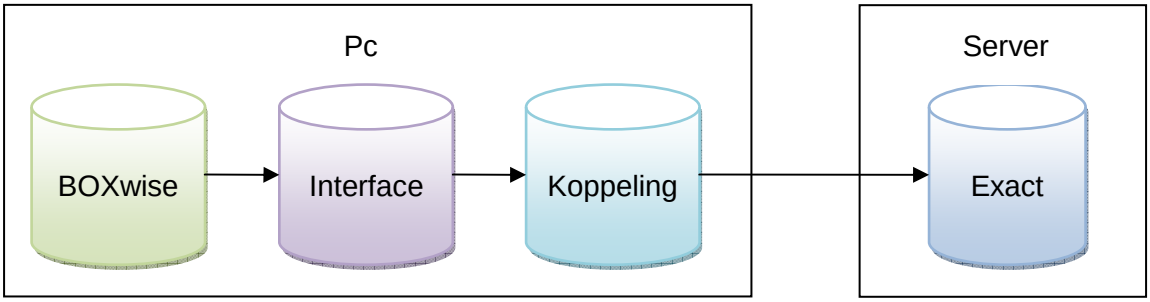
Daarna kreeg ik te maken met de vraag waar de interface en koppelingen zou gaan plaatsen. Omdat BOXwise P&P lokaal op een pc werkt, en Exact (of een ander ERP systeem) op een server, waren er verschillende mogelijkheden. Daarom heb ik kort bekeken waar ik de interface en koppelingen zou gaan plaatsen. Hierdoor kon ik eventuele ideeën opdoen over de verdere uitwerking. Door terug te kijken naar de eerdere projecten die ik heb gedaan kon ik een aantal schetsen maken:

Scenario 1: lokaal, interne interface + externe koppeling
In BOXwise zelf zou de interface worden aangemaakt. Zo was het eenvoudig toegankelijk voor BOXwise. De koppeling zou aanwezig zijn op de pc zelf waardoor het voor BOXwise wederom eenvoudig te benaderen was.



Afbeelding 9: Interne interface, externe koppeling (lokaal)

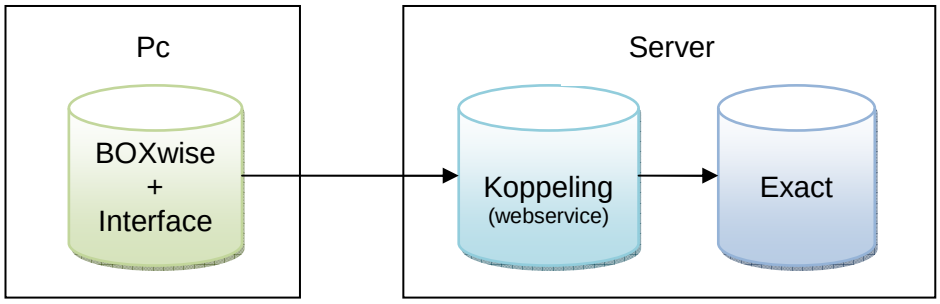
Scenario 2: lokaal, externe interface en koppeling
Dit is hetzelfde idee als het voorgaande idee, maar de engine is in dit geval een apart bestand. Dit kon wellicht handig zijn voor het beheer.



Afbeelding 10: Externe interface en koppeling (lokaal)

Scenario 3: webservices
BOXwise krijgt wederom een interface, die vervolgens een webservice aan zal spreken om gegevens van een ERP systeem op te vragen. Een *webservice*[4] bevat ook bepaalde functionaliteiten, maar heeft als voordeel dat deze binnen een bedrijfsnetwerk op een server of zelfs op het internet kan worden gezet. Hierdoor zijn de functionaliteiten in één keer beschikbaar voor alle BOXwise cliënts in een bedrijf. Dit is ook een pluspunt voor het beheer.

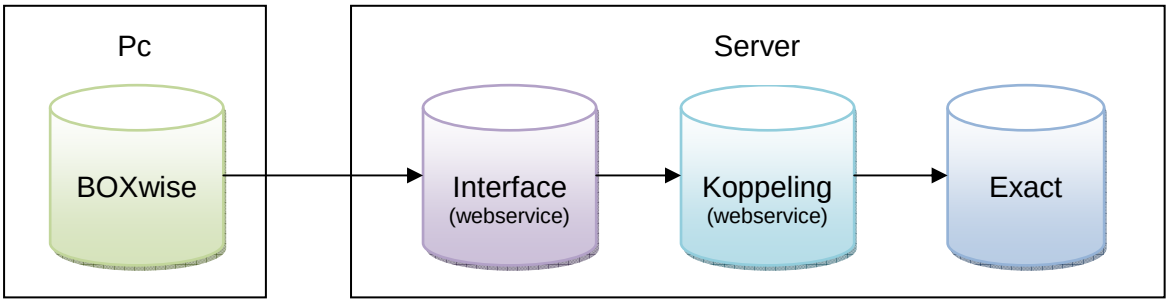
Voor het uitwisselen van gegevens wordt er bij webservices standaard XML bestanden gebruikt. Dit heeft als voordeel dat de gegevens altijd op een gestructureerde manier worden uitgewisseld, maar dit betekent ook dat er een extra slag nodig is om de gegevens weer beschikbaar te maken voor BOXwise. Dit kan een performance probleem geven.



Afbeelding 11: Interne engine en koppeling als webservice

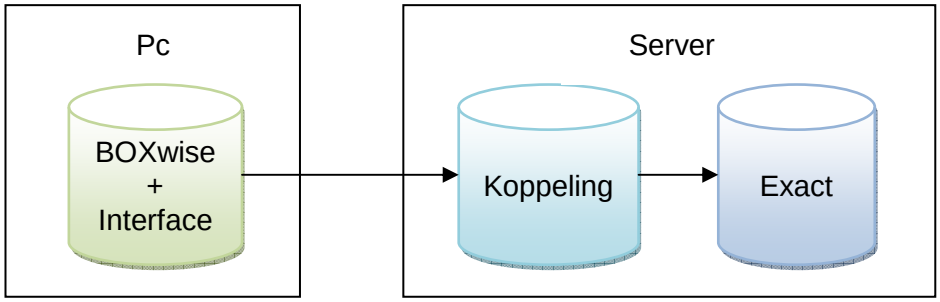
[4] Een webservice is een applicatie component dat toegankelijk is via de standaard webprotocollen (http://nl.wikipedia.org/wiki/Webservice)

Scenario 4: server, interface en koppeling als webservice
Wederom hetzelfde idee als het vorige punt, maar de interface wordt ook als een webservice beschikbaar gesteld op een server. Hierdoor is dit ook meteen weer beschikbaar voor elke BOXwise cliënt binnen een bedrijf.



Afbeelding 12: Engine en koppeling als webservice

Scenario 5: server, koppeling als bestand op server
Het was ook mogelijk om de koppeling op als een bestand op een server te zetten, dat vervolgens via een rechtstreekse verbinding wordt benaderd.



Afbeelding 13: Interne interface, externe koppeling op de server

Uiteindelijk is ervoor gekozen om in BOXwise zelf de interface te bouwen waarbij de koppeling als een apart bestand wordt aangeleverd. Voor het gemak was dit de beste oplossing. BOXwise kon op deze manier eenvoudig gebruik maken van de interface en koppeling.

5.3 Planning

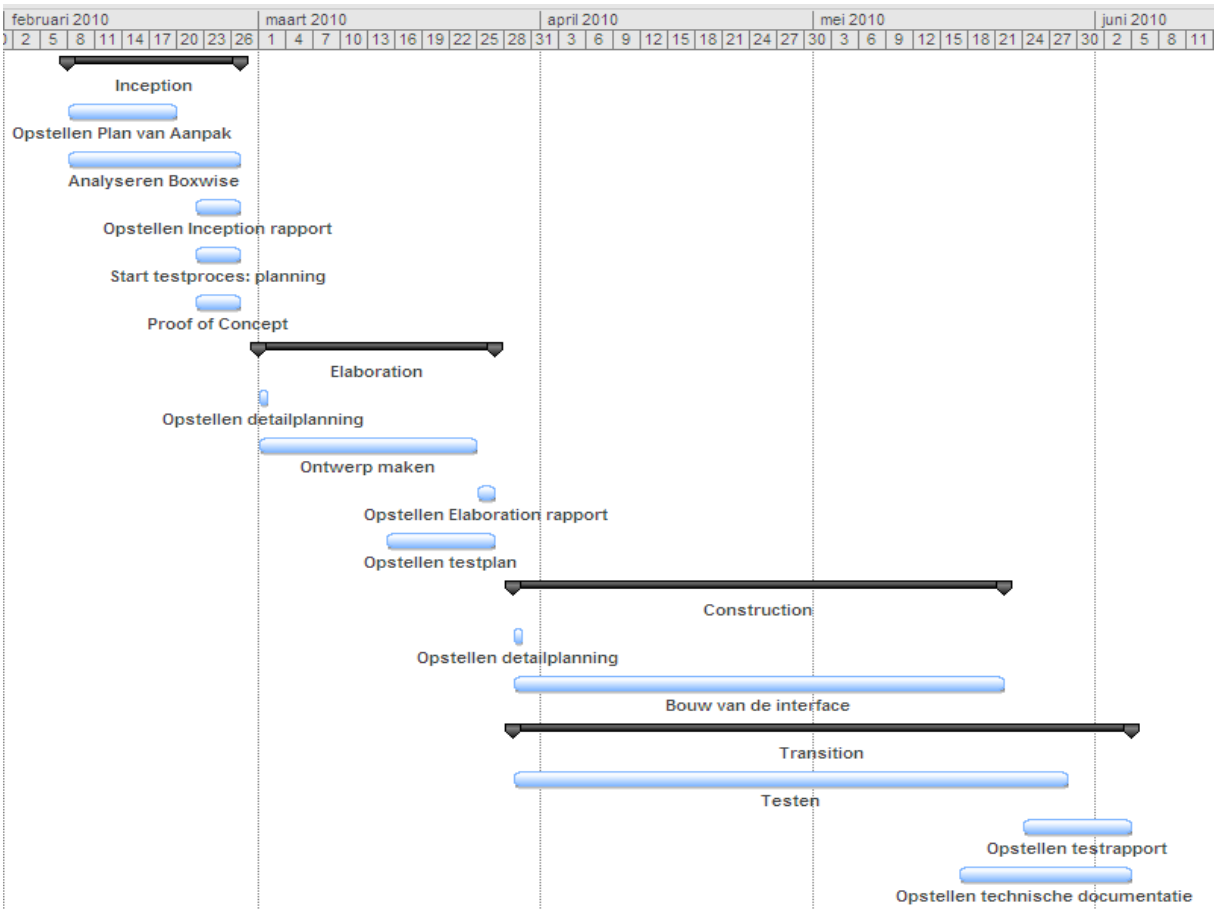
Aan de hand van het ontstane idee van de interface – koppeling structuur heb ik een planning voor het project opgesteld. Door het opstellen van een planning heb ik kunnen bepalen welke activiteiten nodig zijn om de opdracht uit te voeren. Door ook te definiëren hoeveel tijd ik voor de activiteiten nodig zou hebben kon ik inschatten of ik alles op tijd af zou krijgen.

Het gehele project heb ik opgesplitst in twee deelprojecten:

1. **Ontwikkelen van de interface**
   Dit vormt de basis voor een generieke communicatie met andere ERP systemen.

2. **Lostrekken van de huidige Exact Globe koppeling in BOXwise, en hiervan een aparte koppeling bouwen.**
   De koppeling bevat alle informatie om met Exact Globe te communiceren. BOXwise kan deze koppeling gebruiken om gegevens op te vragen via de koppeling.

Er bestond namelijk een kans dat de opdracht niet geheel afgerond kon worden. Het lostrekken van de koppeling kon meer tijd in beslag nemen dan beschikbaar was, wat kon resulteren in een niet voltooid product. Daarnaast was het ook nog niet duidelijk hoeveel werk er zou gaan zitten in het ontwikkelen van de interface. Daarom is het opgesplitst in twee projecten waarbij er in het eerste project de interface wordt ontwikkeld. Als er tijd over was, dan kon ik stapsgewijs beginnen met het lostrekken van de koppeling. Hierdoor zou de overdracht eenvoudiger worden indien het product niet geheel voltooid was. De eerste planning zag er als volgt uit:



Afbeelding 14: Eerste planning

Het leek mij verstandig om de Inception fase ruim te plannen zodat ik een goede indruk kreeg van BOXwise en de werkzaamheden die ik zou gaan uitvoeren. Ik moest niet alleen leren hoe BOXwise werkt, maar ook dat in relatie stond tot mijn opdracht. Door mijn aanpassingen zou BOXwise met andere ERP systemen kunnen werken. Daarom moest ik goed in kaart brengen hoe BOXwise werkt en uitzoeken waar ik aanpassingen moest maken. Daarbij leek het mij slim om de eisen in kaart te brengen zodat ik wist waaraan de aanpassingen moesten voldoen. BOXwise is een product dat wordt gebruikt door klanten, ik kon niet zomaar aanpassingen aan gaan maken. Vervolgens wilde ik een Proof of Concept maken om te illustreren hoe BOXwise op een generieke manier kon gaan werken met andere ERP systemen. Hiervoor zou ik ook gaan zoeken naar mogelijke ideeën. Om anderen op de hoogte te houden van mijn voortgang heb ik besloten om een Inception rapport op te leveren. Hierin zou ik de voorgaande activiteiten verwerken.

Vervolgens heb ik de Elaboration fase gepland zodat ik me kon richten op het ontwerp. Ik ben altijd al een voorstander geweest van het maken van een ontwerp. Hierdoor zou ik kunnen bepalen of aan alle eisen waren voldaan en hoe BOXwise zou gaan werken met de interface. Ik heb ook het ontwerp kunnen laten vallen door meteen te gaan programmeren. Dit leek mij echter niet verstandig gezien het feit dat ik werk op een bestaand product. Er is door andere programmeurs namelijk al bedacht hoe BOXwise werkt. Ik kon dus niet zomaar de broncode gaan wijzigen.
Uiteindelijk zou deze fase eindigen met een ontwerp. Aangezien BOXwise wordt gebruikt door klanten en er regelmatig aanpassingen worden gedaan, leek het mijn verstandig om het ontwerp op te leveren door middel van een Elaboration rapport. Hierin kon ik het ontwerp toelichten zodat het voor de programmeurs binnen TranCon duidelijk werd wat mijn aanpassingen waren en wat het precies doet.

Door het opleveren van een Elaboration rapport had ik voor mijzelf helder welke aanpassingen ik zou gaan maken. Aan de hand daarvan kon ik beginnen met de Construction fase. In deze fase wilde ik me puur bezig houden met het vertalen van het ontwerp in broncode: het programmeren.

Ik vond het belangrijk dat BOXwise goed zou worden getest. Zo kon ik controleren of ik mijn werk goed had gedaan en BOXwise goed functioneert met mijn aanpassingen. Daarom wilde ik eerst een testplan opstellen. Zo kon ik bepalen wat ik precies zou gaan testen en hoe ik dat zou gaan doen. Vervolgens zou ik de tests tegelijk met het ontwikkelen van de software gaan uitvoeren. Hiervoor heb ik gekozen zodat ik direct eventuele fouten kon opsporen. Dan kon ik in een vroeg stadium de fouten corrigeren en voorkwam ik dat de software aan het einde van de ontwikkeling niet goed functioneerde.

5.4 Eisen

Na het opstellen van het Plan van Aanpak bracht ik de eisen in kaart. Hierdoor kon ik vaststellen wat er mogelijk moest zijn met de interface en waar ik rekening mee moest houden.

Om de eisen in kaart te brengen heb ik eerst BOXwise verkend door de werking te bestuderen. Ik noteerde eventuele punten die belangrijk waren voor de aanpassingen aan BOXwise. Daarnaast keek ik ook naar de omgeving waarin BOXwise werkt. Hoewel de eisen puur functioneel waren moest ik ook rekening houden met externe factoren zoals performance en beheer. Vervolgens heb ik in de vorm van een overleg gevraagd wat men precies verwachtte van mijn uitbreiding. Daarbij heb ik de eisen in een lijst samengevat. Hierdoor kon het in één oogopslag worden bekeken. Uiteindelijk kreeg ik de volgende punten:

1. Generiek	**1**
Het moet dermate generiek zijn opgezet dat het automatisch: - Nieuwe koppelingen kan detecteren - Een standaard koppeling kan gebruiken - Instellingen kan uitlezen (zie eis 2) - Standaard aanroep	

2. Instellingen voor koppelingen	**1**
De instellingen moeten zodanig worden opgeslagen dat deze eenvoudig en op een generieke wijze uit te lezen zijn. De instellingen zullen o.a. bestaan uit: - Logo van het desbetreffende ERP systeem - Functionaliteiten die BOXwise met het ERP systeem kan uitvoeren	

3. Zelftest	**3**
Als de juiste koppeling met de instellingen is gekozen en geïnitialiseerd, moet er door middel van een zelftest worden gecontroleerd of er daadwerkelijk een connectie met het desbetreffende ERP systeem kan worden opgezet.	

4. Performance	**2**
De performance moet niet significant achteruit gaan door de implementatie van de nieuwe koppeling.	

5. Updates	**3**
Eenvoudige manier om updates uit te voeren	

Door middel van prioriteiten kon ik bepalen aan welke eisen ik me eerst op moest richten. Hierbij geldt dat 1 de hoogste prioriteit heeft en 3 de laagste. Alle punten waren echter van belang, maar ik heb ervoor gekozen om er eerst voor te zorgen dat BOXwise op een generieke manier kan communiceren met ERP systemen. Daarbij was het ook van belang dat BOXwise te weten kreeg wat er mogelijk is met een ERP systeem. Het achterliggende ERP systeem was altijd Exact Globe 2003 geweest, maar het is mogelijk dat niet alle functionaliteiten die van Exact gebruik worden, ook met andere ERP systemen mogelijk zijn.

Performance was vooral een punt dat ik in mijn achterhoofd moest houden. Alle gegevens worden namelijk real-time uit Exact of verwerkt waarbij de gebruiker niet wilt wachten. Daarom moest ik er rekening mee houden dat ik de interface en koppelingen op een efficiënte manier zou gaan ontwikkelen.

De zelftest en updates waren een extraatje. Als BOXwise op een eenvoudige manier kon worden geüpdatet met nieuwe koppelingen en kon worden getest, dan zou dit alleen maar schelen in het beheer.

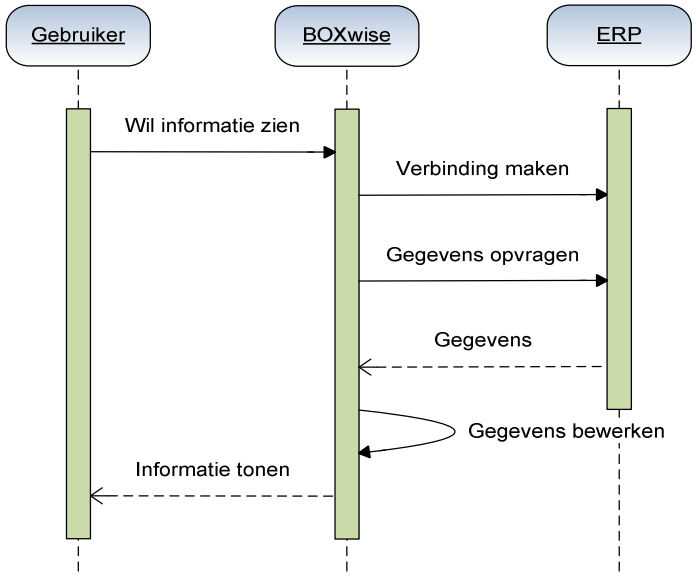
5.5 Informatiestromen

De architectuur van BOXwise zou een aanpassing krijgen. Maar wat voor impact zou dat op de broncode hebben? BOXwise moest met verschillende ERP systemen kunnen communiceren. Dit zou resulteren in een generieke methode om de communicatie met verschillende ERP systemen mogelijk te maken.

Tijdens het verkennen van BOXwise heb ik vooral gekeken wanneer er informatie uit Exact wordt opgevraagd. Die informatie zou namelijk in de toekomstige situatie uit andere ERP systemen worden opgevraagd. Door te bepalen wanneer er informatie wordt opgevraagd kon ik bepalen welke aanpassingen ik moest doen. Daarnaast moest ik ook bepalen welke informatie er tussen BOXwise en Exact wordt uitgewisseld, en wat BOXwise er precies mee doet. Dit proces heeft binnen in het project de naam *informatiestroom*[5] gekregen.

Informatiestromen van BOXwise
BOXwise heeft twee soorten informatiestromen: één om gegevens te ontvangen en één om gegevens te verzenden. In de oude situatie vinden deze informatiestromen plaats tussen BOXwise en Exact.

Voor het ontvangen van gegevens maakt BOXwise verbinding met de database van Exact. In deze database staan alle gegevens die worden gebruikt door Exact. Als de verbinding is opgezet, vraagt BOXwise om bepaalde gegevens, en krijgt deze als ze aanwezig zijn. Tot slot maakt BOXwise deze gegevens geschikt om ze aan de gebruiker weer te geven. Er is hier sprake van een cyclus:

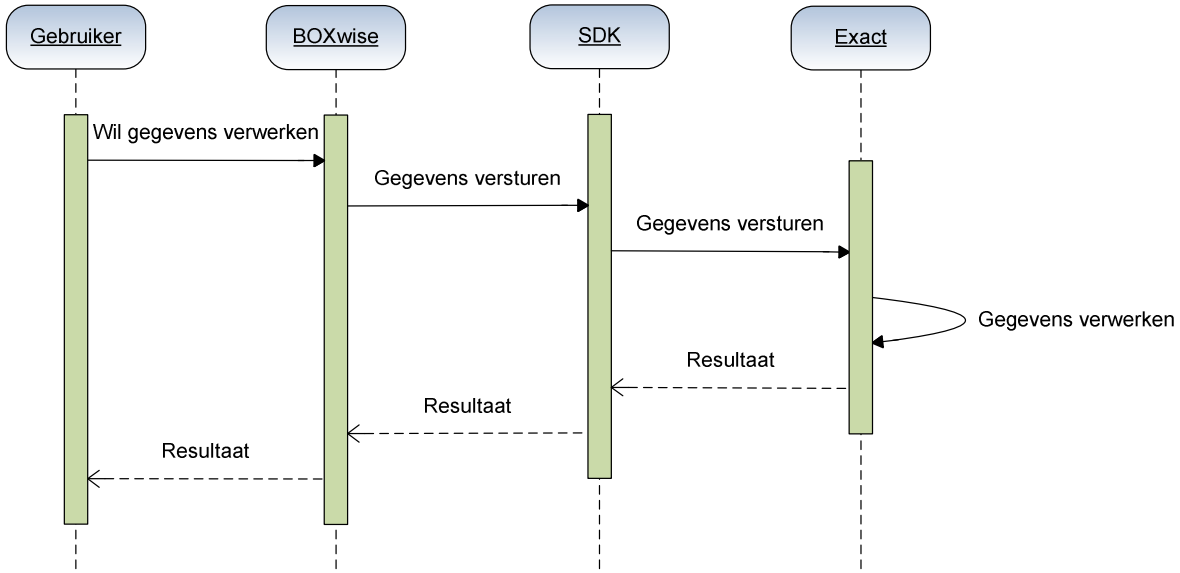


Afbeelding 15: Informatiestroom (gegevens opvragen)

[5] Een informatiestroom is het uitwisselen van gegevens tussen BOXwise en een ERP systeem.

Deze cyclus herhaalt zich meerdere keren, iedere keer als een gebruiker bepaalde informatie wil zien. Uiteindelijk is dit een informatiestroom. Dit is ook te herleiden uit de beschreven functionaliteiten in hoofdstuk 2.3: overal waar informatie wordt getoond, vindt er een bepaalde informatiestroom plaats.

Er is echter nog een informatiestroom waarbij BOXwise gegevens naar een ERP systeem stuurt. Hiervan is sprake als er gegevens moeten worden verwerkt, zoals het bijwerken van de voorraad of het afronden van een order. Hierbij wordt er gebruik gemaakt van een Exact SDK: een Software Development Kit. BOXwise verstuurt de gegevens die verwerkt moeten worden naar de SDK, en de SDK zorgt ervoor dat de gegevens op de juiste manier in Exact komen. De informatiestroom is als volgt:



Afbeelding 16: Informatiestroom (gegevens versturen)

De aanpassingen die ik heb gemaakt hebben ook een impact op de informatiestromen. BOXwise zou nog steeds informatie kunnen opvragen van een ERP systeem, maar dit moest plaats gaan vinden via een generieke methode. Door middel van de generieke methode zou BOXwise immers van diverse ERP systemen informatie kunnen opvragen.

Inventariseren van de informatiestromen

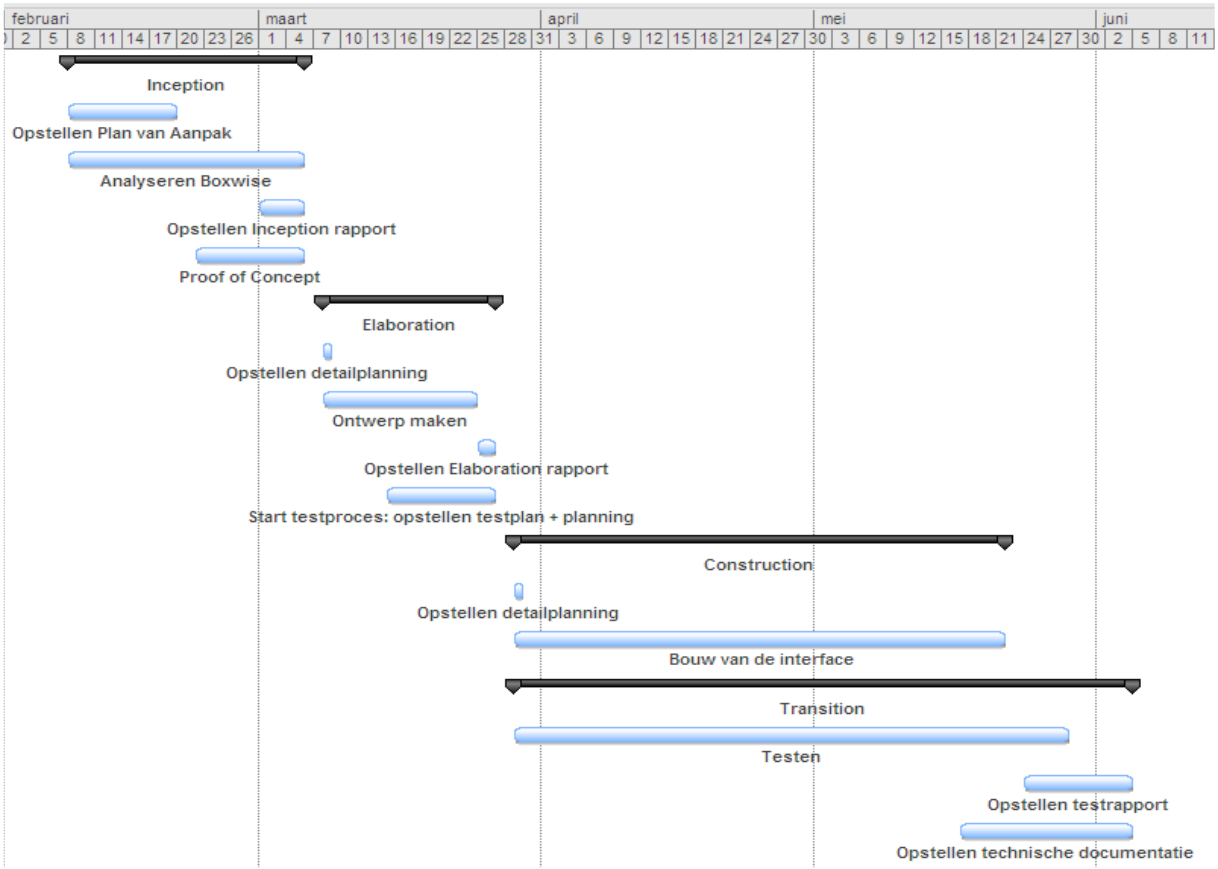
Om de informatiestromen weer te geven heb ik gebruik gemaakt van sequentiediagrammen. Met dit soort diagrammen is het mogelijk om de interactie tussen verschillende objecten in kaart te brengen. Met het gebruik van sequentiediagrammen wilde ik vast leggen wanneer een informatiestroom op gang kwam, en welke informatie werd uitgewisseld. Hiermee creëerde ik voor mijzelf een overzicht, en liet ik zien dat ik begreep hoe BOXwise in elkaar zat. Een bijkomend voordeel was dat een deel van BOXwise ook meteen gedocumenteerd was, iets wat nog niet het geval was.

Het inventariseren deed ik aan de hand van de functionaliteiten in BOXwise (ontvangsten, leveringen en inventarisatie). Elke functionaliteit heeft een aantal stappen (zoals beschreven in hoofdstuk 2.3). Bij elke stap heb ik in de broncode nagelopen op welke momenten er met Exact werd gecommuniceerd. Met andere woorden: ik heb gezocht naar informatiestromen. Bij elk gevonden informatiestroom heb ik een sequentie diagram opgesteld. In het diagram heb ik vastgesteld wanneer een informatiestroom op gang kwam, welke onderdelen van BOXwise daarbij waren betrokken en welke gegevens werden uitgewisseld. Het resultaat

hiervan werd toegevoegd aan een Inception rapport. Daarbij beschreef ik ook kort wat er precies gebeurt in de desbetreffende informatiestroom.

5.6 Onderbreking

Als gevolg van een sterfgeval in de familie heb ik de planning moeten aanpassen. De Inception fase heb ik verlengd met een week zodat ik alle informatiestromen goed kon inventariseren. De Elaboration fase, waarin ik het ontwerp op heb gesteld, heb ik toen verkort met een week. Door het werk dat ik op dat moment al had verricht was het voor mij al deels helder hoe het ontwerp er uit zou komen te zien. Daarom was ik er redelijk zeker van dat het inkorten van de Elaboration fase geen problemen zou opleveren.



Afbeelding 17: Gewijzigde planning

5.7 Proof of Concept

Naar aanleiding van de ontstane ideeën heb ik ook een Proof of Concept gebouwd. Hiermee kon ik aantonen hoe BOXwise op een generieke manier met ERP systemen zou kunnen communiceren.

Door gebruik te maken van online naslagwerken heb ik kunnen uitzoeken hoe BOXwise kon werken met de interface en koppelingen. Het was echter nog te vroeg om al een koppeling voor Exact te ontwikkelen, dus heb ik zelf een kleine database opgezet om gegevens uit op te halen. Door vooral in het begin veel werking te bestuderen door middel van debuggen heb ik BOXwise via een interface laten werken met een koppeling. Ook heb ik vooral geprogrammeerd door middel van *trial and error*. Door verschillende mogelijkheden uit te proberen kon ik zien hoe alles precies werkt en wat er mogelijk is.

Uiteindelijk liet ik het PoC door middel van een korte demonstratie zien. Daarna is er besloten om het idee verder uit te werken.

5.8 Rapportage

Het afronden van de eerste fase werd afgerond met het Inception rapport. Het rapport was de uitkomst van de werkzaamheden die in de eerste weken waren gedaan. Het doel hiervan was om de betrokkenen op de hoogte te houden van mijn werkzaamheden, en te laten zien wat de mogelijke ideeën waren.

Voor inzage is het Inception rapport toegevoegd in Bijlage B.

6. Deelproject: Interface

Het eerste deelproject: de interface. Hiervoor waren alle informatiestromen geïnventariseerd, en waren de mogelijk ideeën in kaart gebracht. Van die ideeën was er één uitgekozen om verder uit te werken.
Om ervoor te zorgen dat BOXwise op een generieke manier een verbinding kon maken met een ERP systeem, moest de interface eerst worden gebouwd. Met deze interface zou het mogelijk worden om aan te geven met welk ERP systeem er gecommuniceerd moest worden. In dit hoofdstuk zal er worden ingegaan op de werkzaamheden die zijn uitgevoerd en welke keuzes ik heb gemaakt.

6.1 Het ontwerp

Ik ben begonnen met het maken van een ontwerp voor de software. Tijdens het maken van de Proof of Concept was er al een idee ontstaan, maar nog niet volledig uitgewerkt. Daarom werkte ik eerst het ontwerp verder uit voordat ik daadwerkelijk de software zou gaan bouwen. Op deze manier kon ik bepalen hoe de interface er uit zo komen te zien en hoe BOXwise daar gebruik van gaat maken. Hierdoor zou ik efficiënter de interface kunnen bouwen en heb ik kunnen controleren of aan alle eisen was voldaan.

Een belangrijk aspect in het ontwerp is het uitwisselen van gegevens tussen BOXwise en de koppeling met het ERP systeem. De gegevens namelijk worden real-time uitgewisseld, en de gebruikers willen zo min mogelijk hinder ondervinden. Elke informatiestroom haalt zijn eigen set aan gegevens op of verstuurt een bepaalde set aan gegevens. Vaak wordt er een kleine set opgehaald of verstuurd, maar soms ook een grote hoeveelheid. Dit laatste kan performance problemen geven. Bij het ontwerp moest ik daar rekening mee houden.

Uitwisselen van informatie
Bij het ontwerp moest er rekening mee houden dat de gegevens op een zo efficiënt mogelijke manier worden verstuurd. Het *.Net framework*[6] biedt verschillende oplossingen om tijdelijk gegevens op te slaan. Door gebruik te maken van mijn eigen ervaringen, het internet en advies van mijn begeleider zocht ik naar de meest geschikte oplossing. Uiteindelijk kwam ik tot de conclusie dat de beste oplossing het gebruik van eigen *objecten*[7] was. Een object kan worden gezien als een verzameling van operaties binnen een programma. Deze objecten kunnen ook worden gezien als informatiedragers: het zorgt er voor dat de juiste informatie naar de juiste plek worden verstuurd. Door het gebruik van eigen objecten kon ik zelf bepalen wat er in het object werd opgeslagen zonder dat er een overhead aan extra functies bij kwam. Het .Net framework heeft vele andere oplossingen, maar het nadeel daarvan is dat die over extra functionaliteiten beschikken. Iets wat voor mijn situatie teveel van het goede was. Een bijkomend voordeel was dat alle gegevens “Strongly Typed” zijn geworden. Hierdoor kan er bij de ontwikkelaars geen twijfel ontstaan over de inhoud en de types.

Het uitwisselen van gegevens was niet het enige wat uitgezocht moest worden. Ik heb ook uitgezocht welke gegevens uitgewisseld moesten worden. Dit heb ik gedaan door in de databaseverbinding op te zoeken welke gegevens in een informatiestroom werden opgevraagd. Hierin stond welke gegevens werden opgehaald. Vervolgens controleerde ik ook in BOXwise welke gegevens daadwerkelijk werden gebruikt. Aan de hand daarvan heb ik bepaald welke gegevens in het object zouden worden opgeslagen. Dit liet ik ook controleren zodat ik zeker wist dat ik niets over het hoofd zag.

[6] Het .Net is een applciatieframework ten behoeve van de naadloze samenwerking van applicaties en bibliotheken in verschillende programmeertalen (http://nl.wikipedia.org/wiki/.NET_framework).
[7] Bron: http://nl.wikipedia.org/wiki/Object_%28informatica%29

Om de objecten te verwerken in het ontwerp maakte ik gebruik van de klassendiagrammen. Hierin legde ik vast welke gegevens in een object werden opgeslagen.

Koppelingen
Er was al besloten om een koppeling als apart bestand aan te leveren waarna BOXwise het kan gebruik om informatie op te vragen / te versturen. Daarom moest ik ook op zoek gaan naar een manier zodat BOXwise deze kan gebruiken. Er kunnen dan wel koppelingen worden gebouwd, maar BOXwise moet deze wel kunnen lokaliseren en valideren.

Ik werd er op gewezen dat ik gebruik kon maken van het configuratie bestand. Dit bestand wordt voor BOXwise gebruikt om bepaalde instellingen in op te slaan. Het is een XML bestand, wat inhoudt dat er eenvoudig en gestructureerd gegevens kunnen worden toegevoegd. Door in het configuratie bestand gegevens over de koppelingen op te slaan, was het voor BOXwise en de ontwikkelaars duidelijk waar de instellingen worden bewaard. Daarom heb ik hier ook gebruik van gemaakt door het configuratie bestand uit te breiden met een extra element ErpProvider. Een voorbeeld staat hier naast. In dat element kunnen vervolgens meerdere koppelingen worden geregistreerd. Hierdoor kan BOXwise flexibel een koppeling selecteren. Omdat BOXwise maar met één ERP systeem tegelijk kan werken (en dus ook met één koppeling), heb ik een default tag toegevoegd. Hierdoor kan eenvoudig in hetzelfde bestand worden aangegeven met welk ERP systeem BOXwise moet werken.

```
<ErpProvider default="">
        <ErpConnections>
                <add name="Exact" type="" />
        </ErpConnections>
</ ErpProvider >
```

Om uit te zoeken hoe ik BOXwise gebruik kon laten maken van het configuratie bestand, heb ik gebruik gemaakt van MSDN. MSDN biedt een naslagwerk aan van het .Net framework. Daar heb ik kunnen opzoeken hoe ik met configuratiebestanden kon werken. Daarnaast keek ik in BOXwise zelf hoe het met het configuratie bestand om gaat.

6.2 Provider model

Tijdens het opstellen van het ontwerp werd er mij verteld dat het handig kon zijn om te kijken naar de *Patterns & Practices*[8] van Microsoft. Het is opgezet door Microsoft om oplossingen aan te bieden voor software ontwikkelaars, gebaseerd op best practices in de praktijk. Het bevat artikelen en video's om bepaalde onderwerpen toe te lichten, handleidingen en software modellen genaamd design patterns. Design patterns zijn generieke software oplossingen die gebruikt kunnen worden bij het ontwikkelen van software. Tijdens het bestuderen van de Patterns & Practices kwam het zogenaamde *Provider model* tegen. Het Provider model heeft deze naam omdat het zijn doel is om functionaliteiten aan te bieden (Engels: provider), gebruikmakend van o.a. een interface.

Deze pattern leek mij zeer geschikt voor BOXwise:
- De pattern sloot aan bij het idee dat was gebaseerd op de shipping layers (hoofdstuk 5.2)
- Door het opstellen van een contract is het mogelijk om voor een koppeling alle informatiestromen te definiëren.
- Het contract wordt in BOXwise opgesteld. Elke koppeling moet er gebruik maken. Hierdoor kan BOXwise op een generieke manier communiceren met de koppelingen.

[8] Patterns & Practices website: http://msdn.microsoft.com/en-us/library/ff649455.aspx

Aan de hand van het Provider model heb ik het ontwerp verder kunnen uitwerken. Voor de interface definieerde ik de informatiestromen. Deze definitie moet vervolgens in de koppeling worden overgenomen zodat BOXwise via de interface er gebruik kan maken. Hierbij legde ik ook per informatiestroom vast welke informatiedragers gebruikt moesten worden om bepaalde gegevens uit te wisselen.

6.3 Bouw

Het bouwen van de interface nam minder tijd in beslag dan ik aan het begin van het project rekening mee had gehouden. Tijdens het opstellen van het ontwerp zag ik dit al aankomen en heb ik de planning ook bijgesteld. Maar ook door het ontwerp zelf heb ik op een efficiënte wijze de interface kunnen bouwen. In het ontwerp heb ik gespecificeerd welke objecten gebruikt zouden worden, en hoe de interface eruit zou komen te zien. Hierdoor heb ik tijdens de bouw alleen maar het ontwerp hoeven te implementeren.

Tijdens de bouw heb ik de iteraties aangehouden zoals ik deze had gepland. Door op deze manier te werken heb ik ervoor kunnen zorgen dat een groep functionaliteiten was geïmplementeerd in de interface zonder dat ik iets vergat.

6.4 BOXwise fusie

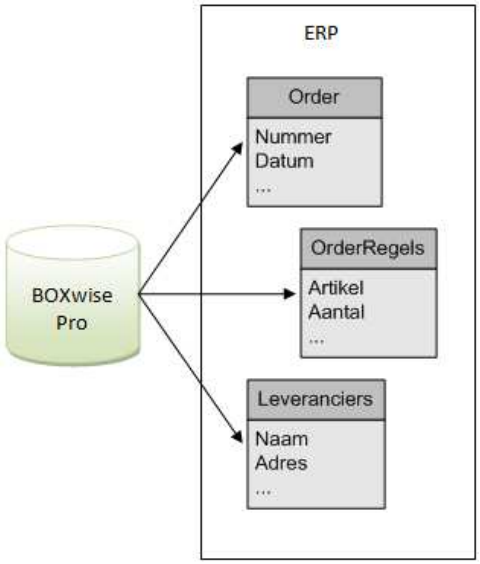
Tijdens het ontwikkelen van de interface kreeg ik te maken met een bericht die een aantal gevolgen had. Er was namelijk een strategische beslissing genomen om BOXwise Pick & Pack en BOXwise Pro te fuseren. Dit hield in dat de twee versies van BOXwise werden samengevoegd tot één versie, maar ook dat er bepaalde gevolgen waren voor mijn opdracht.

Het is niet zo dat de deadline voor de opdracht in gevaar is gekomen. De functionaliteiten van BOXwise P&P zouden worden toegevoegd aan BOXwise Pro waarna er één versie van BOXwise zou ontstaan. Voor mijn opdracht betekende dit dat ik alle informatiestromen moest controleren. Zo kon ik na gaan of de informatiestromen in BOXwise P&P ook bestonden in BOXwise Pro. Daarnaast had BOXwise Pro ook aantal extra informatiestromen die ik ook weer moest verwerken in mijn ontwerp. Dit resulteerde in uitgebreidere iteraties.

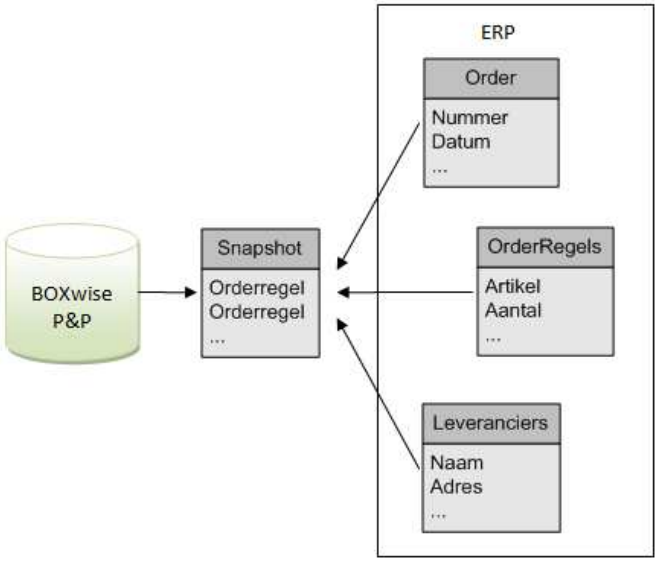
Maar voordat ik met Pro aan de slag ging, heb ik de implementatie van de interface in BOXwise P&P afgerond. Zo heb ik alle iteraties kunnen doorlopen en eventuele aandachtspunten genoteerd. Na het afronden van alle iteraties ben ik terug gegaan naar het ontwerp. Daarbij was ik weer begonnen bij de eerste iteratie. Alle iteraties werden weer doorlopen waarin ik alle informatiestromen nog een keer heb gecontroleerd. Ik controleerde of deze nog overeen kwamen, of dat ze in mijn ontwerp gewijzigd moesten worden. In sommige gevallen waren er ook een aantal nieuwe informatiestromen. Deze heb ik toegevoegd in het ontwerp. Ik heb dezelfde iteraties aangehouden, maar in feite controleerde of breidde ik deze uit.

Informatiedragers
Door de BOXwise fusie heb ik de informatiestromen nog een keer gecontroleerd om er zeker van te zijn dat deze nog correct waren in de nieuwe situatie. Waar ik voornamelijk op lette was hoe de gegevens tussen BOXwise en Exact worden uitgewisseld. In BOXwise Pro wordt er namelijk al gebruik gemaakt van zelf opgestelde objecten. Daarom heb ik de objecten die ik zelf heb opgesteld (beschreven in paragraaf 1) vergeleken met de objecten die aanwezig zijn in BOXwise Pro. Hierbij heb ik gekeken hoe ik deze kon samenvoegen, en stelde eventuele vragen over de gegevens die uitgewisseld moesten worden. Uiteindelijk kwam er per informatiestroom één object uit dat gebruikt zou worden voor de gegevensuitwisseling.

Snapshot
Het was echter niet alleen hier en daar wat aanpassingen doen, voor het verzenden van goederen was er een grotere aanpassing nodig. BOXwise Pro heeft niet de mogelijkheid om een zogenaamd *snapshot*[9] te maken. De verkooporders en alle relevante gegevens worden direct uit de ERP database gehaald (afbeelding x). Het was echter wel de bedoeling dat BOXwise Pro deze functionaliteit zou krijgen.



Afbeelding 18: BOXwise Pro zonder snapshot



Afbeelding 19: BOXwise P&P met snapshot

Bij het maken van een snapshot worden alle verkooporders die op dat moment open staan, opgehaald uit het ERP systeem. Vervolgens worden deze opgeslagen in een tabel in de eigen database van BOXwise. Vanaf dat moment worden alle verkooporders uit de eigen database opgehaald, totdat er een nieuw snapshot wordt gemaakt (afbeelding x). Op deze manier kan BOXwise op een intelligente manier de voorraad toewijzen aan orders en de magazijnmedewerkers alleen laten zien wat echt verstuurd kan worden.

Opslaan van gegevens voor de snapshot
Doordat BOXwise Pro nog geen gebruik kon maken van een snapshot, haalde het alle gegevens op met zijn eigen objecten (informatiedragers). Deze objecten zijn opgesteld voor specifieke gegevens zoals de gegevens van een klant of alle producten van een verkooporder.

Dit riep het bij mij de vraag op: waarom wordt de snapshot in één tabel opgeslagen? In de tabel die wordt gebruikt staan alle producten van alle verkooporders. Bij elk product zijn ook de gegevens van de desbetreffende klant en order toegevoegd. Er is dus sprake van *redundatie*[10]. Om dit te illustreren is er het volgende voorbeeld:

- De snapshot heeft twee orders: A en B. Beide orders zijn van klant TC Studio.
- Order A heeft 4 producten, order B heeft 7 producten

[9] Een snapshot is een momentopname van alle openstaande verkooporders.
[10] Redundantie betekent dat gegevens meerdere malen in een database zijn opgeslagen.

Dit betekent dat er 11 regels zijn waarbij elke regel informatie over de klant en order heeft:

OrderId	Klant	Datum	Product	Aantal	Prijs
31	*TC Studio*	*20-01-2010*	Camera 10Dh	2	€205,00
31	*TC Studio*	*20-01-2010*	Draagtas	12	€20,00
31	*TC Studio*	*20-01-2010*	Lens 38F	5	€35,00
Etc.					

Het leek mij netter om de gegevens te scheiden, waarbij alle relevante gegevens bij elkaar in een tabel staan. Dit proces wordt ook wel *normaliseren*[11] genoemd. Hierdoor kon de kans op fouten door redundantie worden voorkomen.

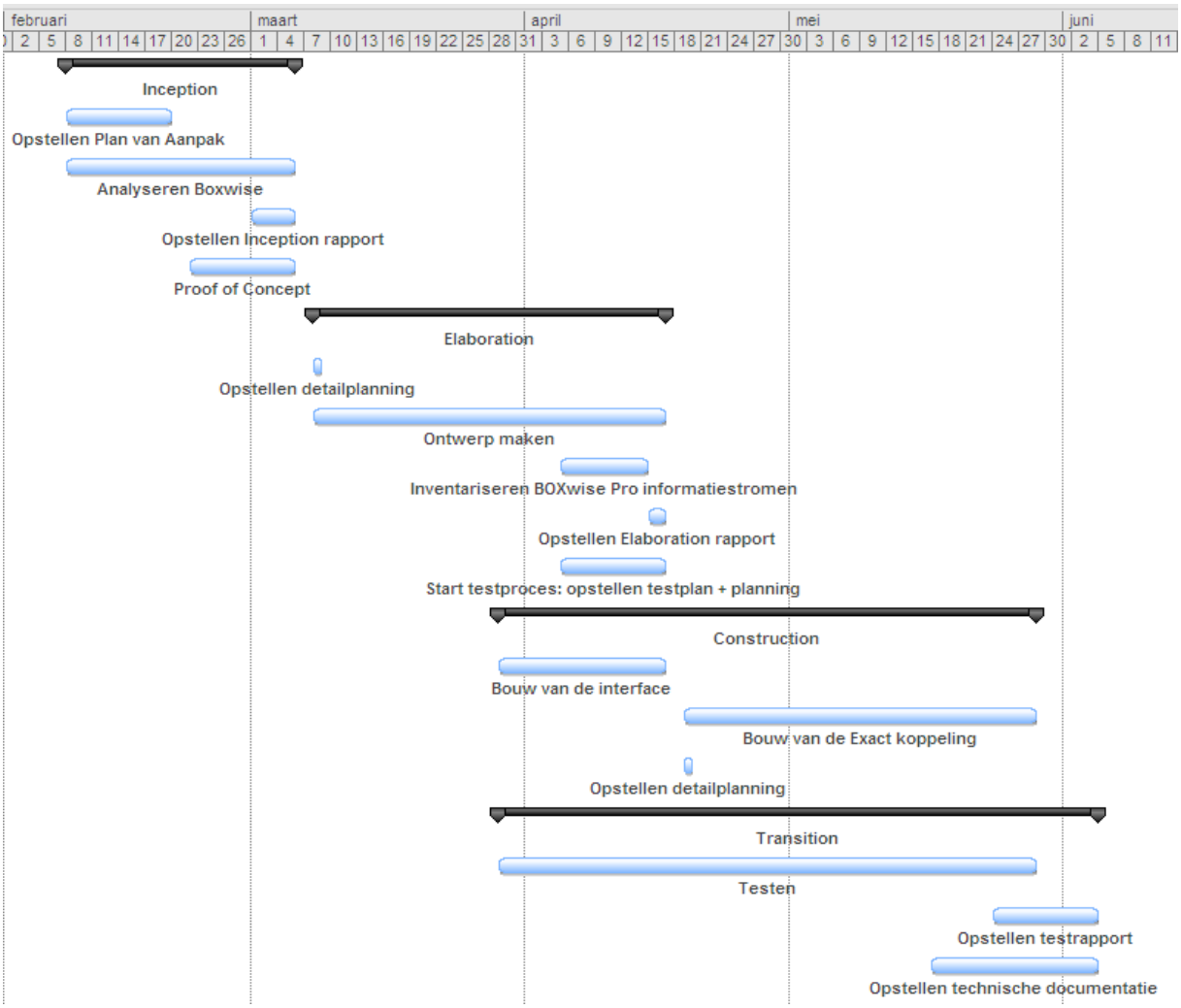
Ik heb dit voorgelegd, maar er is toen besloten om de bestaande snapshot functionaliteit te behouden, en deze in te bouwen in BOXwise Pro. De gebruikte tabel voor de snapshot wordt namelijk altijd eerst geleegd voordat een nieuw snapshot wordt gemaakt. Dit voorkomt automatisch de kans dat er fouten ontstaan door de overbodige data. Daarnaast bestond de snapshot functie al een geruime tijd in BOXwise P&P, en heeft het bewezen dat het werkt.

Verdere ontwikkeling
Na het afronden van het ontwerp voor de nieuwe situatie ging ik verder met het bouwen van het contract. Ik heb weer gebruik gemaakt van de iteraties om de bestaande code te controleren en uit te breiden.

6.5 Planning

Het werd steeds duidelijker dat het ontwikkelen van de interface veel minder tijd zou kosten dan gepland. Daarnaast kreeg ik ook te maken met de fusie tussen BOXwise P&P en Pro. Daarom heb ik de planning voor de interface verkort met zes weken. Deze keuze is gebaseerd op het werk dat ik al had gedaan. Hierdoor had een beter beeld gekregen van wat er nog moest gebeuren. Daardoor kon ik met een redelijke zekerheid bepalen hoeveel tijd ik nog nodig zou hebben voor de interface waarbij de nieuwe situatie in werd meegenomen. Hiervoor schatte ik ongeveer twee weken in. Daarna zou ik beginnen met het lostrekken van de Exact koppeling in BOXwise en hiervan een aparte koppeling bouwen.

[11] Normaliseren is het verwijderen van overbodige gegevens door alle samenhangende gegevens bij elkaar in één tabel toe te voegen.



Afbeelding 20: Gewijzigde planning

6.6 Testen

Door het gebruik van een interface heb ik geen tests opgesteld. Een interface definieert slechts een contract dat BOXwise de mogelijkheid geeft om met een koppeling te communiceren. Ik kon dus niet zomaar de werking van de interface testen aangezien het niets deed. Ik had wel de mogelijkheid om tests aan te maken met zogenaamde *mock objecten*[12]. Hiermee kon ik alvast testen of BOXwise goed gebruik maakt van de interface. Ik heb er echter voor gekozen om hiervan geen gebruik te maken en het testen uit te stellen totdat ik de koppeling zou gaan bouwen. Ik kon dan vervolgens testgevallen opstellen voor BOXwise waarbij de nieuwe informatiestromen werden getest. Er zou dan automatisch gebruik worden gemaakt van de interface.

[12] Een mock object is een object dat de werking van een implementatie simuleert.

6.7 Documentatie

Bij het maken van het ontwerp heb ik de bijbehorende documentatie geschreven. Dit bestond vooral uit het beschrijven van de informatiestromen, en hoe deze plaats zouden vinden als het provider model was geïmplementeerd. Dit resulteerde in een Elaboration rapport en was specifiek gemaakt voor TranCon. Met behulp van deze documentatie was het mogelijk om na te lezen welke informatiestromen er aanwezig zijn. Door de BOXwise fusie kwamen er echter extra informatiestromen bij. Omdat deze moesten worden toegevoegd aan de interface, heb ik eerst uitgezocht welke dat precies waren. De nieuwe informatiestromen documenteerde ik echter niet zo uitgebreid als de eerder gemaakte documentatie. Men had liever dat ik me eerste richtte op de bouw.

Voor de interface zelf werd er ook documentatie opgesteld. Deze documentatie was niet alleen voor TranCon zelf, maar zou ook beschikbaar worden gesteld voor externe ontwikkelaars. Met behulp van de documentatie legde ik uit hoe een koppeling door BOXwise zal worden gebruikt. Ik heb gebruik gemaakt van de mogelijkheid om in de broncode documentatie te schrijven. Vervolgens heb ik met het programma Sandcastle een HTML help bestand gegenereerd. Hierdoor heb ik dicht bij de broncode kunnen beschrijven hoe het werkt, en welke gegevens uitgewisseld worden. Een bijkomend voordeel was dat de gebruikte software om software te ontwikkelen, ook die documentatie kon inlezen. Daardoor kan de documentatie tijdens het ontwikkelen direct worden geraadpleegd.

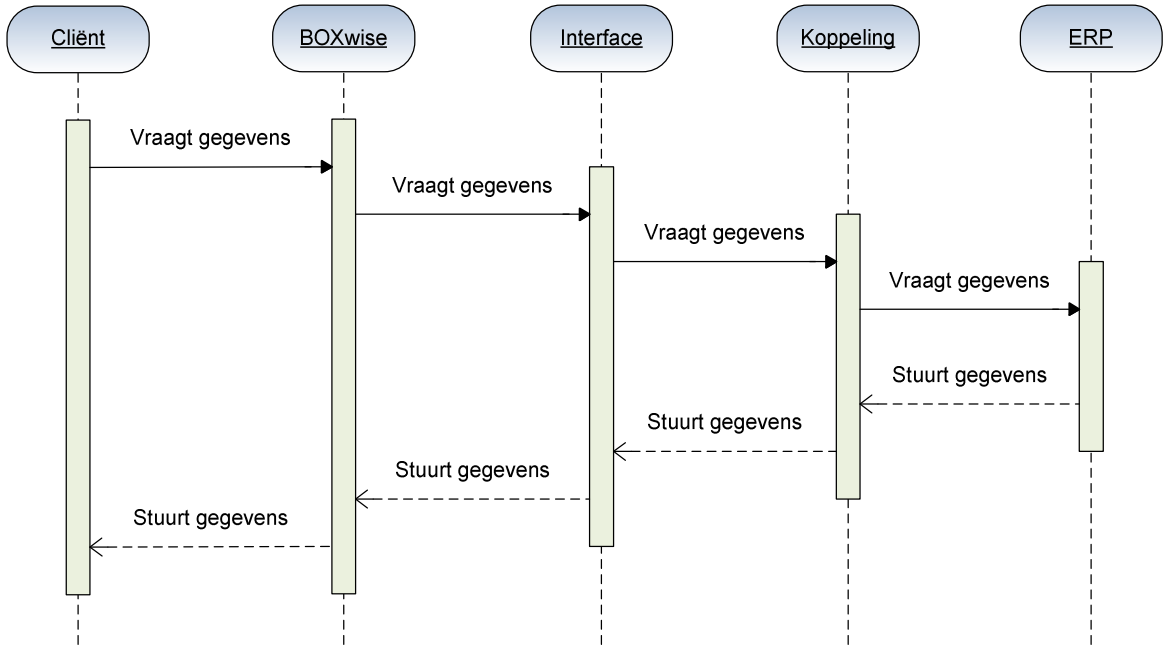
Voor inzage is het Elaboration rapport toegevoegd in Bijlage C. Het HTML help bestand staat op de CD dat is toegevoegd als Bijlage E.

7. Deelproject: Exact koppeling

De Exact koppeling zorgt voor het opzetten van de verbinding met de Exact database en het ophalen / versturen van gegevens. Door BOXwise hiervan gebruik te laten maken via de interface, kan het weer communiceren met Exact. In dit hoofdstuk zal de ontwikkeling worden toegelicht.

7.1 Ontwikkeling van de koppeling

In tegenstelling tot de ontwikkeling van de interface, heb ik voor de koppeling geen apart ontwerp opgesteld. De reden hiervoor was omdat de broncode al bestond, namelijk in BOXwise zelf. Daarnaast heb ik in het ontwerp van de interface ook al beschreven hoe de nieuwe informatiestromen zouden gaan verlopen. Daardoor was het voor mij al duidelijk wat de koppeling moest doen en hoe het zou gaan werken. In plaats van een ontwerp heb ik wel een stappenplan opgezet waarbij de onderstaande afbeelding te gebruiken is.



Afbeelding 21: Nieuwe situatie informatiestroom in stappen

In deze afbeelding wordt weergeven hoe een informatiestroom verloopt in de nieuwe situatie. In de nieuwe situatie maakt BOXwise gebruik van de interface en een koppeling om met een ERP systeem te kunnen communiceren. Hierbij worden de informatiedragers (hoofdstuk 6.4) gebruikt om gegevens uit te wisselen. Het stappenplan is daarom gericht op twee punten:

- Zorgen dat BOXwise correct gebruik gaat maken van de informatiedragers
- Zorgen dat de informatiestromen correct verlopen via de interface en koppeling.

Met die twee punten in gedachten heb ik de volgende stappen kunnen definiëren:

1. Aanroep van de cliënts updaten
   De cliënts roepen BOXwise aan om gegevens op te vragen of te versturen. Bij deze eerste stap moet de juiste informatiedrager worden opgesteld.

2. Aanroep naar de koppeling updaten
   Vervolgens vraagt / stuurt BOXwise de informatiedragers door via de interface aan / naar de koppeling.

Vervolgens in de koppeling:

3. Zorgen dat de koppeling op de juiste manier de benodigde gegevens ontvangt (om gegevens uit het ERP te halen, of er naar toe te versturen)
4. Opzetten van de verbinding
5. Gegevens opvragen / versturen
6. Gegevens geschikt maken om terug te sturen naar BOXwise
7. Gegevens terug sturen naar BOXwise
8. Verbinding sluiten

Door deze stappen te volgen heb ik de bestaande informatiestromen kunnen wijzigen. Hierbij heb ik weer gebruik gemaakt van de bestaande iteraties (ontvangsten, leveringen, inventarisatie). Na het wijzigen van een informatiestroom heb ik meteen de bijbehorende tests opgesteld. Zo kon ik direct de werking testen.

7.2 Testen

In tegenstelling tot de interface kon de koppeling wel uitvoerig worden getest. Het doel van het testen was om te garanderen dat de software goed werkt. Door gebruik te maken van TMap heb ik op een gestructureerde manier een testproces kunnen opzetten. Dit proces begon met een testplan. Met het testplan (toegevoegd als Bijlage D) kon ik laten zien wat ik precies wilde testen, en kon ik voor mijzelf ook bedenken hoe ik het wilde aanpakken. In het plan stelde ik een planning op die ik tegelijk met de ontwikkeling van de koppeling heb laten lopen. Door meteen tests op te stellen voor de nieuwe broncode konden mogelijk fouten direct naar voren komen. Zo zou ik achteraf niet voor onverwachte verrassingen komen te staan. Bij het opstellen van de planning bekeek ik wat er getest moest worden en welke testtechnieken daarbij het meest geschikt waren. Ik wilde in ieder geval testen of de informatiestromen goed worden verwerkt met de interface en koppeling, dus dat heb ik als uitgangspunt genomen.

Unit testing
Ik heb de software getest met behulp van zogenaamde *unit tests*. Dit zijn tests die geautomatiseerd kunnen worden uitgevoerd. Dit heeft als doel om de werking van een programma op een herhaalbare wijze te testen. Een unit test wordt eenmalig geschreven, waarna het op elk willekeurig moment kan worden uitgevoerd. Mocht er een wijziging zijn gemaakt aan de software, dan kan er letterlijk door één druk op de knop worden gecontroleerd of de software nog correct functioneert.

Binnen TranCon wordt al intensief gebruik van gemaakt van unit testing. Hierdoor kwam ik op het idee om daar ook gebruik van te maken. Ik kon het namelijk perfect gebruik om de werking van BOXwise te testen. Door bepaalde functionaliteiten aan te roepen wordt er automatisch gebruik gemaakt van de informatiestromen. Zo kon ik testen of BOXwise op een juiste manier gebruik maakt van de interface en koppeling.

TMap heeft een testtechniek beschreven die goed past bij Unit tests, genaamd *semantische tests*. Deze tests hebben als doel om de relatie tussen gegevens te controleren. In het geval van BOXwise is dat de relatie tussen een vraag en antwoord: BOXwise vraagt om specifieke data, en krijgt vervolgens data terug als antwoord.

Voor de semantische tests heeft TMap een aantal stappen gedefinieerd. Dit begon met het vaststellen van de testsituaties. Tijdens deze stap wordt bepaald wat er wordt getest. Ik was geïnteresseerd in de werking van BOXwise met de interface en koppeling waarbij eventuele fouten goed worden opgevangen. Dit is van belang omdat gebruikers niet onnodig moeten worden lastig gevallen met onduidelijke meldingen. Hierbij kon ik een aantal scenario's schetsen:

- BOXwise vraagt om gegevens, en krijgt deze vervolgens van de koppeling
- BOXwise vraagt om gegevens, maar krijgt deze niet omdat de gegevens niet aanwezig zijn.
- BOXwise vraagt om gegevens, maar levert verkeerde data aan. Hierdoor treedt er een fout op.

De volgende stap bestond uit het bepalen van de logische testgevallen, ofwel de mogelijke situaties die zich konden voordoen. Hierbij kon ik de informatiedragers gebruiken. Omdat er slechts een beperkte hoeveelheid gegevens worden opgeslagen in de informatiedragers, heb ik in vele gevallen alle situaties kunnen testen. Bij de informatiedragers die groter waren nam ik slechts een subset om te testen. Hierbij heb ik bekeken welke gegevens BOXwise echt nodig heeft om goed te functioneren.

Alle testsituaties heb ik samengevoegd in het testplan. Zo kon ik anderen laten zien wat ik precies ging testen.

De derde stap was het definiëren van de fysieke testgevallen, ofwel de waardes van de testsituaties. Deze heb ik pas opgezocht tijdens het bouwen van de koppeling. Zo kon ik geconcentreerd blijven op één informatiestroom tegelijk. De juiste testwaardes haalde ik rechtstreeks op uit Exact. De onjuiste testwaardes creëerde ik door de juiste testwaardes zodanig te bewerken dat deze niet meer het gewenste resultaat gaven.

Vervolgens bouwde ik de unit tests aan de hand van de testgevallen. Die kon ik ook meteen uitvoeren om te controleren of de code goed werkt. Af en toe mislukte een test omdat er toch een fout bleek te zijn in de broncode. Dit was alleen maar goed, zo wist ik direct dat er iets niet klopte. Voor elke test kon ik het resultaat bekijken waarbij ook de fout wordt weergeven. Zo kon ik in de broncode van BOXwise door middel van een debugger controleren wat er precies fout is gegaan en de fout corrigeren. Uiteindelijk heb ik voor elke informatiestroom een aantal unit tests kunnen opstellen. Bij eventuele toekomstige wijzigingen kunnen de unit tests eenvoudig opnieuw worden uitgevoerd om te controleren of alles nog goed functioneert.

Andere tests
Gezien de geringe tijd die nog beschikbaar was is er besloten om het bij unit tests te laten. Deze bieden namelijk al een goede basis.

7.3 Documentatie

Voor de koppeling zelf heb ik een document opgesteld waarin ik de werking toelichtte. Zo is het mogelijk voor anderen om eenvoudig te bekijken hoe het precies werkt. Hierbij heb ik gebruik gemaakt van sequentie diagrammen en toelichtende teksten.

Omdat ik gebruik heb gemaakt van unit testing, werden de resultaten altijd automatisch gegenereerd. Doordat die resultaten eenvoudig opnieuw kunnen worden gegenereerd heb ik geen rapport opgesteld met alle resultaten. Door alle tests op tijd af te ronden waren er ook geen aandachtspunten die nog gerapporteerd moesten worden.

8. Evaluatie

In dit hoofdstuk zal ik terug kijken op mijn afstudeerproject. Dit zal ik doen door eerst de ontwikkeling van de producten te evalueren. Daarna zal ik terug kijken op mijn aanpak en tot slot komen de competenties aan bod.

8.1 Producten

Interface
Over de interface ben ik tevreden. BOXwise kan er goed gebruik van maken en is nu een stuk flexibeler geworden.

Het ontwikkelen van de interface verliep soepel. Door een duidelijk ontwerp op te stellen heb ik BOXwise snel kunnen aanpassen zodat het met de interface werkt. Dat was allemaal gebaseerd op het Provider model. Hierbij kan ik mij afvragen of het niet slimmer was geweest als ik nog andere alternatieve oplossingen had gekeken. Na het ontdekken van het Provider model heb ik daar al vrij snel aan vast gehouden. Simpelweg omdat de pattern zeer geschikt was voor mijn project. Maar aan de andere kant is dit ook goed verlopen. Anders was ik misschien meer tijd kwijt voor het zoeken naar eventuele alternatieve oplossingen.

Exact koppeling
Over de Exact koppeling ben ik ook tevreden. Tijdens het ontwikkelen van de koppeling heb ik duidelijk gemerkt dat een goede basis belangrijk is. In dit geval was dat het eerste project, het ontwikkelen van de interface. Doordat de goede basis kon ik snel en gestructureerd de koppeling bouwen zonder dat de basis veel moest worden gewijzigd.

Het gebruik van een stappenplan in plaats van een ontwerp paste goed bij de ontwikkeling van de koppeling.

Ontwerp
Ik vind dat het vooraf maken van een ontwerp een goede stap was. Hierdoor kon ik snel de software bouwen doordat ik vooral al had bedacht hoe het zou gaan werken. Wel was het verstandiger geweest als ik vaker vragen had gesteld over de werking van BOXwise. Vaak zocht ik zelf op hoe iets werkte, maar het zou beter zijn geweest als iemand een uitleg gaf. Daardoor kreeg ik een beeld van het gehele proces en kon ik sneller mijn werk hervatten.

Daarnaast zou het beter zijn geweest als ik BOXwise Pro er vanaf het begin had bij betrokken. In eerste instantie werkte ik met BOXwise P&P, maar door BOXwise Pro erbij te betrekken kon ik ook eventuele ideeën op doen. Vooral bij het uitwisselen van informatie tussen BOXwise en Exact, waarbij ik kon leren van het gebruik van eigen objecten om informatie uit te wisselen. Dit is vooral een leermoment voor later: ik moet gebruik maken van wat er beschikbaar is. Zo hoef ik niet altijd het wiel opnieuw uit te vinden.

De intentie om ontwerpen uit te werken zal altijd blijven. Ik blijf het een prettige manier vinden om na te denken over de werking van een programma waarbij ik ook goed kan nalopen of aan alle eisen zijn voldaan. Echter, ik heb gemerkt dat het niet altijd heel diep hoeft te worden uitgewerkt. Zoals bij het ontwikkelen van de koppeling, daar was het definiëren van stappen al voldoende. Het maken van (gedetailleerde) ontwerpen zal dus altijd een afweging blijven, waarbij factoren zoals kennis, tijd en beslissingen van leidinggevenden altijd een rol zullen spelen.

Testen
Het testen van de software was iets wat ik serieus moest nemen. Uiteindelijk zou mijn werk in productie worden genomen. Het opstellen van een testplan was een stap in de goede richting, maar ik had duidelijker moeten vermelden wat ik zou gaan testen. Zo zou het voor andere personen ook helder zijn wat ik precies zou gaan testen en hoe ik het zou gaan uitvoeren.

Het testplan stelde ik ook pas op toen de koppeling moest worden gebouwd. Achteraf gezien zou het beter zijn geweest als het testplan werd opgesteld tijdens de ontwikkeling van de interface. Toen leek het me nog te vroeg om er aan te beginnen aangezien er nog niets zinnings kon worden getest. De interface was immers alleen een contract voor de koppeling. Maar tijdens het opstellen van de testgevallen voor de koppeling realiseerde ik me dat het wel nuttiger was geweest om eerder te beginnen met het testplan. De tests konden weliswaar niet worden uitgevoerd, maar ik kon wel alvast de testgevallen opstellen. Hierdoor kon ik al kritisch bekijken of de interface compleet was, en of de informatiedragers werden gebruikt.

Voor het testen zelf heb ik gebruik gemaakt van Unit tests. Deze heb ik gebruikt voor elke informatiestroom zodat ik zeker wist dat de juiste gegevens werden opgehaald. De testgevallen stelde ik op voordat ik met de bouw begon, maar de werkelijke testwaardes zocht ik pas op als de tests gemaakt moesten worden. Zo kon ik me blijven concentreren op één informatiestroom. Het testen hielp mij ook om na te denken of de argumenten wel klopten, of dat de functionaliteiten wel noodzakelijk waren.
Achteraf gezien pakte deze manier van werken wel goed uit, maar het zou beter zijn geweest als ik tijdens het opstellen van de testgevallen meteen de waardes had opgezocht. Hierdoor zou de gehele voorbereiding eerder plaats hebben gevonden, en dacht ik eerder na of een bepaalde methode/argument wel logisch is. Hierdoor kon ik onduidelijkheden eerder opsporen, en waren er minder correcties nodig.

Het testen op zich vond ik zeker nuttig voor de gemaakte software. Hierdoor kon ik garanderen dat het goed werkte, en kon ik kritisch nadenken over de werking. Ik heb echter in de gaten dat ik ook tijdens het ontwerpen zelf kritischer moet nadenken over bepaalde functionaliteiten. Tijdens mijn afstudeerproject liep er nog een project dat werd uitgevoerd door de vaste medewerkers. Hierbij werden de tests gemaakt door één persoon, terwijl iemand anders de software schreef. Dit is zeker een goede zaak aangezien de ontwikkelaar blind kan worden voor zijn eigen werk. Maar het maakte ook duidelijk ik zelf kritisch moet nadenken tijdens het ontwerpen van software, en niet tijdens het opstellen van een testplan.

Rapportage
De rapporten die ik opleverde vond ik redelijk in orde. Het gaf goed weer wat ik had gedaan, maar ik had beter kunnen toelichten hoe alles tot stand is gekomen. Daardoor kon alles eenvoudig worden nagelezen in plaats van dat men dit aan mij moet vragen. Ik moet dus goed mijn keuzes blijven toelichten.

Documentatie
Over de documentatie van de engine ben ik tevreden. Door per informatiestroom te beschrijven hoe het werkt met de interface en koppeling is het inzichtelijk voor anderen zonder de broncode te bestuderen. Het gebruik van een HTML Help bestand vind ik zeker een goede oplossing om broncode te documenteren. Voor programmeurs is het eenvoudig te gebruiken en het maakt duidelijk wat er met de interface kan worden gedaan. Ik zal dit zeker vaker toepassen.

8.2 Aanpak

Planning
Het uitvoeren van het project aan de hand van de planning liep in eerste instantie niet zoals ik in gedachte had. Dit had voornamelijk te maken met overschatten in de hoeveelheid werk, en de fusie tussen de twee BOXwise versies. Aan het begin van het project wist ik niet precies hoe hoeveel tijd de ontwikkeling van de interface zou kosten. Daarom had ik de gehele afstudeer periode gepland voor de ontwikkeling van de interface zodat ik zeker wist dat er een goede basis zou zijn. Ik had het echter de planning nauwkeuriger kunnen samenstellen als ik deze vaker had besproken. Door te overleggen of de planning goed was ingeschat en kritisch de activiteiten te definiëren had ik een strakkere planning kunnen opleveren.

De fusie tussen BOXwise P&P en Pro kwam uit een onverwachte hoek. De eerste versie van de planning was namelijk vrij lineair: het begon bij het verkennen, ging door naar het ontwerp en eindigde met de bouw en het testen van de software. Ik had geen rekening gehouden met eventuele veranderingen in het project omdat ik deze niet verwachtte. Maar toen de fusie ter sprake kwam ben ik er wel soepel mee om gegaan. Dit deed ik door naar de planning te kijken en deze aan te passen aan de nieuwe situatie. Hierbij keek ik ook terug naar de activiteiten die ik al had uitgevoerd. Zo kon ik inschatten hoeveel tijd ik kwijt zou zijn om mijn ontwerp aan te passen aan de nieuwe situatie.

Gebruik van RUP
Het gebruik van RUP voor dit project vond ik over het algemeen goed verlopen. Voor mij is het een flexibele methodiek geweest om mee te werken. Bij juist gedefinieerde iteraties hoeven veranderingen in de planning geen grote impact te hebben, RUP geeft de mogelijkheid om een stap terug te doen. Iteratief werken is voor mij zeker een fijne manier van ontwikkelen.

Het gebruik van de RUP fasering heeft mij goed geholpen om mijn project in te plannen. Ondanks de wijzigingen die ik af en toe heb moeten maken, heb ik wel goed gebruik kunnen maken van alle fases. Daarnaast kon ik goed alle activiteiten scheiden door het gebruik van alle fases. Zo heb ik stapsgewijs naar een eindresultaat kunnen werken.

De keuze om niet alle mogelijkheden van RUP te gebruiken heeft voor dit project ook goed uitgepakt. Het gebruik van de fasering en iteraties is bleek voldoende te zijn om het project goed uit te voeren. Ik vind het zeker goed toepasbaar bij kleine projecten, maar zie zeker mogelijkheden om RUP in grotere projecten te gaan gebruiken.

Gebruik van TMap
Het gebruik van TMap is helaas beperkter gebleven dan gedacht. Ik heb me vooral gericht op de unit tests waardoor er weinig tijd over bleef om gebruik te maken van andere tests. Het vormt echter wel een goede basis om te controleren of BOXwise goed functioneert met de interface en koppeling.

Ik vond het wel goed toepasbaar voor mijn project. Door het gebruik van TMap en semantische tests heb ik gestructureerd kunnen testen. Wel moet ik er rekening mee houden dat ik op tijd begin met het testproces.

BOXwise fusie
De beslissing om de twee BOXwise versies te fuseren was een onverwachte wending. Ik ben daar echter flexibel mee omgegaan door de situatie te bekijken en mijn activiteiten aan te passen. Met het gebruik van RUP was ik ook in staat de planning zonder al te grote wijzigingen te corrigeren. Het iteratief werken heeft hier zeker zijn nut bewezen. Wel zou het verstandiger zijn geweest als ik meer tijd had besteed aan het verkennen van BOXwise Pro. BOXwise Pro is qua architectuur anders dan BOXwise P&P, daardoor waren sommige zaken voor mij even onduidelijk. Door actiever vragen te stellen zou ik sneller in staat zijn geweest om te begrijpen hoe BOXwise Pro werkt.

8.3 Competenties

Uitvoeren analyse door definitie van requirements
Door eerst de eisen in kaart te brengen kon ik bepalen aan welke eisen de software moest voldoen. Het opstellen van de eisen verliep soepel, al kon het iets uitgebreider worden gedocumenteerd. Het is nu bij een korte opsomming gebleven. Het zou beter zijn geweest als ik de eisen verder had uitgewerkt waarbij ik beschreef wat de verantwoordelijkheden van de interface en koppelingen beschreef.

Ontwerpen systeemdeel
Het opstellen van het ontwerp vond ik zeer geslaagd voor dit project. Door redelijk diep het ontwerp te specificeren, zoals het uitwisselen van specifieke gegevens, liep de bouw van het systeemdeel voorspoedig. Het zou wel beter zijn geweest als ik kritischer had nagedacht over het ontwerp tijdens het ontwerpen zelf. Tijdens het opstellen van het testplan ging ik nadenken of bepaalde stappen of het gebruik van bepaalde gegevens wel logisch waren. Maar dit moest meer tijdens het ontwerpen plaats vinden. Bij een testgericht software ontwikkeling is deze manier van werken wel mogelijk, maar de kans bestaat dat iemand anders tests gaat ontwikkelen. Het is dan beter als ik meteen kritisch na denk dan dat ik constant feedback krijg van de testers omdat ik niet goed heb nagedacht over een specifiek onderdeel.

De gebruikte UML diagrammen (klassen en sequentiediagrammen) vond ik een geschikte manier om het ontwerp vorm te geven. Hierbij heb ik de use cases achterwege gelaten omdat er voor de gebruikers niets veranderde. Het zou echter wel beter zijn geweest als ik hiervan gebruik had gemaakt om de functionaliteiten van BOXwise in kaart te brengen. Ik realiseer me dat ik als snel mijn aandacht had gericht op de informatiestromen. Maar door ook gebruik te maken van use cases kon ik voor anderen weergeven de functionaliteiten van BOXwise zijn.

Bouwen systeemdeel
Deze competentie heb ik tweemaal uitgevoerd. De eerste keer voor de interface, de tweede keer voor de Exact koppeling. Door het opstellen van iteraties kon ik me richten op een specifieke functionaliteit. Ook door elke informatiestroom stapsgewijs te wijzigen naar de nieuwe situatie (het gebruik van een aparte koppeling) heb ik BOXwise goed kunnen aanpassen. Hierdoor heb ik gestructureerd de interface en koppeling kunnen bouwen. Het zou wel beter zijn geweest als ik de voortgang beter had gecommuniceerd. Dan zouden anderen beter op de hoogte zijn geweest en konden zij eventuele feedback geven. Dit kon alleen maar ten goede komen van het product.

Initiëren en plannen van het testproces
Dit is redelijk goed verlopen. Het testproces heb ik samen laten lopen met het ontwikkelproces zodat de nieuwe broncode direct kon worden getest. Zo kon ik eventuele fouten direct kunnen oplossen. Wel zou het zijn beter geweest als ik tijdens de ontwikkeling van de interface al het testplan had opgesteld. Dit had ik uitgesteld omdat het nog niet

logisch vond om de testgevallen op te stellen. Een interface doet namelijk niets, de werking daarvan kan dus niet getest worden. Daarom stelde ik het initiëren en plannen van het testproces uit tot de ontwikkeling van de koppeling. Dit verliep soepel. Ik was bij de start al begonnen met het testplan en het opstellen van de testgevallen. Door de testwaardes pas tijdens het ontwikkelen op te zoeken, kon ik geconcentreerd aan één onderdeel tegelijk werken. Echter, gezien over het gehele afstudeerproject vond het initiëren en plannen redelijk laat plaats. Het zou beter zijn geweest als ik het testplan meteen bij de ontwikkeling van de interface had opgesteld. Hierdoor zou de gehele voorbereiding eerder hebben plaats gevonden waarbij van tevoren al kon worden nagedacht over de werking van de software.

Ik moet er dus op letten dat ik meteen moet beginnen met het testproces als ik me heb voorgenomen om software gestructureerd te testen. Dit komt alleen maar ten goede van het product.

Uitvoeren van en het rapporteren over het testproces
Het uitvoeren is goed verlopen. Voor elke informatiestroom bouwde ik meteen de unit tests, gebaseerd op de testgevallen in het testplan. Hierdoor heb ik meteen eventuele fouten kunnen ontdekken en oplossen. Door van tevoren al de testgevallen heb ik kunnen bepalen wat ik precies wilde testen. Deze manier van werken vond ik zeker goed passen bij het project.

Het rapporteren van alle resultaten is echter vrij beperkt gebleven. De resultaten heb ik niet samengevoegd in een document omdat alle testen met succes werden uitgevoerd.

Woordenlijst

BOXwise
Een magazijnbeheer oplossing voor het mkb. Technieken als touchscreen en barcodescanning in combinatie met een gebruiksvriendelijk userinterface worden gebruikt om magazijnmedewerkers eenvoudig hun magazijntaken uit te laten voeren. Het werkt met het ERP systeem Exact Globe 2003 om gegevens te tonen en te verwerken.

ERP systeem
Een programma dat binnen een bedrijf wordt gebruikt ter ondersteuning van de bedrijfsprocessen. [13]

Informatiedrager
Object dat wordt gebruikt om de gegevens tussen BOXwise en een ERP systeem uit te wisselen.

Informatiestroom
De gegevens die worden uitgewisseld tussen BOXwise en een ERP systeem.

Interface
Definieert een zogenaamd contract tussen twee applicaties waardoor zij op een gestandaardiseerde manier met elkaar kunnen communiceren.

Klassendiagram
Beschrijft de objecten binnen een programma en de relaties tussen die objecten.

Koppeling
Een extensie dat door BOXwise wordt gebruikt om verbinding met een ERP systeem te maken, om vervolgens gegevens op te vragen / verzenden.

Object
Verzameling van operaties binnen een programma.

RUP
Ration Unified Process. Het is een ontwikkelmethodiek voor softwareontwikkeling waarbij het iteratief werken één van de kenmerken is.

Sequentie diagram
Weergeeft de volgorde van een scenario in een programma.

TMap
Team Management Approach. Het is een gestructureerde testmethodiek. Met behulp van TMap is het mogelijk om vanaf het begin van een software ontwikkeltraject actief bezig te zijn met testen.

UML
Unified Modeling Language. Het is een modelleertaal om ontwerpen voor informatiesystemen te kunnen maken

[13] Bron: http://nl.wikipedia.org/wiki/Enterprise_resource_planning

Gebruikte bronnen

BOXwise
http://BOXwise.nl/

CARTwise
http://cartwise.nl/

Microsoft Best Practices
http://msdn.microsoft.com/en-us/practices/default.aspx

Patterns
http://www.dofactory.com/Patterns/Patterns.aspx

RUP
http://www.rupopmaat.nl/

Testen volgens TMap
Auteurs : Martin Pol, Ruud Teunissen, Erik van Veenendaal
Druk : 2e druk
Uitgever : Uitgeverij Tutein Nolthenius

TranCon
http://www.trancon.nl/

Bijlage A: Plan van Aanpak

Plan van Aanpak

Auteur: Ray Wijshake

Datum: 19 februari 2010

Versie: 2

Inhoudsopgave

1. Inleiding

Dit document is opgesteld in het kader van het afstuderen bij TranCon. Hierin zal worden beschreven hoe het project wordt aangepakt, wat de planning is, en wat er wordt opgeleverd.

In het volgende hoofdstuk wordt eerst de opdracht besproken. Vervolgens zal er in hoofdstuk 3 dieper op de aanpak worden ingegaan. Er zal onder andere worden besproken welke methodieken er gebruikt worden en hoe de voortgang wordt bewaakt. In hoofdstuk 4 zal de planning worden besproken.

2. Opdracht

In dit hoofdstuk zal er nader worden ingegaan op de opdracht.

2.1 Opdrachtomschrijving

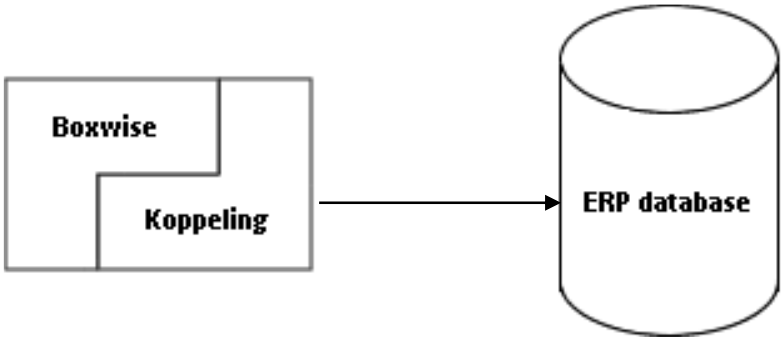
De opdracht omvat het ontwerpen, bouwen en testen van een generieke engine die de koppeling tussen Boxwise en een willekeurig ERP systeem mogelijk maakt.

2.2 Boxwise

Boxwise is een warehouse management oplossing voor het MKB. Door de combinatie van touchscreen, gebruiksvriendelijke interface en het gebruik van barcodescanning is het voor magazijnmedewerkers eenvoudig om orders te verzamelen. Hierbij worden de gegevens direct uit het ERP systeem Exact Globe 2003 gehaald en verwerkt waardoor de orders altijd up-to-date zijn.

2.3 Huidige situatie

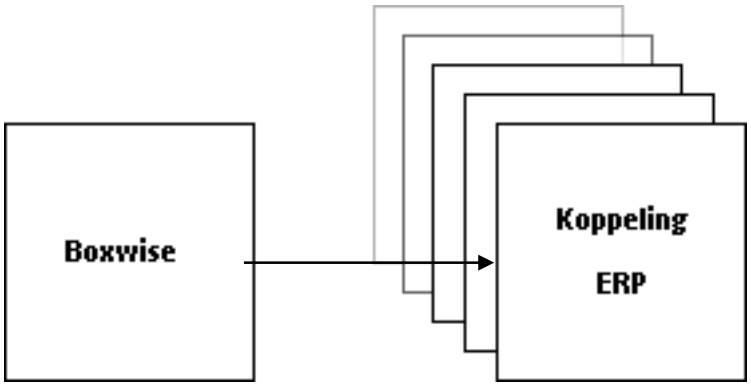
In de huidige situatie, zoals schematisch weergeven is in figuur 1, is de koppeling in de code van Boxwise verweven. Als Boxwise informatie uit het ERP systeem nodig heeft, zal het via de ingebouwde koppeling connectie maken met de database van het ERP systeem. Vervolgens krijgt Boxwise de informatie waar het om heeft gevraagd, en zal deze verder verwerken. Deze koppeling werkt zoals het moet werken, maar het biedt geen mogelijkheid om met een ander ERP systeem verbinding te maken. Op dit moment werkt de koppeling alleen met het ERP systeem Exact Globe 2003. Indien Boxwise met een ander ERP systeem moet communiceren, zal de code aangepast moeten worden. Dit maakt het onderhoud erg arbeidsintensief naar mate er meer ERP systemen ondersteund moeten worden.



Figuur 1: Huidige situatie

2.4 Gewenste situatie

Om Boxwise generieker te maken, zal de koppeling tussen Boxwise en het ERP systeem worden losgetrokken. Dit is schematisch weergegeven in figuur 2.



Figuur 2: Gewenste situatie

Door van de koppeling een aparte abstractielaag te maken, is het mogelijk om eenvoudiger om een koppeling met een willekeurig ERP systeem te maken. Boxwise blijft de functionaliteiten bieden die het al kon bieden. Het verschil is dat Boxwise voortaan aan de koppeling vraagt om gegevens uit het ERP systeem te halen. De koppeling bepaalt vervolgens zelf hoe de verbinding met het ERP systeem wordt gemaakt en hoe de gegevens worden opgevraagd. Boxwise houdt zich hier helemaal niet mee bezig, maar krijgt uiteindelijk via de koppeling wel de informatie waarom het heeft gevraagd.

Door de nieuwe koppeling kan Boxwise alsnog informatie van een ERP systeem krijgen, maar door de koppeling dus gescheiden te houden en generiek op te zetten, is het mogelijk dat er ook andere ERP systemen gebruikt kunnen worden. Hierdoor is Boxwise flexibeler, en wordt een grotere markt bediend.

3. Aanpak

In dit hoofdstuk zal de aanpak van het project worden toegelicht.

3.1 RUP

Voor het managen van dit project zal er gebruik gemaakt gaan worden van RUP. RUP staat voor Rational Unified Process en is een systeemontwikkeltechniek. Deze methodiek biedt o.a. verschillende werkwijzen, technieken en richtlijnen aan, waarbij er gewerkt wordt met iteraties. Dit houdt in dat er korte periodes worden gepland waarbij er aan een specifiek onderdeel wordt gewerkt.
RUP definieert ook een aantal fases voor het ontwikkelproces, en deze worden ook gebruikt om dit project in te delen:

- **Inception**
  In deze fase wordt het project gestart, en zal er worden gekeken of het project haalbaar is. Daarnaast wordt onder andere het Plan van Aanpak worden opgesteld en de risico's geïnventariseerd.

- **Elaboration**
  Gedurende deze fase zal het ontwerp worden opgesteld. Daarnaast zal het testplan worden opgesteld. De fase wordt afgesloten met het Elaboration rapport waarin het gekozen ontwerp wordt toegelicht.

- **Construction**
  Na het opstellen van het ontwerp zal er in deze fase daadwerkelijk gestart worden met de bouw.

- **Transition**
  In deze fase wordt de gebouwde software getest. Uiteindelijk zal het testrapport en de technische documentatie worden opgesteld.

Hierbij is het de bedoeling dat er iteratief wordt gewerkt. Aan de hand van de requirements zullen de functionaliteiten met een hogere prioriteit eerst worden gebouwd en getest. Deze cyclus zal zich herhalen totdat het correct functioneert. Vervolgens wordt de volgende functionaliteit opgepakt, en herhaalt de cyclus zich weer, totdat het volledige product is gebouwd en getest.

3.2 UML

UML staat voor Unified Modelling Language. Om de werking van het nieuwe stuk software te beschrijven, zal hiervan gebruik worden gemaakt. Hierbij zullen een aantal relevante diagrammen gebruikt worden:

- **Use cases**
  Voor het schematisch weergeven van de mogelijke functionaliteiten.

- **Datamodel**
  Voor het weergeven van de gebruikte tabellen die Boxwise nodig heeft, en welke relevant zijn voor dit project.

- **Klassendiagrammen**
  Voor het beschrijven van de classes.

- **Sequentie diagrammen**
  Voor het beschrijven van de informatiestromen binnen Boxwise.

- **Toestand diagrammen**
  Voor het beschrijven van de verschillende toestanden van de objecten.

Door middel van deze diagrammen met beschrijvende teksten moet het duidelijk worden wat de functionaliteiten zijn en hoe deze werken.

3.3 Testen

Om te kunnen garanderen dat de nieuwe koppeling goed functioneert, zal deze worden getest. Hierbij zal TMap worden gebruikt. Dit begint met het opstellen van een testplan in de Elaboration fase, waarin wordt beschreven wat er wordt getest, welke testmethoden er worden gebruikt en hoe de bevindingen worden gerapporteerd. Vervolgens zal er tijdens de bouw al worden getest, elke keer als er een functionaliteit gereed is. Mochten er bepaalde bevindingen worden gedaan, dan kunnen deze tijdens de bouw al worden verholpen. Hierdoor wordt er voorkomen dat er aan het einde van het project een niet werkend product wordt opgeleverd. Uiteindelijk zal er een testrapport worden opgesteld waarin de testresultaten worden beschreven, met eventuele aanbevelingen.

3.4 Bewaken voortgang

Dit project duurt 17 weken. Om te voorkomen dat de opdracht niet kan worden afgerond is het van belang om de voortgang te bewaren. Zo kan er op tijd worden gesignaleerd of de einddatum niet kan worden gehaald. Daarom zullen er een aantal maatregelen worden genomen:

- **Voortgangsgesprekken**
  Tijdens deze gesprekken zal er worden besproken hoe het project verloopt, wat de status is en welke problemen er zijn. Mochten de problemen dermate groot zijn dat de deadline van het gehele project in gevaar komt, dan moet er worden bekeken welke activiteiten een lagere prioriteit krijgen.

- **Deadlines producten**
  Voor de producten die worden opgeleverd, maar ook voor de deelproducten, zullen er deadlines worden opgesteld. Voor de rapporten en het Proof of Concept zijn de deadlines al voor aanvang van het project vastgelegd.

- **Planning**
  Bij start van het project is een globale planning opgesteld. Daarnaast zal er voor elke fase een detailplanning worden opgesteld. Er zal regelmatig op de planning worden teruggekeken, om zo te controleren welke activiteiten uit kunnen lopen. Indien dit het geval is, moet er worden gekeken welke activiteiten een lagere prioriteit krijgen.

- **Portfoliowebsite**
  Er is een portfoliowebsite opgezet om de rapporten op te zetten. Door de conceptversies en definitieve versies daar op te zetten, kunnen de betrokken personen de voortgang zien. Daarnaast zullen zij ook op de hoogte worden gesteld als er een nieuwe versie online staat, en zal er worden gevraagd of er feedback kan worden gegeven.

3.5 Risico analyse

Bij aanvang van het project zullen de risico's in kaart worden gebracht. Op deze punten zal extra op moeten worden gelet om te voorkomen dat het eindproduct niet op tijd kan worden opgeleverd.

3.6 Rapporten

Er zullen diverse documenten worden opgesteld. Deze zullen aan de start en einde van een fase worden opgesteld. Zo kunnen de betrokken personen lezen wat er gaat gebeuren, en wat er is gedaan. De rapporten zullen ook beschikbaar worden gesteld op de portfoliowebsite, zodat deze te allen tijde ingezien kunnen worden.

3.7 Documentatie

Om verdere ontwikkeling en beheer eenvoudiger te maken, maar ook om eenvoudig de werking van de software te kunnen beschrijven, zal er documentatie worden opgesteld. Dit zal gebeuren door technische documentatie te schrijven, waarin door middel van UML diagrammen de werking wordt beschreven. Daarnaast zal de code zelf ook van documentatie worden voorzien waarin wordt beschreven hoe het werkt. Uiteindelijk kunnen er met behulp van bepaalde programma's zoals Sandcastle, documenten worden gegenereerd. Hierdoor hoeft de code niet te worden bekeken om te begrijpen hoe het werkt. Aan het einde van het project zal er ook een testrapport worden opgeleverd waarin de bevindingen van alle testen staan.

3.8 Software

De volgende software zal worden gebruikt tijdens het project:

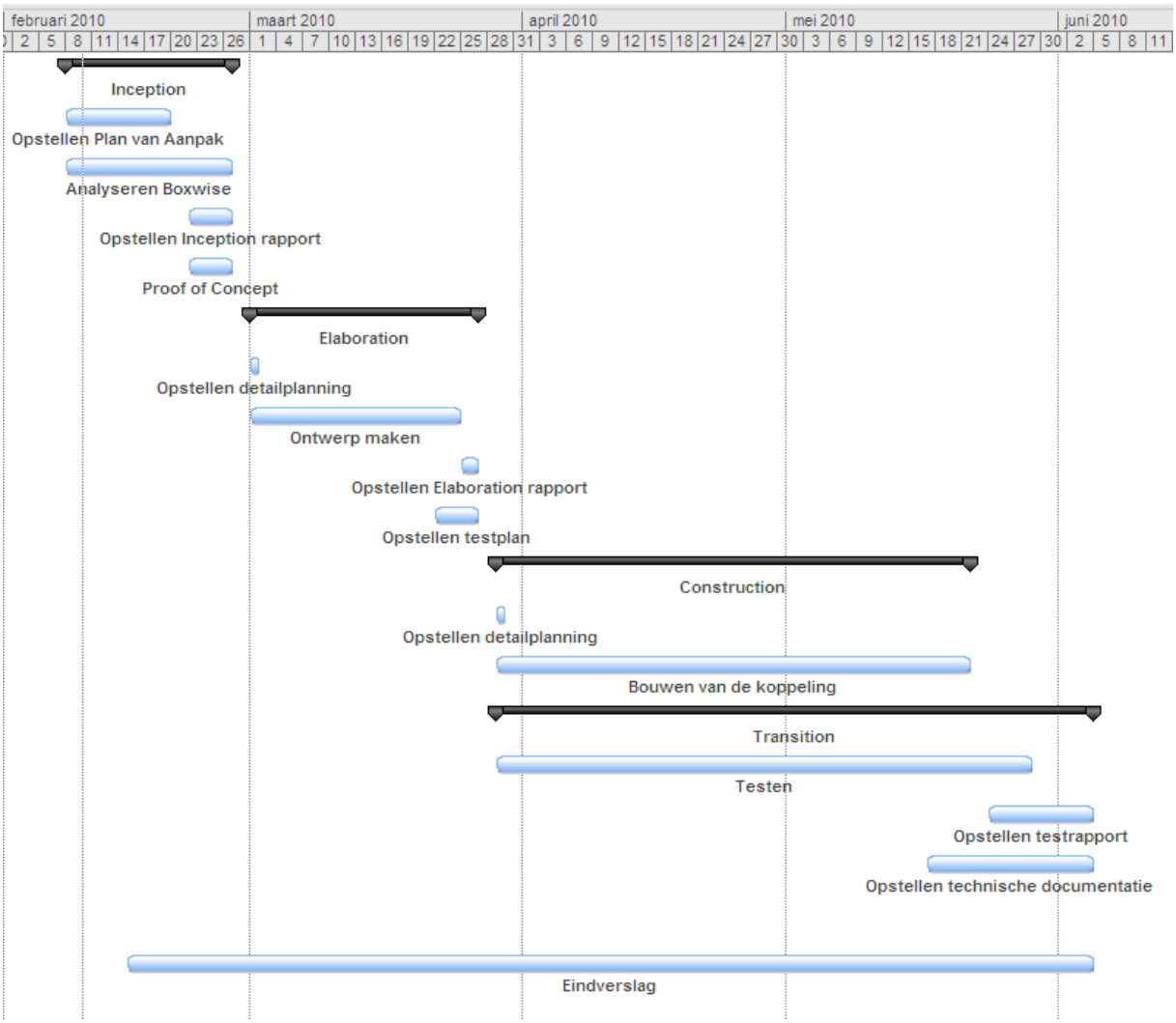
- Microsoft Office voor de documentatie en planning
- Microsoft Office voor het opstellen van de UML diagrammen
- Visual Studio 2005 voor het bouwen van de software
- Sandcastle voor de technische documentatie

4. Planning

In dit hoofdstuk zal de planning en mijlpaalproducten worden beschreven.

4.1 Globale planning

Hieronder staat de globale planning voor dit project en beschrijft in hoofdlijnen wat de activiteiten zullen zijn.



Figuur 3: Globale planning

4.2 Detailplanning

In elke fase zal er een detailplanning worden gemaakt. Hierin zal worden aangegeven welke taken worden uitgevoerd en hoeveel tijd deze in beslag zullen nemen. Deze planningen zullen op de portfoliowebsite worden gezet en worden besproken met de begeleidende personen.

4.3 Mijlpaalproducten

In de onderstaande tabel zijn de mijlproducten weergeven. Deze hebben een vaste week waarin het ingeleverd zal worden. Deze producten zijn als mijlpaalproducten gekozen omdat ze een start of einde van een fase aanduiden.

Producten	Week
Plan van Aanpak	2
Analyse van Boxwise en de huidige koppeling	3
Inception rapport	3
Elaboration rapport	7
Testplan	7
Technische documentatie	17
Testrapport	17
Eindverslag	17

Bijlage B: Inception rapport

Inception rapport

Auteur: Ray Wijshake

Datum: 11 juni 2010

Versie: 1

Inhoudsopgave

1. Inleiding

Dit document is opgesteld als einddocument van de Inception fase. Hierin is onderzocht wat Boxwise precies is, wat er moet veranderen, en welke eisen daar aan worden gesteld.

Het begint in hoofdstuk 2 waarbij de opdracht nog een keer wordt beschreven. Vervolgens zal in hoofdstuk 3 het product Boxwise worden toegelicht, waarbij er wordt beschreven wat er zal veranderen aan dit product. In hoofdtuk 4 zullen de eisen worden toegelicht, waarna er in hoofdstuk 5 de informatiestromen van Boxwise worden toegelicht. Tot slot zal in hoofdstuk 6 het Proof of Concept en ideeën die mogelijk waren, toegelicht.

2. De opdracht

In dit hoofdstuk wordt de opdracht nog een keer omschreven. Hierbij is ook een afbakening gemaakt i.v.m. de beschikbare tijd.

2.1 Opdrachtomschrijving

Het ontwerpen, bouwen en testen van een generieke engine die de koppeling tussen Boxwise en een willekeurig ERP systeem mogelijk maakt.

2.2 Afbakening

De opdracht is enigszins afgebakend. Dit is gedaan omdat het bouwen van de generieke engine EN het lostrekken van de huidige koppeling teveel tijd kan kosten. Daarom is de opdracht afgebakend om te voorkomen dat er een half voltooid product wordt opgeleverd. Door deze afbakening is de opdracht in drie of meerdere projecten op te splitsen:

1 **Ontwikkeling van de generieke “engine”**
Deze engine vormt de basis voor het gebruik van Boxwise met meerdere ERP systemen.

2 **Lostrekken van de huidige Exact Globe koppeling in Boxwise, en hiervan een aparte koppeling bouwen.**
De koppeling bevat alle informatie die Boxwise nodig heeft om met het ERP systeem Exact Globe 2003 te kunnen werken, en kan worden aangesproken door de generieke engine.

3 - ∞ **Bouwen van de overige koppelingen, waardoor Boxwise met andere ERP systemen kan werken (zoals Afas Profit, Agresso Business World, etc).**
Deze koppelingen bevatten ook alle informatie dat Boxwise kan gebruik om met andere ERP systemen te kunnen werken.

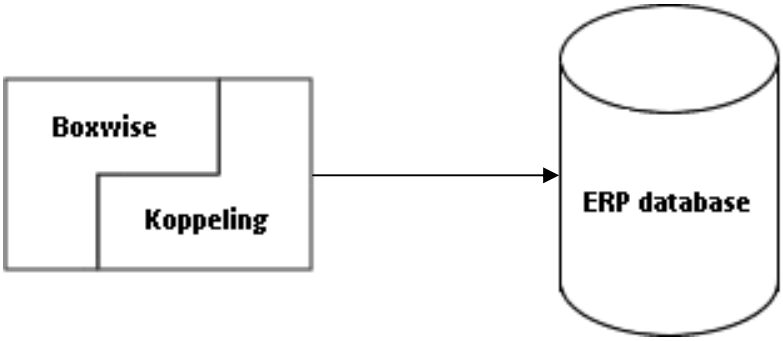
Tijdens dit afstuderen zal de focus liggen op het ontwikkelen van de generieke engine. Mocht er tijd over zijn, dan kan er een start worden gemaakt met het lostrekken van de huidige koppeling in Boxwise met Exact Globe.

3. Boxwise

Boxwise is een warehouse management oplossing voor het MKB. Door de combinatie van touchscreen, gebruiksvriendelijke interface en het gebruik van barcodescanning is het voor magazijnmedewerkers eenvoudig om orders te verzamelen. Hierbij worden de gegevens direct uit het ERP systeem Exact Globe 2003 gehaald en verwerkt waardoor de orders altijd up-to-date zijn.

3.1 Huidige situatie

In de huidige situatie kan Boxwise alleen een verbinding maken met het ERP systeem van Exact, Exact Globe 2003. Dit is schematisch weergegevens in afbeelding 1.



Afbeelding 22: Huidige situatie

De koppeling met Exact Globe is verweven in de code van Boxwise. Indien Boxwise gegevens van het ERP systeem nodig heeft, maakt het via de ingebouwde koppeling verbinding met de database van het ERP systeem. De gevraagde informatie wordt opgehaald en vervolgens kan Boxwise deze gegevens gebruiken voor:

- **Het ontvangen van goederen.** Goederen komen een magazijn binnen, en met Boxwise kunnen deze worden geregistreerd, gecontroleerd en worden opgeslagen in de database van het ERP systeem.
- **Het verzenden van goederen.** Klanten plaatsen orders, en met behulp van Boxwise kunnen deze orders uit het ERP systeem worden gehaald. Vervolgens kan Boxwise de picklijst opstellen waarmee vervolgens de orders kunnen worden verzameld. Als de orders zijn verzameld, wordt er in Boxwise geregistreerd welke artikelen zijn verzameld. Het kan immers gebeuren dat een product niet op voorraad was, waardoor een order niet volledig kan worden verstuurd. Boxwise verwerkt de gegevens van de orders, en kan hierbij ook de geschiedenis bijhouden.
- **Tellingen.** Het is mogelijk dat de fysieke voorraad van artikelen niet overeen komt zoals deze in het ERP systeem is geregistreerd. Daarom moeten er tellingen worden uitgevoerd zodat de voorraad die in het ERP systeem is geregistreerd, ook overeen komt met de fysieke voorraad.

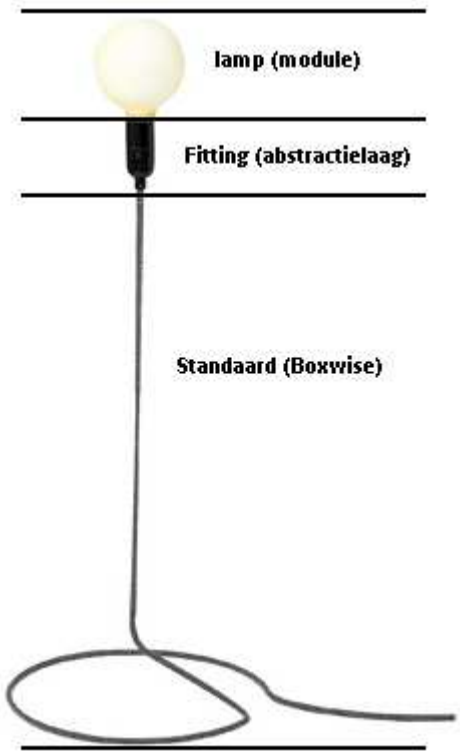
De huidige koppeling werkt zoals het moet werken, maar het grote nadeel is dat het maar slechts met één ERP systeem werkt: Exact Globe 2003.

3.2 Nieuwe situatie

Om Boxwise meer flexibel te maken, zal de koppeling tussen Boxwise en het ERP systeem worden losgetrokken. Boxwise moet helemaal niks weten van het ERP systeem waarmee het gaat werken. Al deze kennis moet apart worden aangeleverd, zodat er kan worden gekozen met welk ERP systeem Boxwise gaat werken. De nieuwe situatie is schematisch weergegeven in afbeelding 2.

Het is te vergelijken met een lamp. Een lamp heeft meerdere dingen nodig om te kunnen werken:

- **Een standaard:** iets waardoor de lamp zijn werk kan doen, op de juiste positie.
- **De fitting:** de lamp moet ergens kunnen worden ingedraaid, zodat het de stroom krijgt om licht te geven.
- **De lamp** zelf, iets wat geeft als wij het nodig hebben in het donker, namelijk licht.

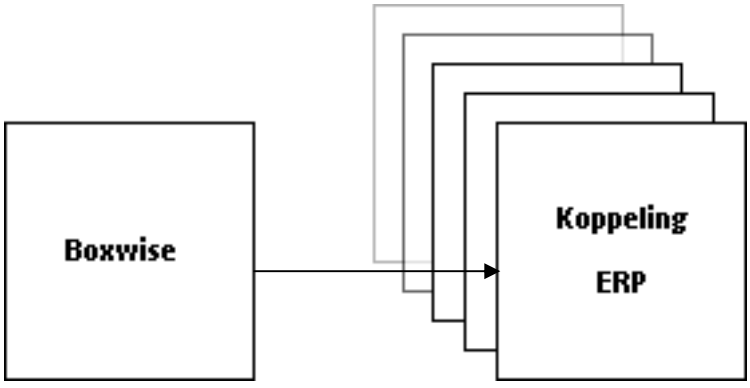


Afbeelding 23: Nieuwe situatie (metafoor)

Om dit nu te vergelijken met Boxwise:

- **Een standaard:** dit is Boxwise. Het maakt niet uit waar je het neer zet, het is er. Alleen doet het nog niet zo veel, daarvoor heeft het meer nodig.
- **De fitting:** iets wat in een standaard kan worden geplaatst zodat er uiteindelijk een lamp in kan worden gedraaid. In dit geval is dat een abstractielaag dat door Boxwise kan worden aangeroepen, om zo zijn benodigde informatie binnen te halen. Deze abstractielaag kan vervolgens een willekeurige module aanspreken om verbinding met het ERP systeem te maken (zie volgend punt).
- **De lamp:** iets wat in de fitting kan worden gedraaid, en waardoor het geheel compleet is. In dit geval is dat een generieke module (de lamp) dat de functionaliteiten bevat om gebruik te maken van de database van een ERP systeem. Deze module kan door Boxwise worden gebruik, via de abstractielaag (de fitting). Het maakt niet uit wat voor lamp er wordt gebruikt (spaarlamp, ledlamp, etc.), of in het geval van Boxwise, welke module er wordt gebruikt (voor Exact Globe, Agresso Business World, etc.). Zolang de abstractielaag de module kan aanspreken, is het mogelijk om Boxwise met elk ERP systeem te laten werken.

Dus door deze manier is Boxwise veel flexibeler, en kan het met meerdere ERP systemen werken in plaats van slechts één ERP systeem.



Afbeelding 24: Nieuwe situatie

4. Eisen

Voor dit project zijn een aantal eisen opgesteld. Hierbij wordt aangegeven welke eisen een hogere prioriteit hebben, waarbij 1 de hoogste prioriteit is, en 3 de laagste.

1. Generiek	**1**
Het moet dermate generiek zijn opgezet dat het automatisch: - Nieuwe koppelingen kan detecteren - Een standaard koppeling kan gebruiken - Instellingen kan uitlezen (zie eis 2)	

2. Instellingen voor koppelingen	**1**
De instellingen moeten zodanig worden opgeslagen dat deze eenvoudig en op een generieke wijze uit te lezen zijn. De instellingen zullen bestaan uit: - Logo van het desbetreffende ERP systeem - Functionaliteiten die Boxwise met het ERP systeem kan uitvoeren (Boxwise heeft een aantal functionaliteiten, maar het is mogelijk dat niet alles alle input door een ERP systeem geleverd kan worden).	

3. Zelftest	**3**
Als de juiste koppeling met de instellingen is gekozen en geïnitialiseerd, moet er door middel van een zelftest worden gecontroleerd of er daadwerkelijk een connectie met het desbetreffende ERP systeem kan worden opgezet.	

4. Performance	**2**
De performance moet niet significant achteruit gaan door de implementatie van de nieuwe koppeling.	

5. Updates	**3**
Eenvoudige manier om nieuwe koppelingen toe te voegen of nieuwe versies te plaatsen.	

Naar mate het ontwerp steeds meer in kaart wordt gebracht, zullen deze eisen ook worden vertaald naar de functionaliteiten van de generieke engine. Hierbij zal worden aangegeven welke functionaliteiten een hogere prioriteit hebben, en welke slechts een extraatje zijn.

5. Analyse informatiestromen

Met Boxwise is het dus mogelijk om bepaalde informatie overzichtelijk te presenteren en te verwerken. De gegevens moeten ergens vandaan komen, in dit geval van een ERP systeem. Hiervoor maakt Boxwise verscheidene malen verbinding met het ERP systeem. In de huidige situatie beschikt Boxwise zelf nog over deze intelligentie, maar het is de bedoeling dat deze intelligentie wordt verplaatst naar een aparte module. De informatiestromen die hierbij horen, zullen dus uiteindelijk anders verlopen. Daarom zijn deze eerst in kaart gebracht, zodat het duidelijk is op welke moment Boxwise informatie van een ERP systeem nodig heeft, en hoe de daarbij behorende informatiestromen verlopen. Deze informatiestromen zullen in dit hoofdstuk worden toegelicht, en zijn onder te verdelen in drie paragrafen:

- Ontvangen van goederen
- Verzenden van goederen
- Tellingen

Voor elk informatiestroom is een stroomdiagram opgesteld. Deze zijn opgenomen in Bijlage A, en door middel van nummering zal naar de desbetreffende diagram worden verwezen.

5.1 Ontvangen van goederen

Voor deze informatiestromen worden de stroomdiagrammen 1 t/m 5 gebruikt.

1. Ophalen leveranciers

Deze stap omvat het ophalen van de leveranciers

- De gebruiker start in het hoofdscherm.
- Er wordt op het menu item “Ontvangen” geklikt.
- Boxwise maakt connectie met de Exact database.
- De leveranciers worden opgevraagd en vervolgens weergeven.

2. Ophalen orders van leveranciers

Deze stap omvat het ophalen van de orders van de leveranciers.

- De gebruiker heeft het scherm “Ontvangen” voor zich.
- Er wordt op een leverancier geklikt.
- Boxwise maakt verbinding met de Exact database.
- De orders van de geselecteerde leverancier worden opgehaald.
- De orders worden aan de gebruiker weergeven.

3. Ophalen artikelen van een order

Deze stap omvat het ophalen van de artikelen van een geselecteerde order. Vervolgens worden deze weergeven aan de gebruiker.

- De gebruiker heeft het scherm “Selecteer bestellingen” voor zich.
- Er wordt op een order geklikt.
- Boxwise maakt connectie met de Exact database en vraagt de artikelen van de geselecteerde order
- Boxwise krijgt de gegevens en weergeeft deze aan de gebruiker.

4. Order verwerken

Deze stap omvat het verwerken van een order, nadat de artikelen zijn gescand.

- De gebruiker heeft het scherm “Uitpakken” voor zich.
- Er wordt op “Afronden” geklikt.
- Boxwise roept de Exact SDK aan om de order te verwerken.
- Als de orders goed zijn verwerkt, kan de gebruiker vervolgens labels uitprinten. Indien er fouten zijn opgetreden, wordt deze weergeven.

5. Lijst leveranciers verversen

Deze stap omvat het verversen van de lijst met leveranciers in het scherm “Ontvangen”

- De gebruiker klikt op “Verversen”.
- Boxwise maakt connectie met de Exact database.
- De leveranciers worden opgevraagd en vervolgens weergeven.

5.2 Verzenden van goederen

Voor deze informatiestromen worden de stroomdiagrammen 6 t/m 13 gebruikt.

6. Import orders

Deze stap omvat het maken van een snapshot van de Exact database. De informatie wordt uit de Exact database gehaald, en vervolgens opgeslagen in de eigen database. Hiermee wordt voorkomen dat orders of artikelen worden bewerkt in het ERP systeem terwijl deze worden verzameld in het magazijn.

- Boxwise maakt connectie met de Exact database.
- Alle ordergegevens worden opgehaald.
- Voordat de snapshot in de eigen database wordt opgeslagen, worden de tabellen geleegd.
- De snapshot wordt opgeslagen.
- De order informatie wordt vervolgens uit de eigen database gehaald.

7. Orders weergeven

Deze stap omvat het weergeven van de orders aan de gebruiker.

- De gebruiker start in het hoofdscherm.
- Er wordt op “Verzenden” geklikt.
- Boxwise maakt verbinding met de eigen database, en haalt de openstaande orders op.
- De orders worden weergeven aan de gebruiker.

8. Orders weergeven (label)

Deze stap omvat het weergevan van de orders met een specifieke label.

- De gebruiker is in het scherm “Verzenden”, en klikt op een label.
- Boxwise maakt verbinding met de eigen database, en haalt de orders met de geselecteerde label op.
- De orders worden weergeven aan de gebruiker.

9. Ophalen artikelen van een order

Deze stap omvat het ophalen van de artikelen van een geselecteerde order.

- De gebruiker klikt op een order.
- Boxwise maakt connectie met de eigen database, en haalt de artikelen van de geselecteerde order op.
- De artikelen worden weergeven aan de gebruiker.

10. Order afronden

Deze stap omvat het afronden van een order, zodat het kan worden verstuurd naar de klant.

- Er wordt op “Afronden” geklikt.
- Boxwise maakt connectie met de Exact database
- De order wordt als “unlocked” in de database gemarkeerd, zodat het kan worden verwerkt.
- De order informatie wordt naar de Exact SDK gestuurd, die het vervolgens in Exact verwerkt.
- De order wordt weer als “locked” gemarkeerd.
- De gebruiker kan de labels uitprinten, of krijgt een melding indien er fouten zijn opgetreden.

11. Order info

Deze stap omvat het weergeven van alle beschikbare informatie van een order.

- Er wordt op “Order info” geklikt.
- Boxwise maakt verbinding met de Exact database
- De order informatie wordt opgehaald.
- De naam van de ontvanger wordt opgehaald.
- Het adres van de ontvanger wordt opgehaald.
- Vervolgens wordt er een verbinding gemaakt met de eigen database
- De geschiedenis van de order wordt opgehaald, waarmee vervolgens een grafiek wordt weergeven.

12. Detailinformatie van een artikel

Deze stap omvat het weergevan een de gedetailleerde informatie van een geselecteerde artikel.

- De gebruiker selecteert een artikel.
- Er wordt op “Artikel info” geklikt.
- Boxwise maakt connectie met de Exact database, om zo de gedetailleerde informatie van een artikel op te halen.
- De gedetailleerde informatie wordt aan de gebruiker weergeven.

13. Barcode van een artikel wijzigen

Deze stap omvat het wijzigen van de barcode van een artikel.

- De gebruiker klikt op “Barcode”.
- Boxwise haalt alle codes van de geselecteerde artikel op.
- Nadat de barcode van het artikel is gescanned, wordt de code van het artikel gewijzigd

5.3 Tellingen

Voor deze informatiestromen worden de stroomdiagrammen 14 t/m 16 gebruikt.

14. Ophalen magazijnen

Deze stap omvat het ophalen van de magazijnen. De gebruiker moet vervolgens aangeven voor welk magazijn de tellingen uitgevoerd moeten worden.

- De gebruiker start in het hoofdscherm.
- Er wordt op “Tellingen”geklikt.
- Boxwise maakt verbinding met de Exact database.
- De magazijnen worden uit de Exact database opgehaald.
- De magazijnen worden aan de gebruiker weergeven.

15. Selecteer magazijn

Deze stap omvat het selecteren van een magazijn waarvoor de tellingen uitgevoerd moeten worden.

- De gebruiker selecteert een magazijn.
- Boxwise maakt verbinding met de Exact database.
- De producten van de geselecteerde magazijn worden opgehaald.
- De producten worden weergeven aan de gebruiker.

16. Afronden

Deze stap omvat het afronden van de tellingen, waarbij de voorraad wordt bijgewerkt.

- Er wordt op “Afronden” geklikt.
- Boxwise maakt verbinding met de eigen database
- De gegevens van de tellingen worden opgehaald.
- Boxwise roept de Exact SDK aan
- De gegevens van de tellingen worden naar de SDK gestuurd, waarna de voorraad wordt bijgewerkt.

6. Proof of Concept

De Inception fase is afgerond met het opleveren van een Proof of Concept. Hiermee werd aangetoond dat het idee mogelijk is, wat er mogelijk is, en dat er verder kan worden gebouwd met dit ontwerp.

Voor dit PoC werden de volgende functionaliteiten aangetoond:

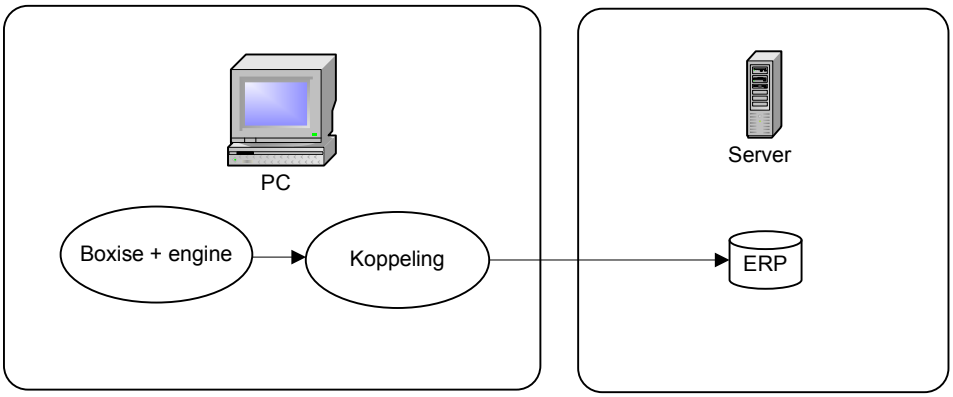
- Boxwise moet controleren op beschikbare koppelingen. Zonder die koppelingen is het niet mogelijk om gebruik te maken van een ERP database.
- Boxwise kan een snapshot van een order maken, waarbij de informatie via de externe koppeling uit de ERP database wordt gehaald. Het maken van een snapshot is essentieel zodat een order niet wordt gewijzigd in het ERP systeem terwijl deze wordt verzameld in het magazijn.
- Boxwise kan een order verwerken, waarbij de informatie via de koppeling naar de Exact SDK wordt gestuurd, die het vervolgens verwerkt.

6.1 Oplossingen

Om eerst terug te gaan naar het idee dat was ontstaan: Boxwise gaat een engine aanspreken om informatie van een ERP systeem op te vragen. De engine bepaalt vervolgens welke koppeling gebruikt moet worden. Een koppeling bevat alle intelligentie die nodig is om informatie van een ERP systeem op te vragen of te verwerken. De engine gebruikt de koppeling om de juiste informatie op te halen. Tot slot krijgt Boxwise zijn gevraagde informatie van een koppeling terug, via de abstractielaag.
Door de intelligentie die nodig is om gegevens uit een ERP systeem te halen, of te bewerken, in een apart bestand te plaatsen, is het mogelijk om Boxwise meer flexibel te maken. Boxwise zelf hoeft niet aangepast te worden om met een ander ERP systeem te communiceren, maar gaat gebruik maken van aparte bestanden die zulke intelligentie bevat. Hierdoor is het slechts een kwestie van het aanleveren van een nieuw bestand, en Boxwise kan gegevens uit een ander ERP systeem halen.

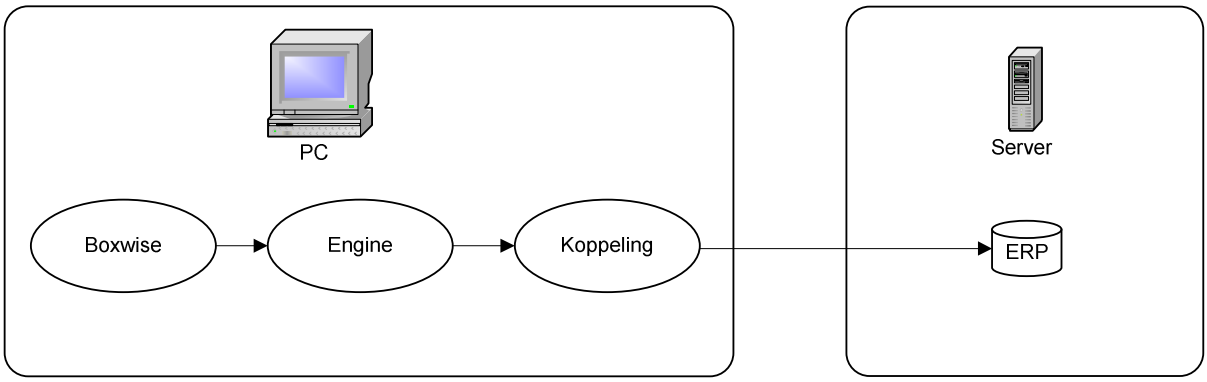
Voor dit idee waren er meerdere oplossingen mogelijk, die uiteindelijk lokaal op een pc of op een server geplaatst zouden worden:

Lokaal, interne engine + externe koppeling
Binnen Boxwise zelf zou de engine worden aangemaakt. Dit is de centrale aanspreekpunt voor Boxwise om gegevens van een ERP systeem via een externe koppeling op te vragen. De externe koppeling is een apart bestand dat op de pc zelf zou worden geplaatst. Boxwise spreekt via de engine het bestand aan, en kan gegevens binnenhalen of naar een ERP systeem sturen. Door de engine intern in te bouwen, en de koppelingen lokaal op te slaan, is het eenvoudiger Boxwise om er gebruik van te maken zonder dat de performance significant afneemt. Het nadeel hiervan is dat elke pc moet worden geüpdatet als er een nieuwe koppeling beschikbaar is.



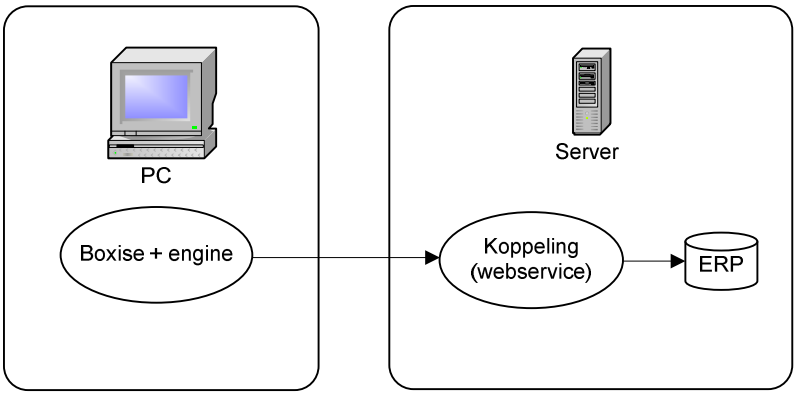
Afbeelding 25: Interne engine + externe koppeling

Lokaal, exern engine en koppeling
Dit is hetzelfde idee als het voorgaande idee, maar de engine is in dit geval een apart bestand. Technisch gezien zijn er echter extra stappen nodig om gegevens uit een ERP systeem te halen, of te bewerken, waardoor de performance af kan nemen.



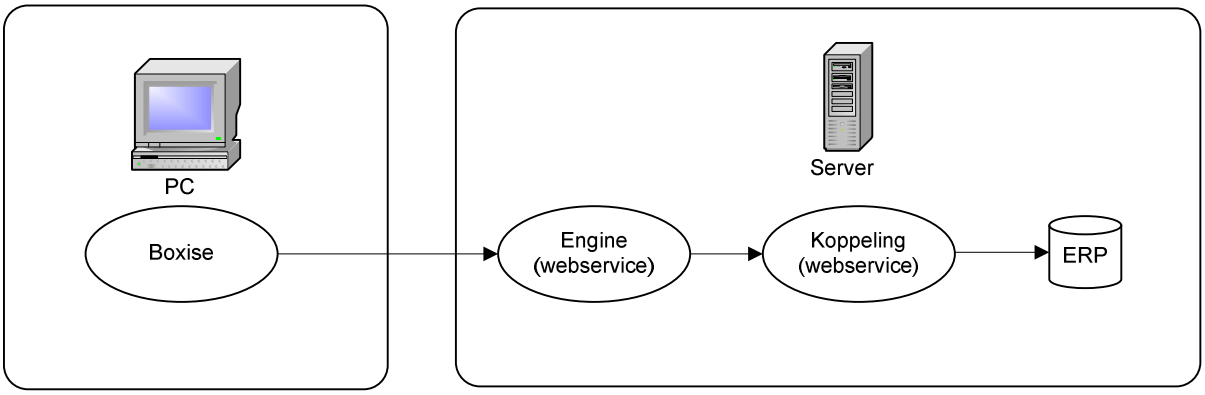
Afbeelding 26: Externe engine en koppeling

Webservices
Boxwise krijgt wederom een engine, die vervolgens een webservice aan zal spreken om gegevens van een ERP systeem op te vragen. Een webservice bevat ook bepaalde functionaliteiten, maar heeft als voordeel dat deze binnen een bedrijfsnetwerk op een server of zelfs op het internet kan worden gezet. Hierdoor zijn de functionaliteiten in één keer beschikbaar voor alle Boxwise cliënts in een bedrijf. Mocht er een nieuwe koppeling nodig zijn, dan wordt deze via een webservice beschikbaar gesteld, en meteen kunnen alle cliënts deze gebruiken.
Voor het uitwisselen van gegevens wordt er bij webservices standaard XML bestanden gebruikt. Dit heeft als voordeel dat de gegevens altijd op een gestructureerde manier worden uitgewisseld, maar dit betekend ook dat er een extra slag nodig is om de gegevens weer beschikbaar te maken voor Boxwise.



Afbeelding 27: Interne engine + externe koppeling als webservice

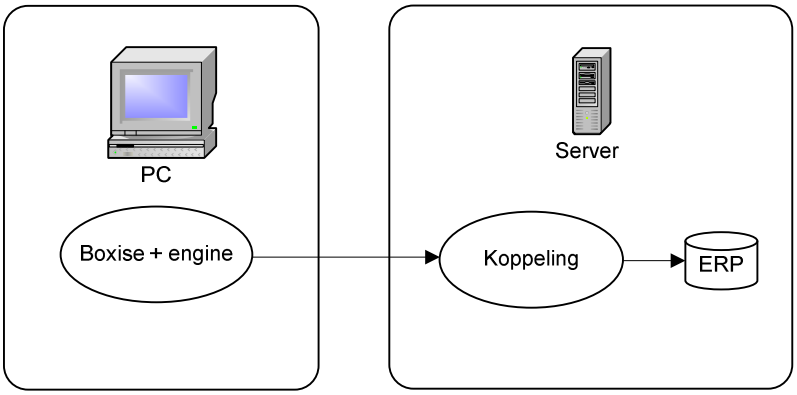
Server, engine en koppeling als webservice
Wederom hetzelfde idee als het vorige punt, maar de engine wordt ook als een webservice beschikbaar gesteld op een server. Hierdoor is dit ook meteen weer beschikbaar voor elke Boxwise cliënt binnen een bedrijf.



Afbeelding 28: Externe engine en koppeling als webservice

Server, koppeling als bestand op server

Het was ook mogelijk om de koppelingen op als een bestand op een server te zetten, die vervolgens via een rechtstreekse verbinding kan worden benaderd.



Afbeelding 29: Interne engine + externe koppeling

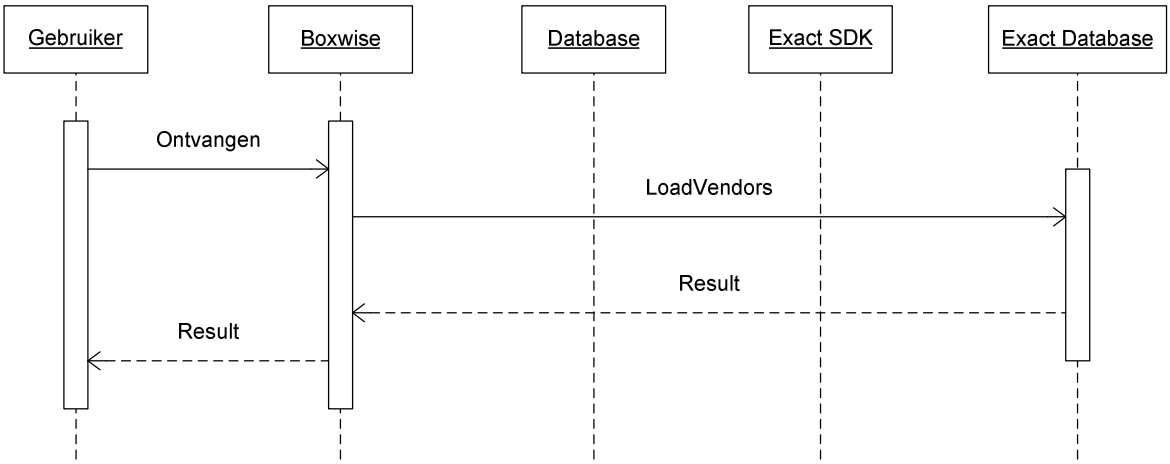
6.2 Gekozen oplossing

Voor deze opdracht waren er meerdere oplossingen, en er is gekozen om de engine in Boxwise zelf te bouwen. De koppelingen worden aparte bestanden die lokaal zullen worden opgeslagen. Boxwise zal dus via de engine een koppeling aan kunnen spreken, om zo gegevens uit een ERP systeem te halen, of er naartoe te sturen.

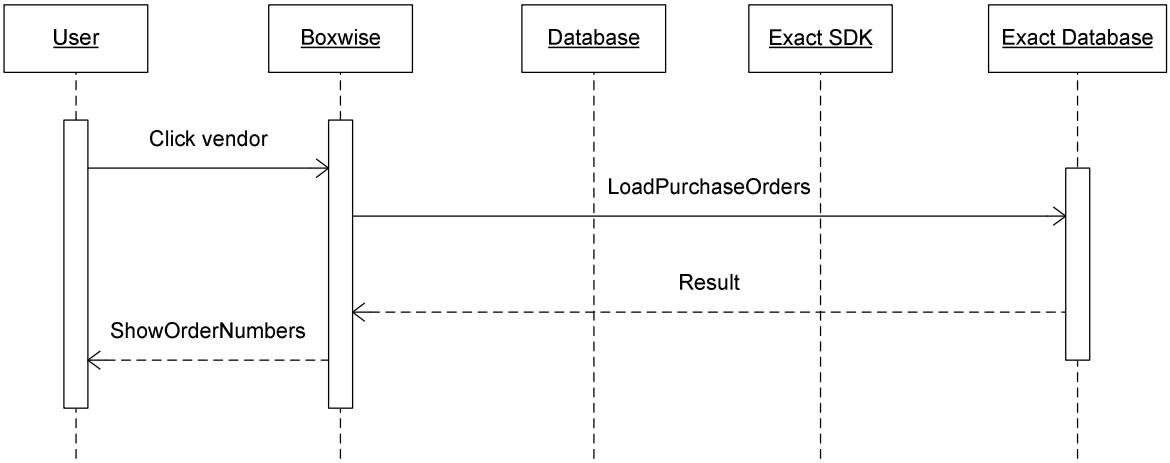
Deze oplossing is gekozen zodat het eenvoudig in BOXwise aan te passen is. De koppelingen zullen lokaal worden geplaatst zodat deze ook eenvoudig door BOXwise te benaderen zijn.

Bijlage A: Stroomdiagrammen informatiestromen van Boxwise

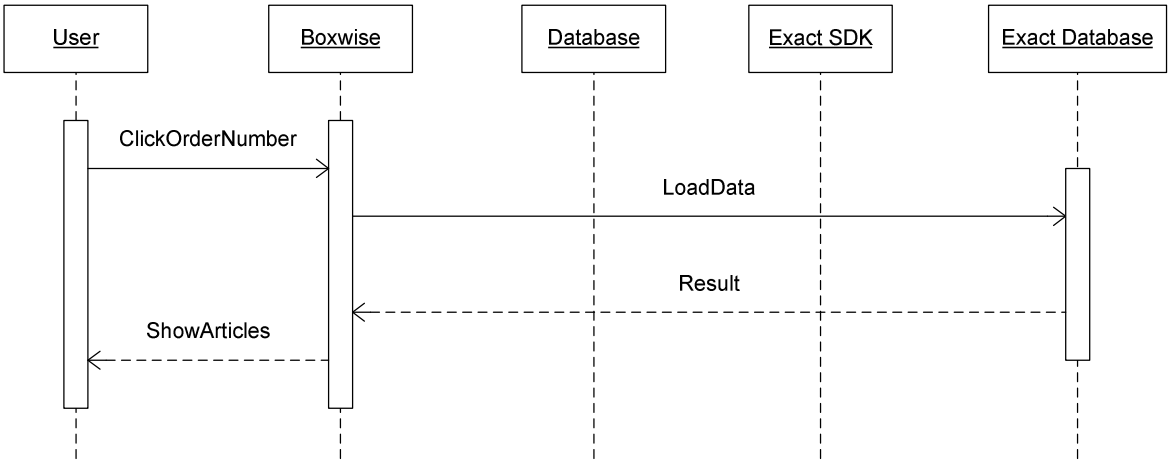
1. Ontvangen goederen | Ophalen leveranciers



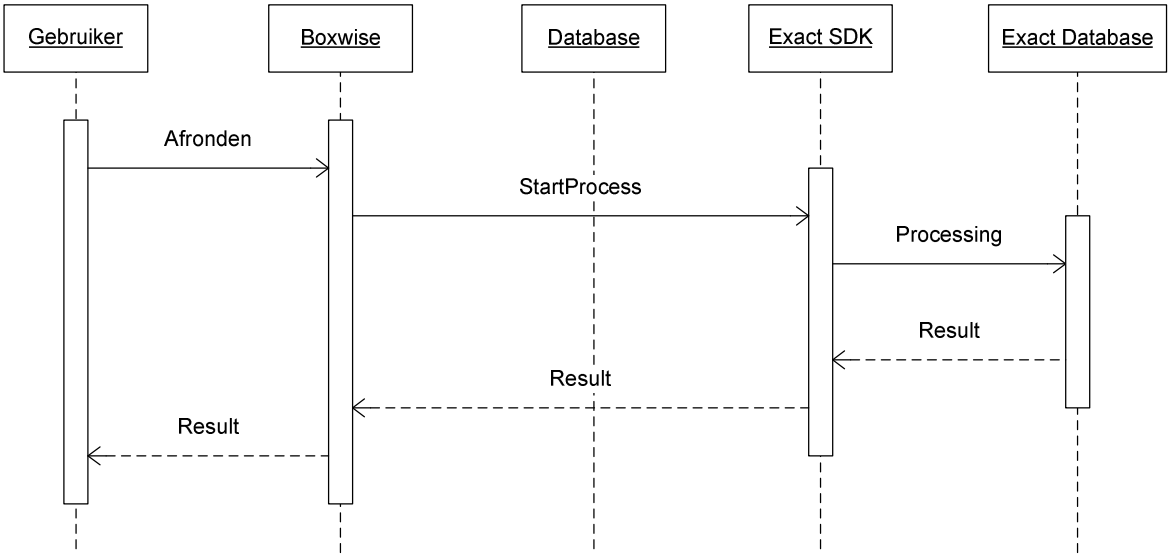
2. Ontvangen goederen | Ophalen orders van leverancier



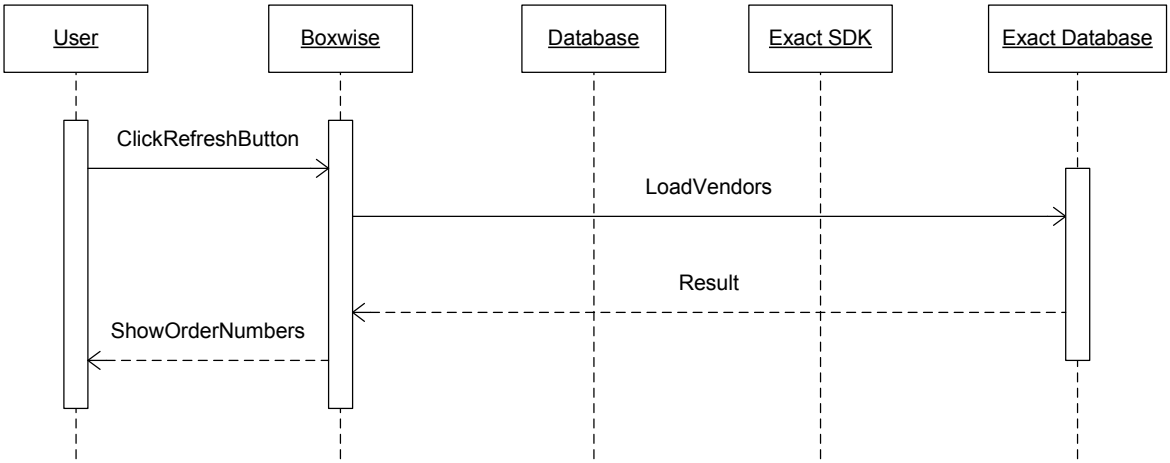
3. Ontvangen goederen | Ophalen artikelen van order



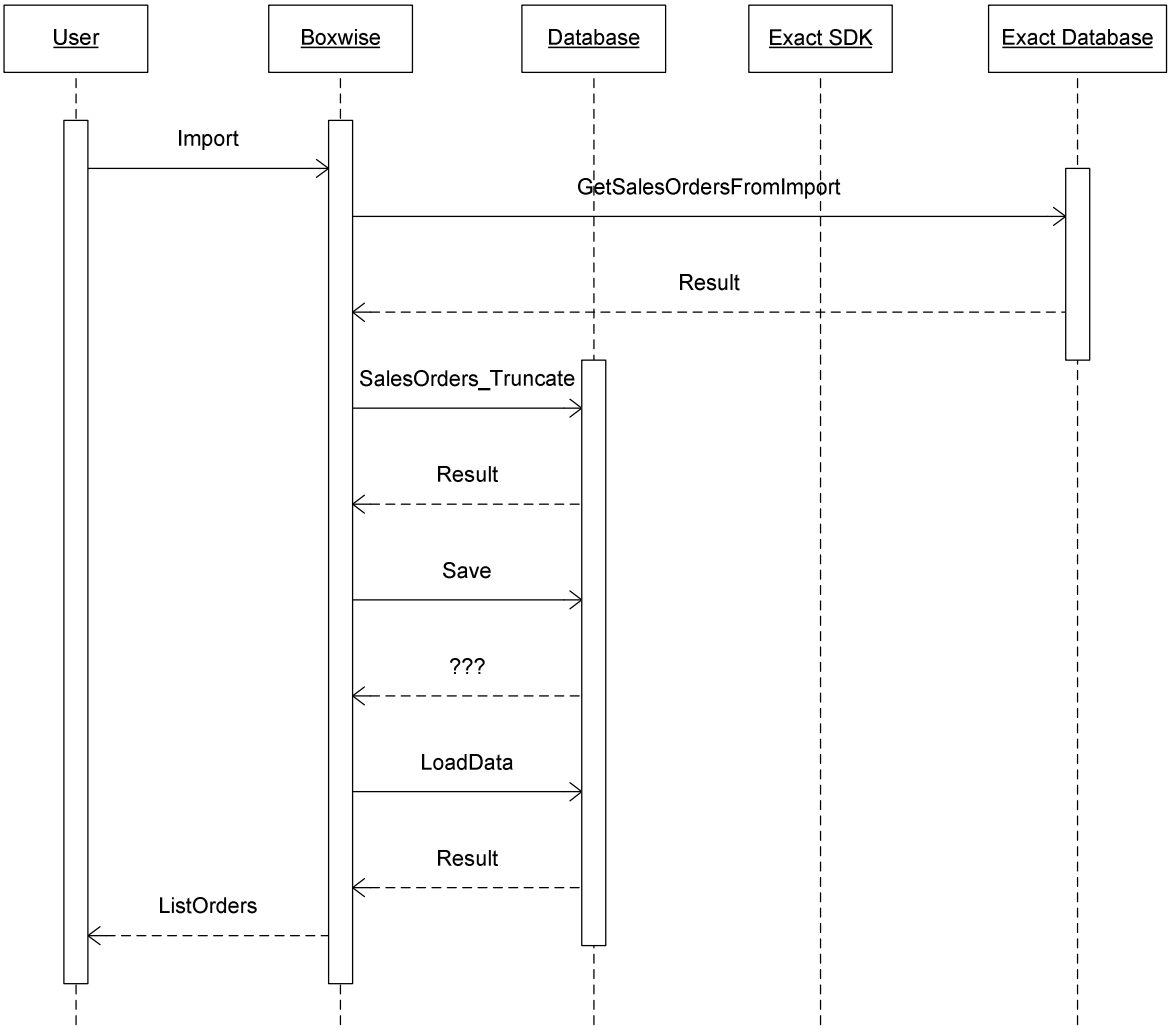
4. Ontvangen goederen | Order verwerken



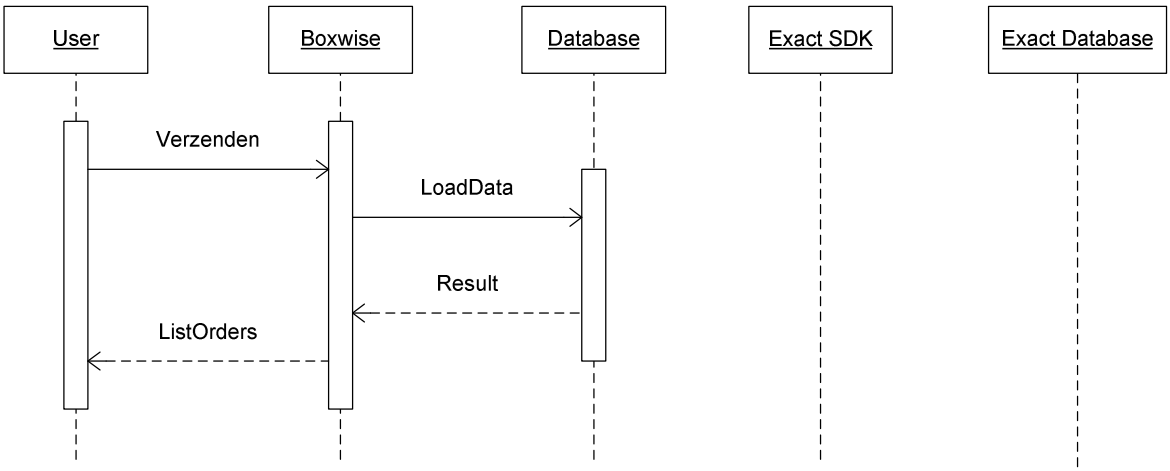
5. Ontvangen goederen | Lijst leveranciers verversen



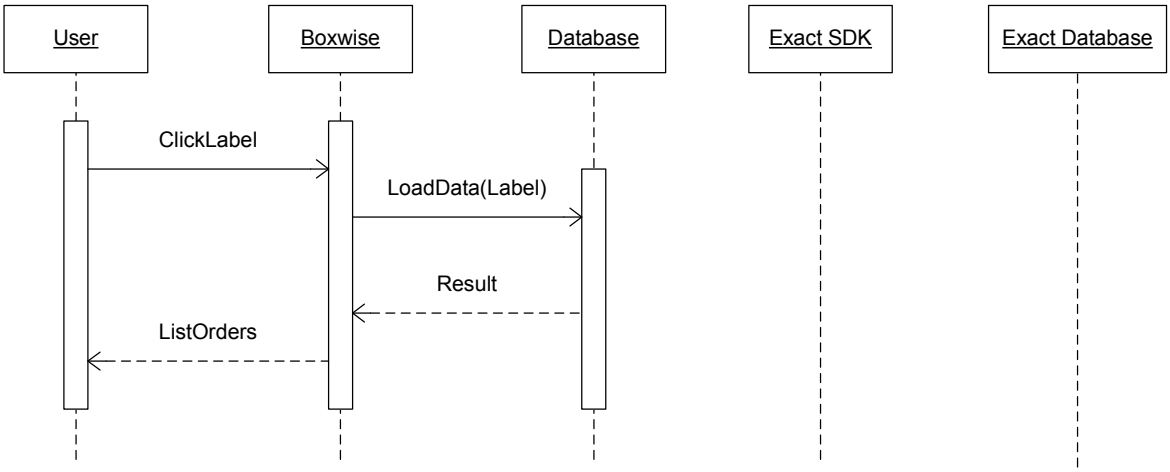
6. Orders verzenden | Import orders (snapshot)



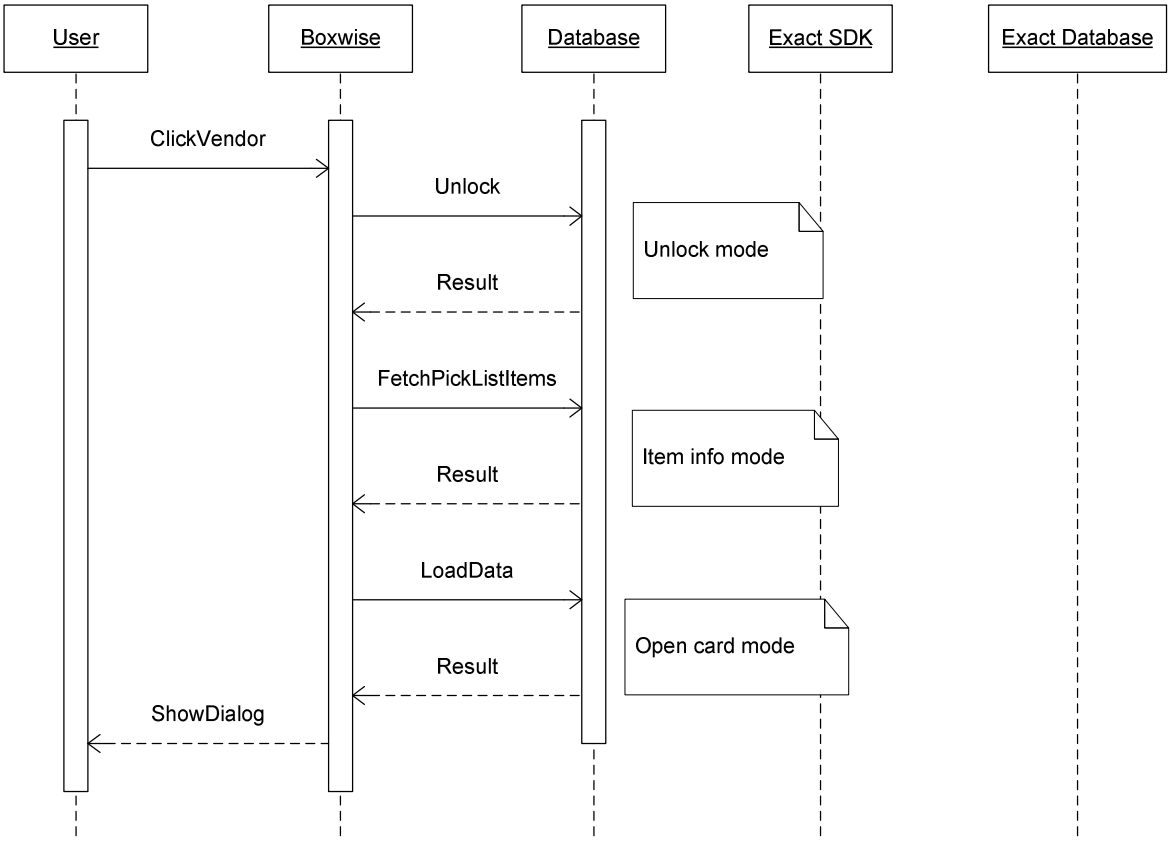
7. Verzenden | Orders weergeven



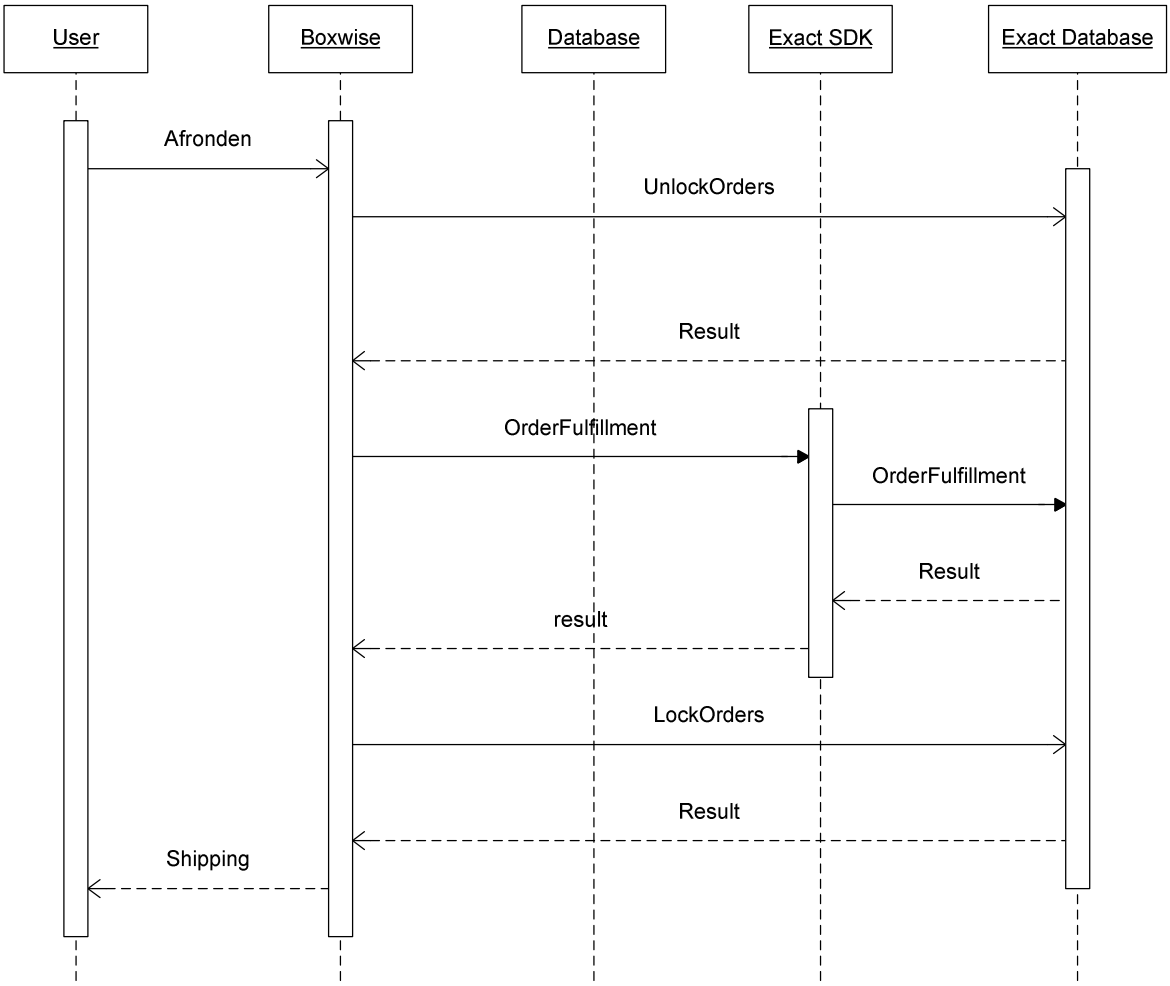
8. Verzenden | Orders weergeven (met een geselecteerde label)



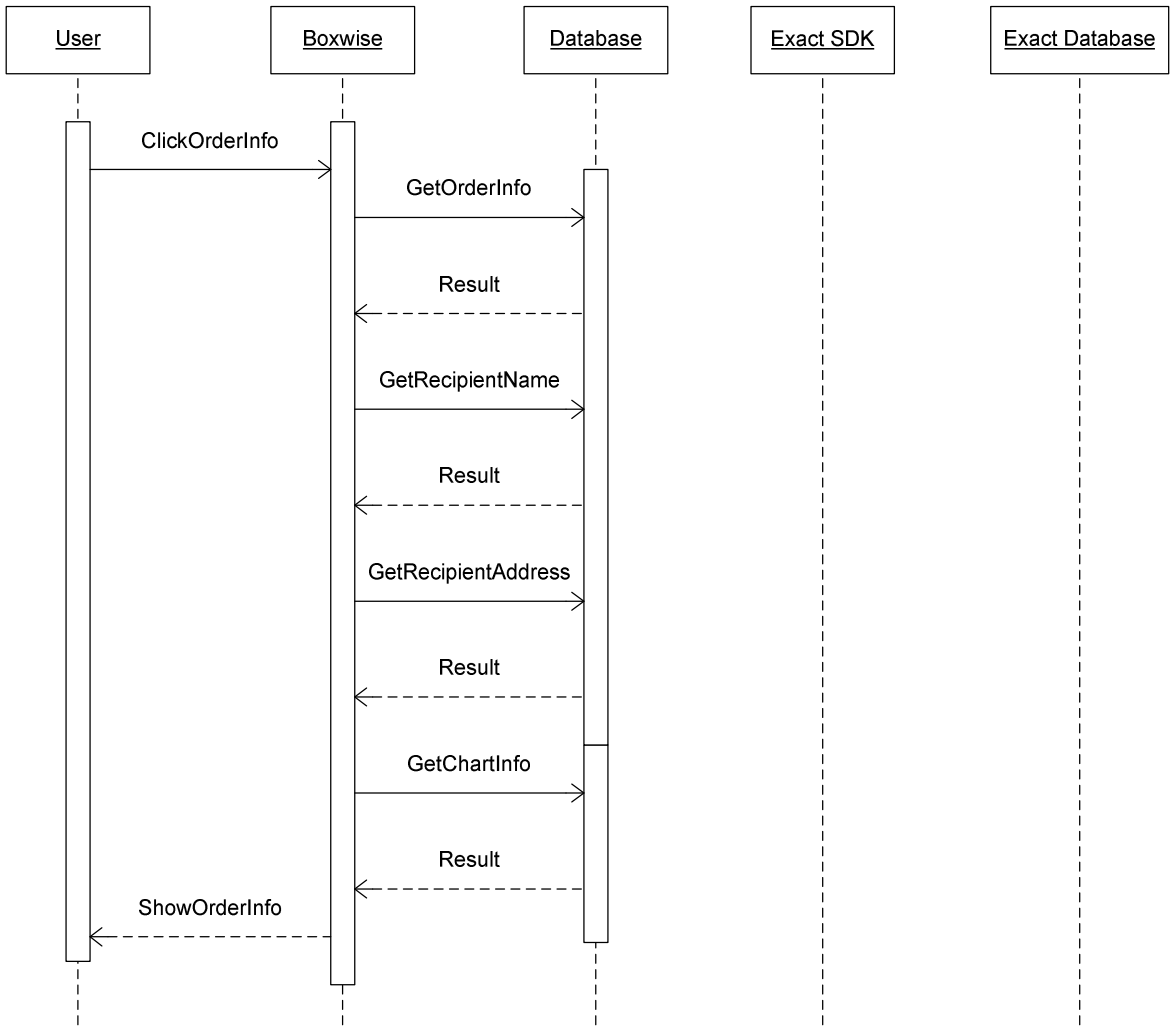
9. Verzenden | Ophalen artikelen van order



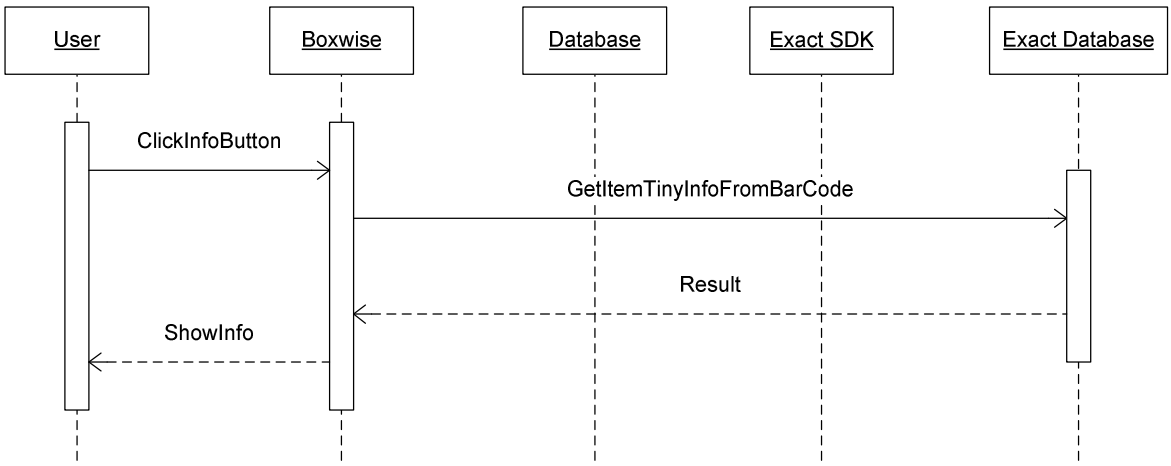
10. Verzenden | Order afronden



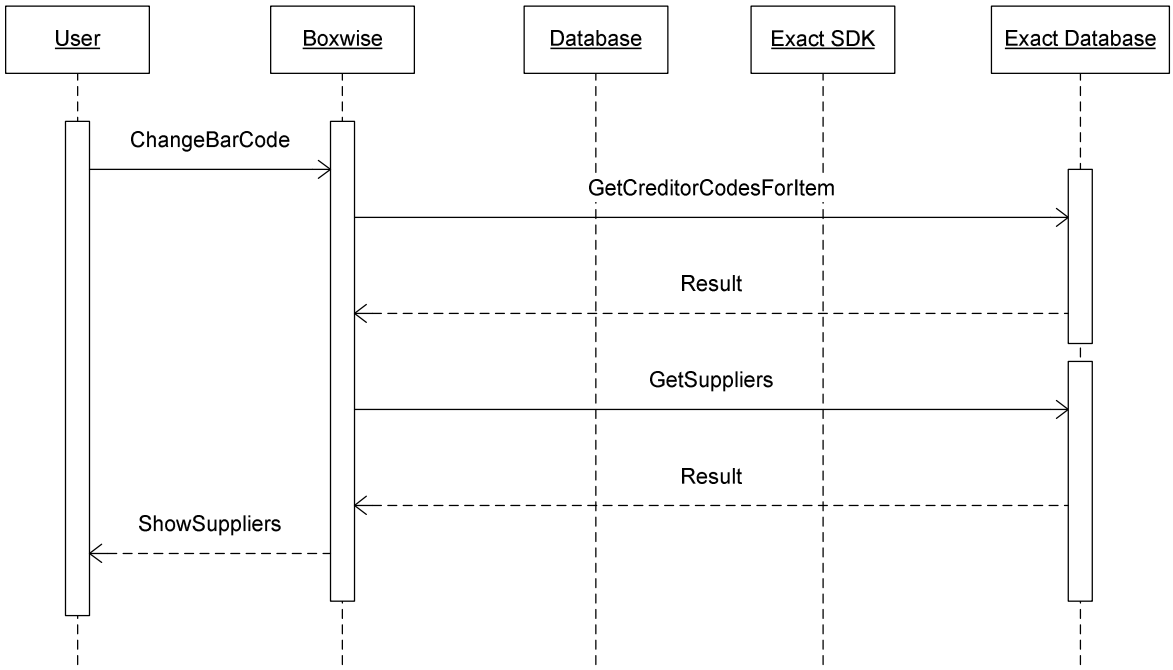
11. Verzenden | Order info



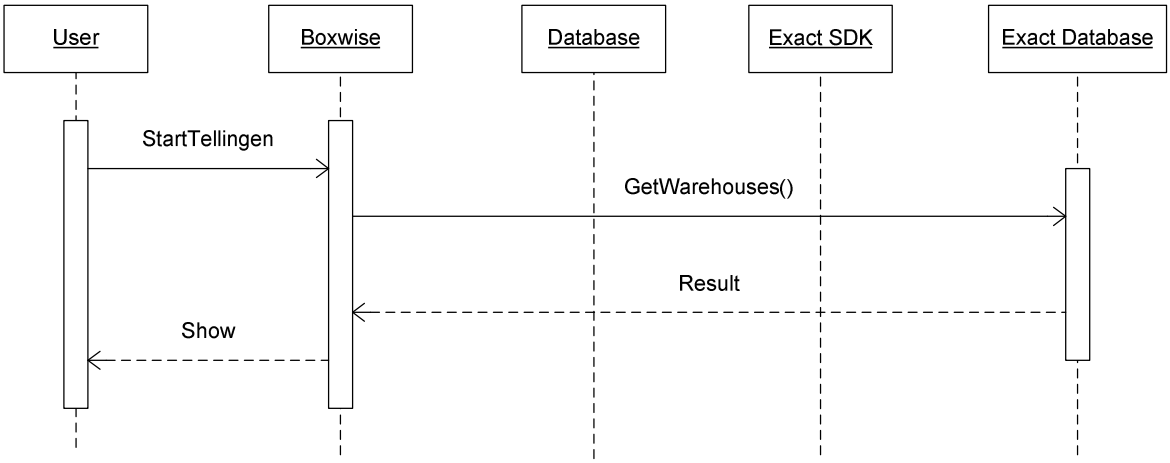
12. Detail informatie van artikel



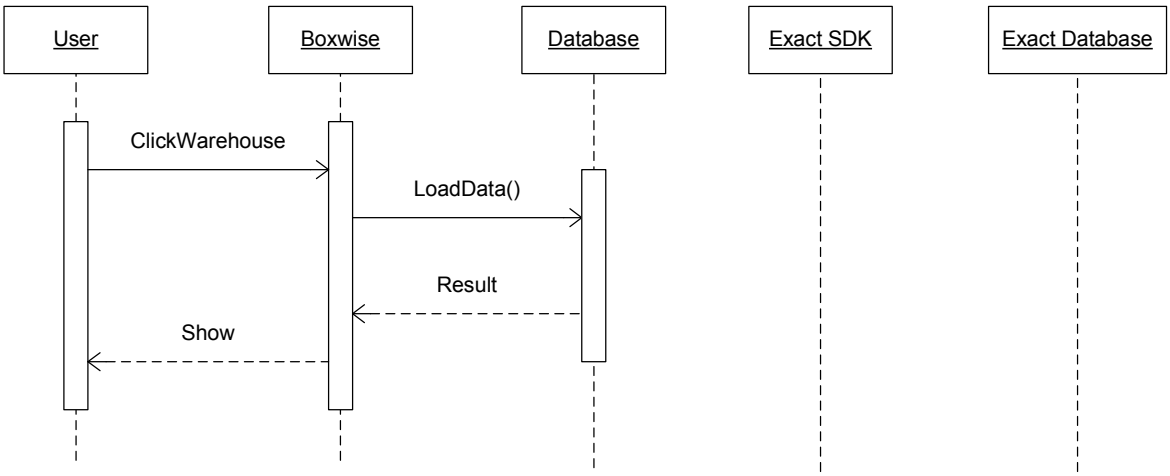
13. Barcode van artikel wijzigen



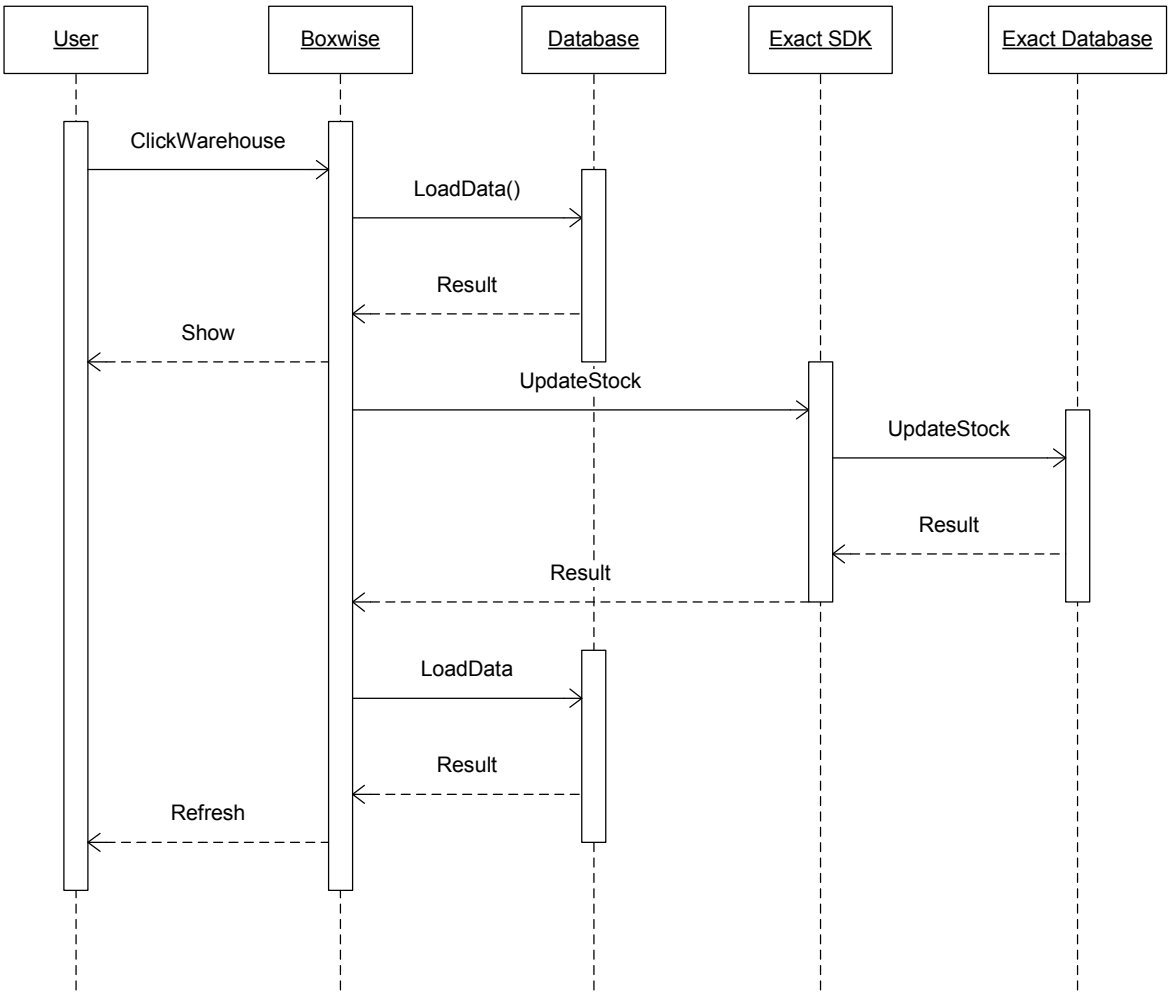
14. Tellingen | Ophalen magazijnen



15. Tellingen | Selecteer magazijn



16. Tellingen | Afronden



Bijlage C: Elaboration Rapport

Elaboration rapport

Auteur: Ray Wijshake

Datum: 11 juni 2010

Versie: Versie 1

Inhoudsopgave

1. Inleiding

Dit rapport is opgesteld om het ontwerp en de werking van de nieuwe abstractielaag in BOXwise toe te lichten. Dit wordt gedaan aan de hand van sequentie- en klassendiagrammen met verklarende teksten.

Allereerst wordt in hoofdstuk 2 het principe van de engine uitgelegd. Daarna wordt er in hoofdstuk 3 dieper op in gegaan door de nieuwe informatiestromen te beschrijven. Hierbij zal er worden verwezen naar Bijlage A waar de desbetreffende sequentie diagrammen te vinden zijn.

2. Werking engine

In dit hoofdstuk zal de werking kort worden toegelicht.

2.1 Provider Pattern

De abstractielaag zal gaan werken m.b.v. het zogenaamde “Provider pattern”. Hiermee is het mogelijk om een decoupled application te maken waarbij er een scheiding wordt gelegd tussen de API en de implementatie. In feite een interface, maar met de mogelijkheid om eenvoudig andere implemenaties te selecteren.

Factory pattern
Creëert een interface voor het te creëren object.

Adapter pattern
Dit maakt het mogelijk om eenvoudig te wisselen tussen meerdere implementaties.

Singleton
Zorgt ervoor dat er één instantie van een object wordt gemaakt.

Facade

De pattern bestaat uit de volgende onderdelen:

- API class
  In dit geval BOXwise

- Provider settings
  Een sectie in de app.config waarin de specifieke concrete providers (koppelingen) worden geregistreerd.

- Abstract Provider
  Definieert het contract (ofwel een interface) voor de implementatie.

- Concrete provider
  Bevat de implementatie van het contract (de koppeling). Overerft van de abstract provider.

2.2 Sequentie diagrammen

Om de informatiestromen in kaart te brengen, zijn er sequentie diagrammen opgesteld. Deze zijn te vinden in Bijlage A. In het volgende hoofdstuk worden deze toegelicht waarbij er naar de desbetreffende diagram wordt verwezen.

2.3 Objecten

Voor het uitwisselen van informatie tussen BOXwise en een koppeling zijn er een aantal objecten opgesteld. Met behulp van deze objecten kan BOXwise bepaalde informatie opvragen of versturen, waarna er verder met de informatie kan worden gewerkt. In de volgende paragrafen zullen de objecten worden toegelicht.

3. Informatiestromen

In dit hoofdstuk worden de informatiestromen toegelicht. In de oude situatie beschikte BOXwise zelf over de intelligentie om een verbinding met Exact te maken, maar het is de bedoeling dat dit via een centraal punt gaat verlopen. Vervolgens kan er door het gebruik van externe koppelingen met meerdere ERP systemen gecommuniceerd worden.

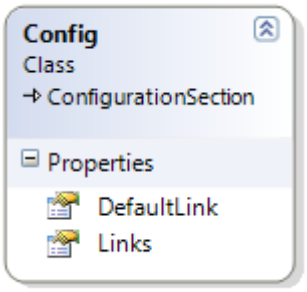
Voor het beschrijven van de informatiestromen worden sequence diagrams gebruikt. Bij elk informatiestroom wordt naar de desbetreffende diagram verwezen. Tevens worden de gebruikte objecten beschreven.

3.1 Initialiseren koppelingen

Initialiseren Sequence diagram: 1

Werking:
- BOXwise start op
- BOXwise roept de abstractielaag aan om de standaard koppeling te laden
- In de abstractielaag wordt de standaard koppeling geselecteerd m.b.v. de klasse Config.
- Config leest de specifieke sectie in de app.config voor de koppelingen, en slaat de standaard koppeling op in de property “DefaultLink”.
- Vervolgens wordt om een object te maken van de class binnen de assembly, die overerft van de LinkProvider.
- Als het creëren van het object succesvol is verlopen, wordt er een test uitgevoerd.

Gebruikte objecten:
- *Config*
  Verantwoordelijk voor het inlezen van het configuratiebestand, om vervolgens een standaard koppeling op te slaan.



Overerft van:
- *ConfigurationSection*
  Representeert een sectie van het app.config bestand (in dit geval de sectie met de koppelingen)

Input (parameters):
- Geen

Output:
- Geen

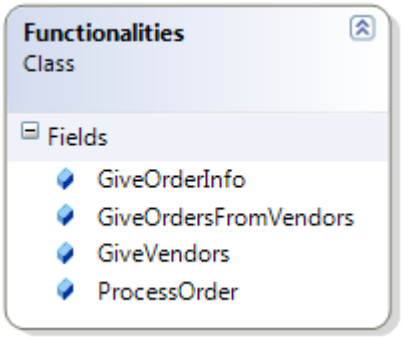
Controleren functionaliteiten Sequence diagram: 2

Werking:

- Na het initialiseren van de koppeling wordt er aangegeven dat er moet worden gecontroleerd welke functionaliteiten mogelijk zijn met het desbetreffende ERP systeem.
- De koppeling vult het object Functionalities, die als het ware functioneert als checklist.
- De koppeling stuurt het object terug
- BOXwise loopt door het object en schakelt functionaliteiten uit die niet worden ondersteund door het ERP systeem.

Gebruikte objecten:

- *Functionalities*
  Object bestaande uit booleans waarmee wordt aangegeven welke functionaliteiten worden ondersteund door het ERP systeem.



Input (parameters):

- Geen

Output:

- Functionalities object

3.2 Ontvangen goederen

Ontvangen goederen | Ophalen leveranciers Sequence diagram: 3

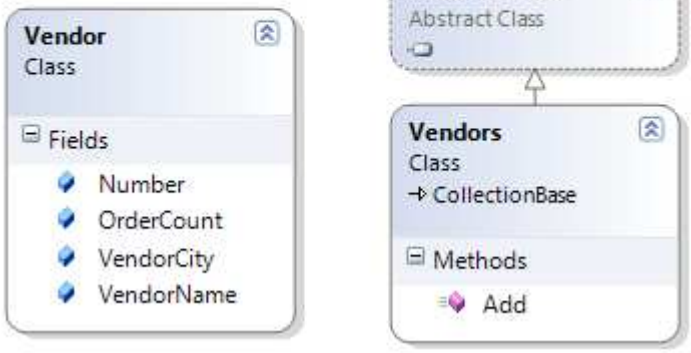
Werking:

- BOXwise roept via de abstractielaag de methode aan om leveranciers op te halen die een order hebben geleverd.
- Voor elke leverancier wordt er een Vendor object opgesteld, waarbij de desbetreffende fields worden gevuld.
- Nadat een Vendor object is gevuld, wordt deze toegevoegd aan het Vendors object. Dit is een verzamelobject die alle leveranciers bevat.
- Het Vendors object wordt naar BOXwise gestuurd, waarna BOXwise deze uitleest om de inhoud te tonen aan de gebruiker.

Gebruikte objecten:

- *Vendor*
  Bevat leverancier specifieke gegevens

- *Vendors*
  Verzamelobject voor leveranciers. Bevat geen of meerdere Vendor objecten.



Input (parameters):

- Geen

Output:

- Vendors object

Ontvangen goederen | Ophalen orders var een leverancier Sequence diagram: 4

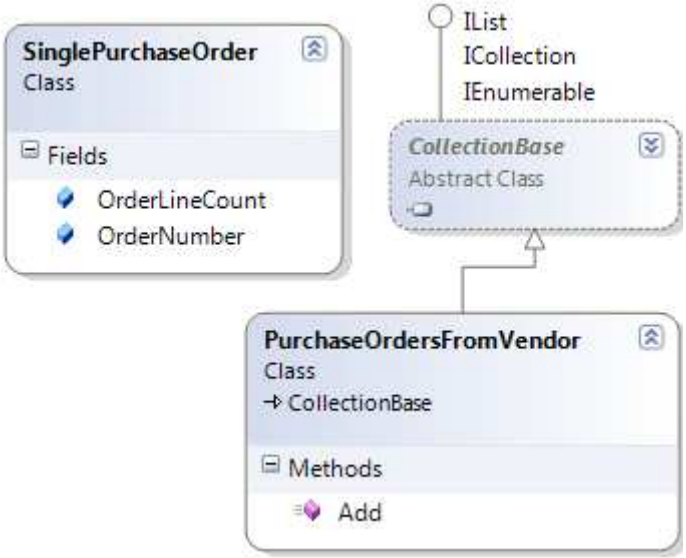
Werking:

- BOXwise roept via de abstractielaag de methode aan om de orders van een specifieke leverancier op te halen.
- De koppeling haalt de orders van de specifieke leverancier op.
- Als er één of meerdere orders zijn gevonden, wordt er voor elke order een SinglePurchaseOrder object opgesteld.
- Nadat een SinglePurchaseOrder is opgesteld, wordt deze toegevoegd aan PurchaseOrdersFromVendor object. Dit is een verzamelobject dat overerft van CollectionBase.
- Het PurchaseOrdersFromVendor object wordt naar BOXwise gestuurd, waarna BOXwise deze uitleest om de inhoud te tonen aan de gebruiker.

Gebruikte objecten:

- *SinglePurchaseOrder*
  Bevat gegevens van een enkele order.

- *PurchaseOrdersFromVendor*
  Verzamelobject die de orders van de leveranciers. Bevat geen of meerdere SinglePurchaseOrder objecten.



Input (parameters):

- String vendorNumber

Output:

- PurchaseOrdersFromVendor object

Ontvangen goederen | Ophalen artikelen van de geselecteerde orders

Sequence diagram: 5

Werking:
Voor elke order wordt het volgende uitgevoerd (foreach loop op de order nummers):

- BOXwise roept via de abstractielaag de methode aan om de specifieke gegevens van de geselecteerde order op te halen.
- De koppeling haalt de specifieke gegevens van een order op, waarna het PurchaseOrders object wordt gevuld.
- Het PurchaseOrders object wordt naar BOXwiser gerstuurd, die het vervolgens verwerkt.
- BOXwise roept via de abstractielaag de methode aan om de orderregels van de geselecteerde orders op te halen.
- De koppeling haalt de orderregels op aan de hand van de orders nummers die het van BOXwise krijgt.
- Voor elke orderregel wordt er een PurchaseOrderLine gevuld.
- Nadat een PurchaseOrderLine is opgesteld, wordt deze toegevoegd aan het PurchaseOrderLinesCollection object. Dit is een verzamelobject dat overerft van CollectionBase.
- Het PurchaseOrderLinesCollection object wordt naar BOXwise gestuurd, die het vervolgens verwerkt.

Gebruikte objecten:

- *PurchaseOrders*
  Bevat de specifieke gegevens van een order.

- *SinglePurchaseOrderLine*
  Bevat een enkele orderregel.

- *PurchaseOrderLinesCollection*
  Verzamelobject dat alle orderregels van een specifieke order bevat.
  Bevat één of meerdere OrderLines objecten.
  Overerft van CollectionBase.

Input (parameters):

- String purchaseOrderNumber

Output:

- PurchaseOrders object
- PurchaseOrderLinesCollection object

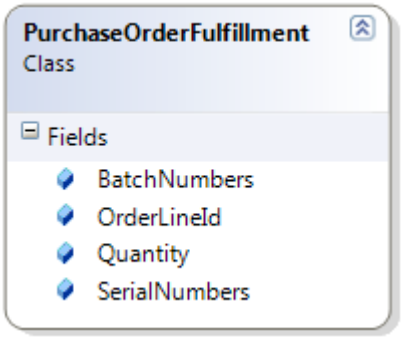
Ontvangen goederen | Verwerken orders	Sequence diagram: 6

Werking:

- Voor elke order dat verwerkt moet worden stelt BOXwise het PurchaseOrderFulfillment object op.
- BOXwise stuurt het PurchaseOrderFulfillment object via de abstractielaag naar de koppeling.
- De koppeling verwerkt vervolgens de desbetreffende order.

Gebruikte objecten:

- *PurchaseOrderFulfillment*
  Bevat de gegevens van een order om deze te verwerken in het ERP systeem.



Input (parameters):

- PurchaseOrderFulfillment

Output:

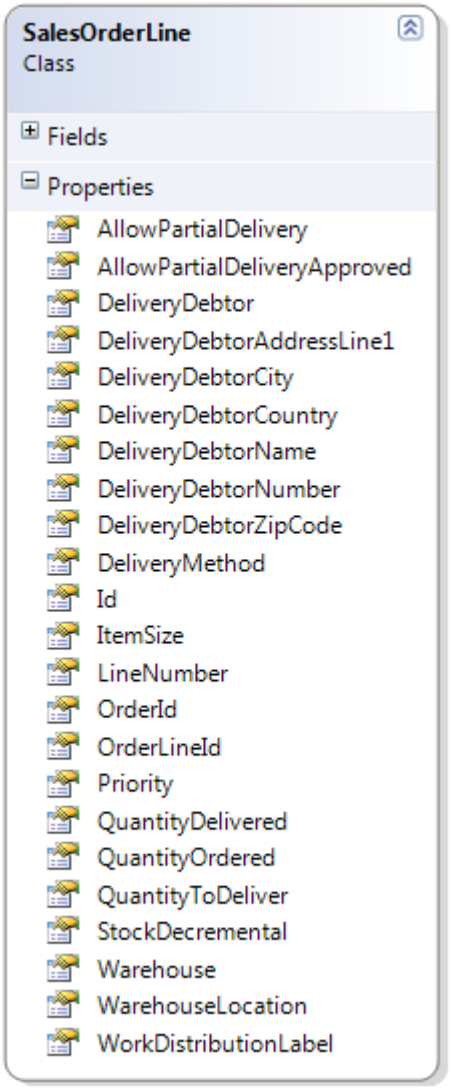
- Boolean succes / failed

3.3 Verzenden goederen

Verzenden goederen | Snapshot Sequence diagram: 7

Werking:
- Er wordt een tweede thread gestart.
- BOXwise voert een truncate statement uit op de tabellen AutoSaves, PickSelection en ImportSalesOrders.
- BOXwise roept via de abstractielaag de methode aan om alle sales orders op te halen.
- De gegevens van elk gevonden orderregel wordt opgeslagen in een OrderLine.
- Elk gevonden orderregel wordt via een delegate in BOXwise meteen verwerkt.
- Nadat het importeren is voltooid, wordt alle data vanuit de BOXwise database geladen.

Gebruikte objecten:
- *SalesOrderLine*
  Bevat alle gegevens van een orderregel.



Input (parameters):
- Delegate

Output:
- OrderLine object

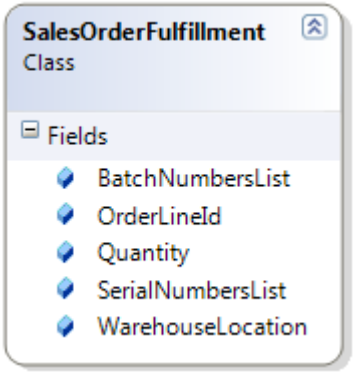
Verzenden goederen | Order verwerken Sequence diagram: 8

Werking:

- Er wordt een tweede thread gestart om een order te verwerken.
- Per product wordt er gecontroleerd of er voldoende op voorraad is.
- Als er voldoende op voorraad is, dan worden alle gegevens van een order verzameld.
- De order wordt verwerkt door elk order regel apart naar de koppeling te sturen, die het vervolgens zal verwerken.

Gebruikte objecten:

- *SalesOrderFulfillment*
  Bevat alle gegevens van een enkele orderregel.



Input (parameters):

- SalesOrderFulfillment

Output:

- Geen

3.4 Tellingen

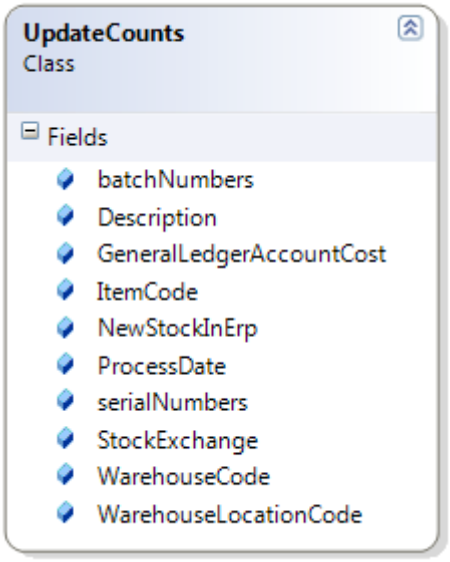
Tellingen | Afronden Sequence diagram: 9

Werking:

- Data wordt geladen van de tabel ... waarna er berekeningen worden uitgevoerd.
- De gebruiker geeft input (omschrijving).
- Vervolgens wordt er een tweede thread gestart waarin de gegevens worden verzameld om deze te verwerken in het ERP systeem.
- Het UpdateCounts object wordt opgesteld, waarna deze wordt verstuurd naar de koppeling om de resultaten van de tellingen te verwerken.

Gebruikte objecten:

- *UpdateCounts*
  Bevat alle resultaten van de tellingen die verwerkt moeten worden in het ERP systeem.



Input (parameters):

- UpdateCounts object

Output:

- Int erpEntryNumber

3.5 Algemene functies

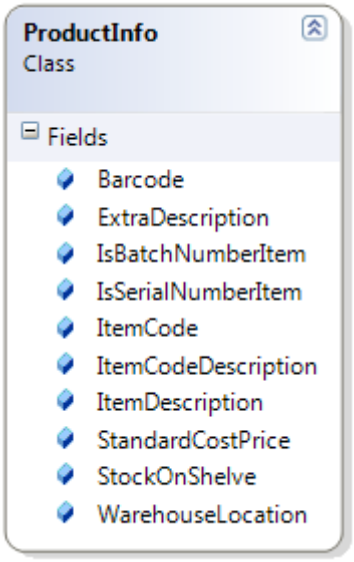
Detail informatie artikel Sequence diagram: 10

Werking:

- BOXwise roept via de abstractielaag de methode aan om de detail informatie van een artikel op te halen. Hierbij wordt het object ProductInfo meegestuurd (via een “out” reference).
- In de koppeling wordt het object ProductInfo gevuld en er wordt aangegeven dat het ophalen is gelukt (stuurt een boolean terug). Indien het is mislukt, wordt er een false terug gestuurd.
- Indien het ophalen van de informatie is gelukt, wordt de informatie door BOXwise verwerkt.

Gebruikte objecten:

- ProductInfo
  Bevat alle gegevens van een artikel.



Input (parameters):

- String itemCode
- String warehouseCode
- Out ProductInfo

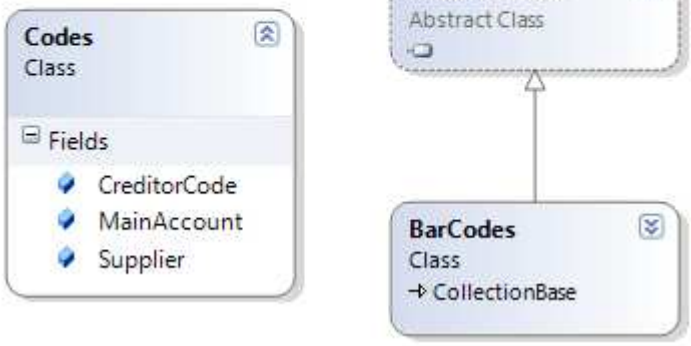
Output:

- Boolean success / failed

Update barcode Sequence diagram: 11

Werking:
- BOXwise roept via de abstractielaag de methode aan om alle barcodes van een product op te halen.
- De koppeling vult het object Codes voor elk gevonden barcode.
- Elk Codes object wordt vervolgens opgeslagen in het object Barcodes.
- De koppeling stuurt het object BarCodes terug waarna BOXwise deze verwerkt.

Gebruikte objecten:
- Codes
  Object voor het opslaan van een enkele barcode van een product.
- BarCodes
  Verzamelobject voor de barcodes.
  Bevat geen of meerdere Codes objecten.
  Overerft van CollectionBase.



Input (parameters):
- String itemCode

Output:
- BarCodes object

Order info Sequence diagram: 12

Werking:
- BOXwise roept via de abstractielaag de methode aan om de naam van de klant op te halen.
- De koppeling stuurt de naam als String terug.
- BOXwise roept via de abstractielaag de methode aan om het adres van de klant op te halen.
- De koppeling stuurt het adres als String terug.

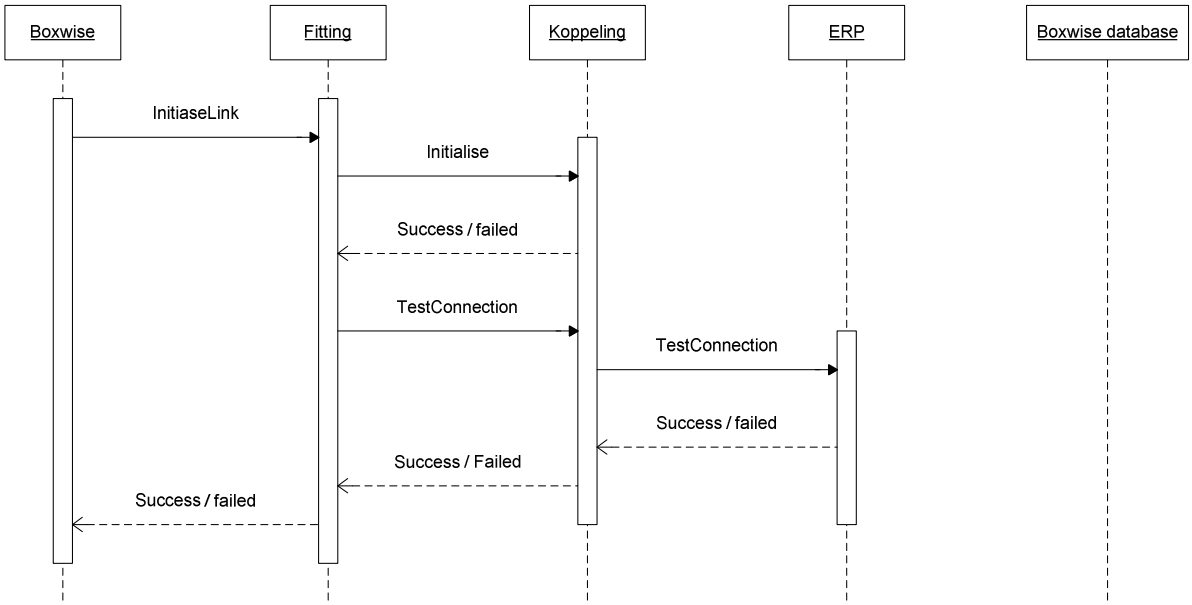
Gebruikte objecten:
- Geen.

Input (parameters):
- String deliveryDebtor

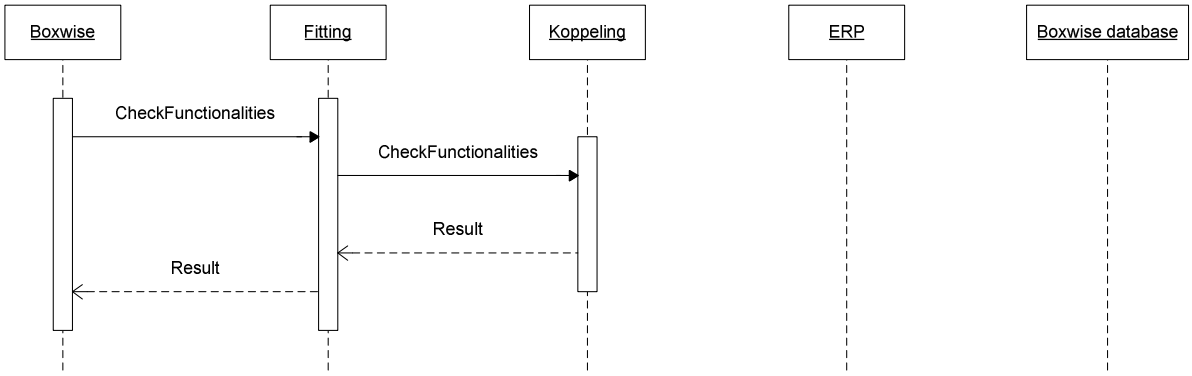
Output:
- String recipientName
- String recipientAddress

Bijlage A: Sequentie diagrammen

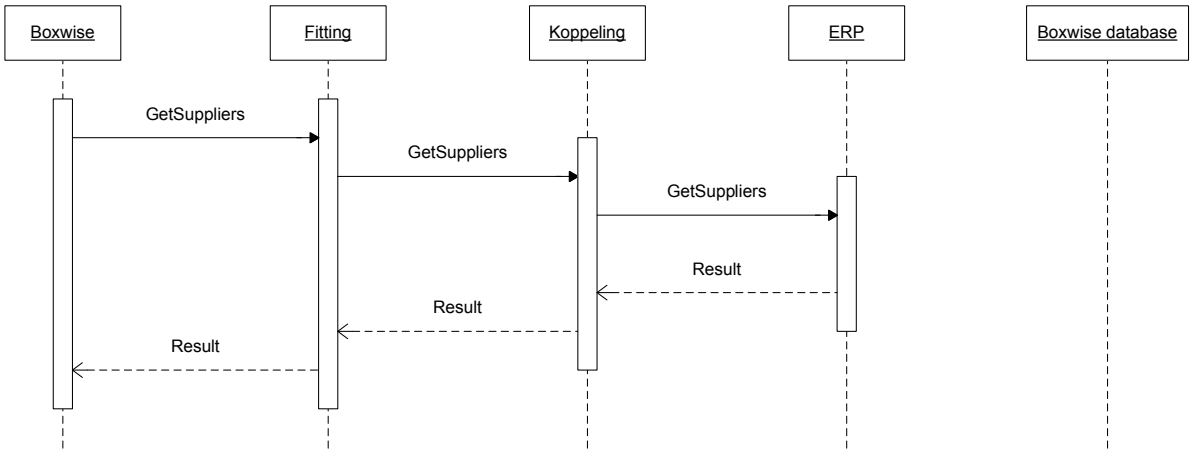
1. Initialiseren koppeling



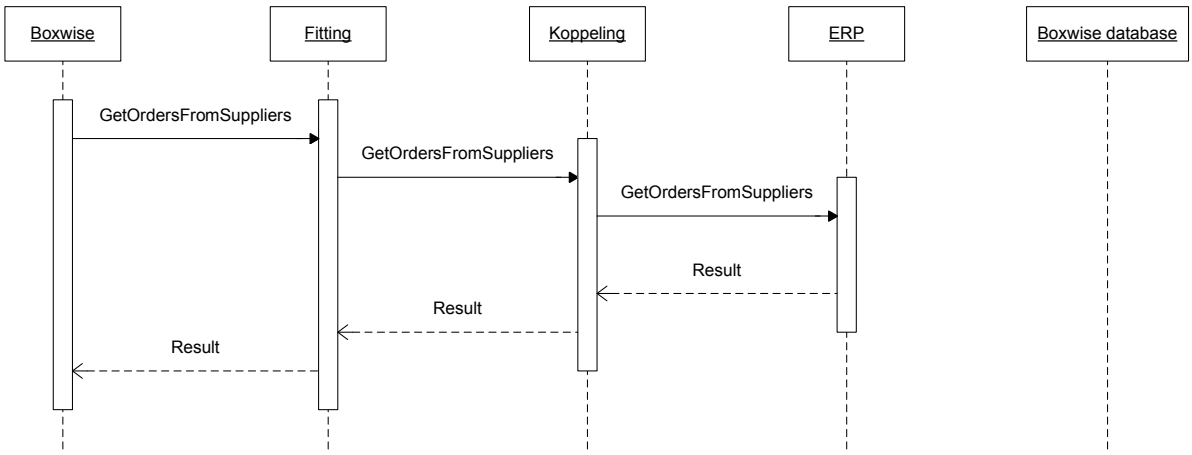
2. Controleren functionaliteiten



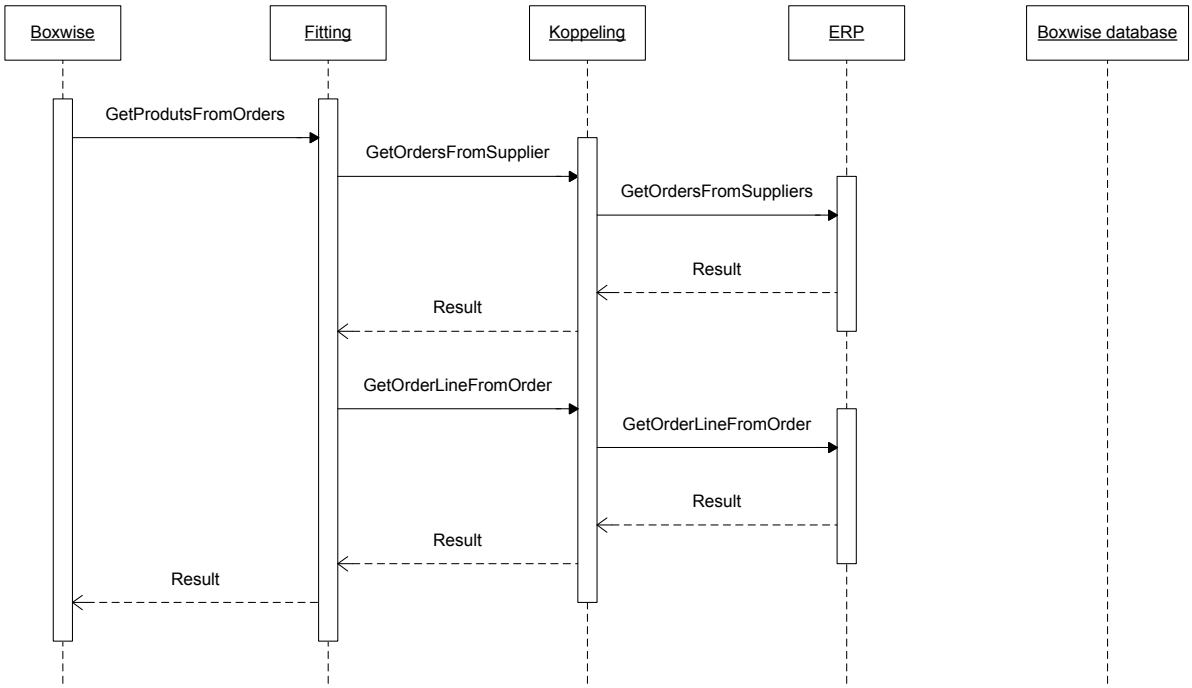
3. Ontvangen goederen | Ophalen leveranciers



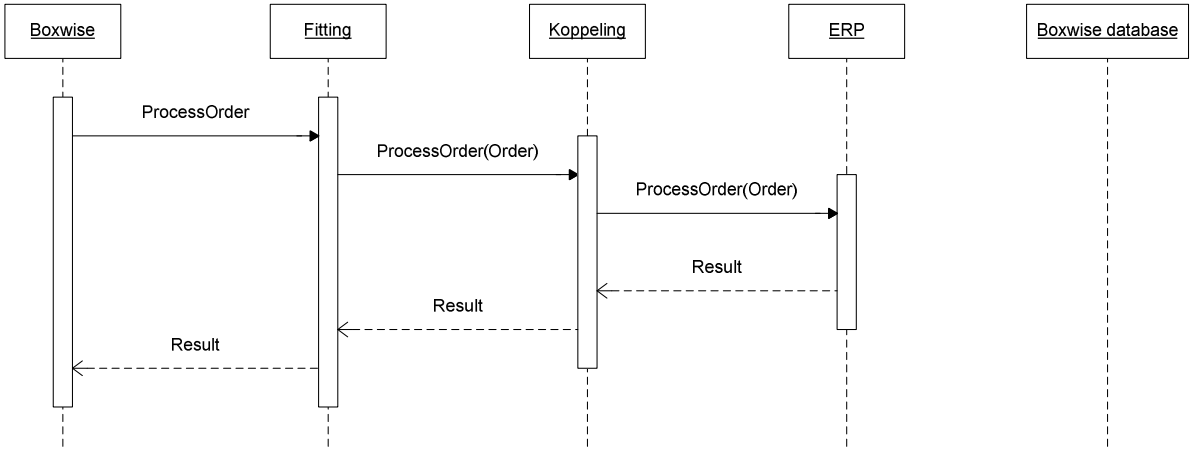
4. Ontvangen goederen | Ophalen orders van leveranciers



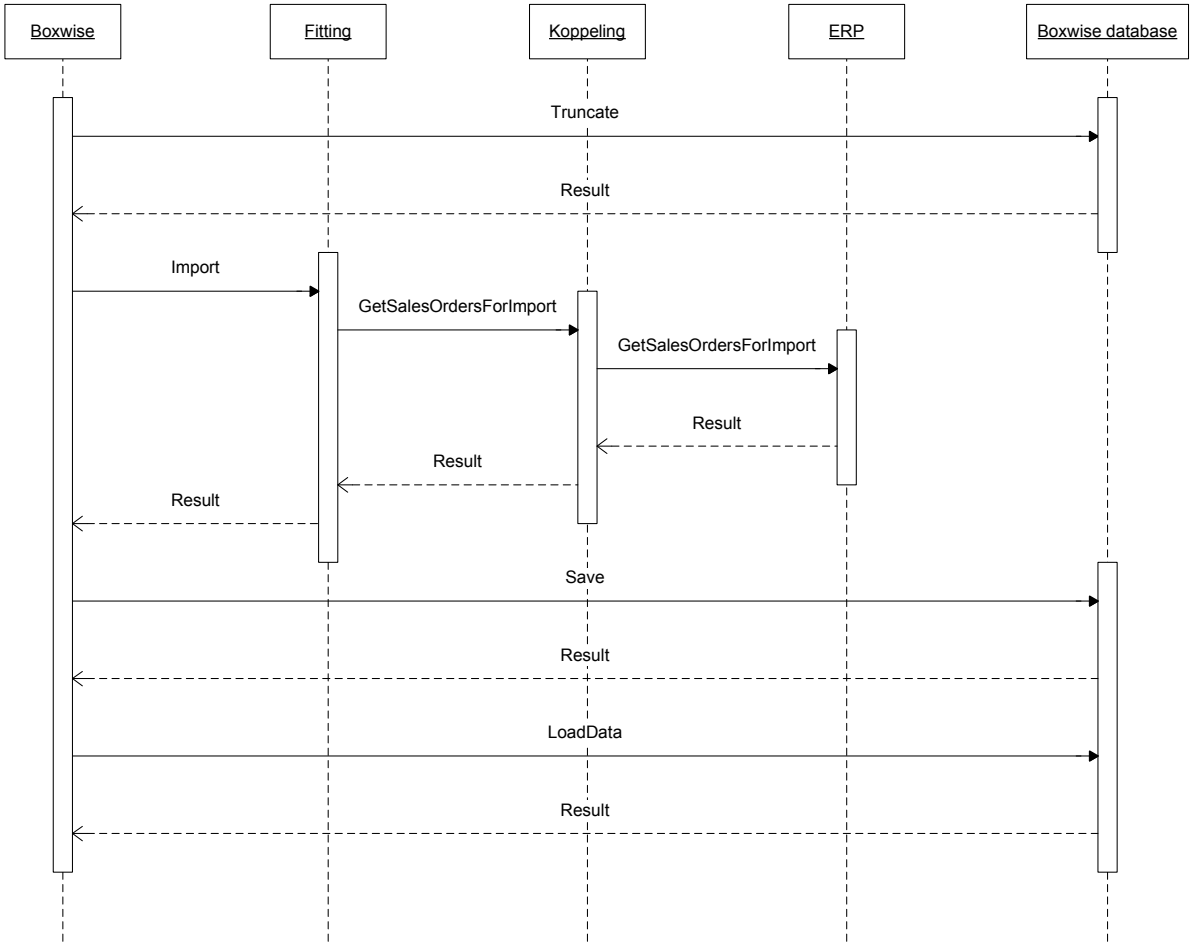
5. Ontvangen goederen | Ophalen artikelen van orders



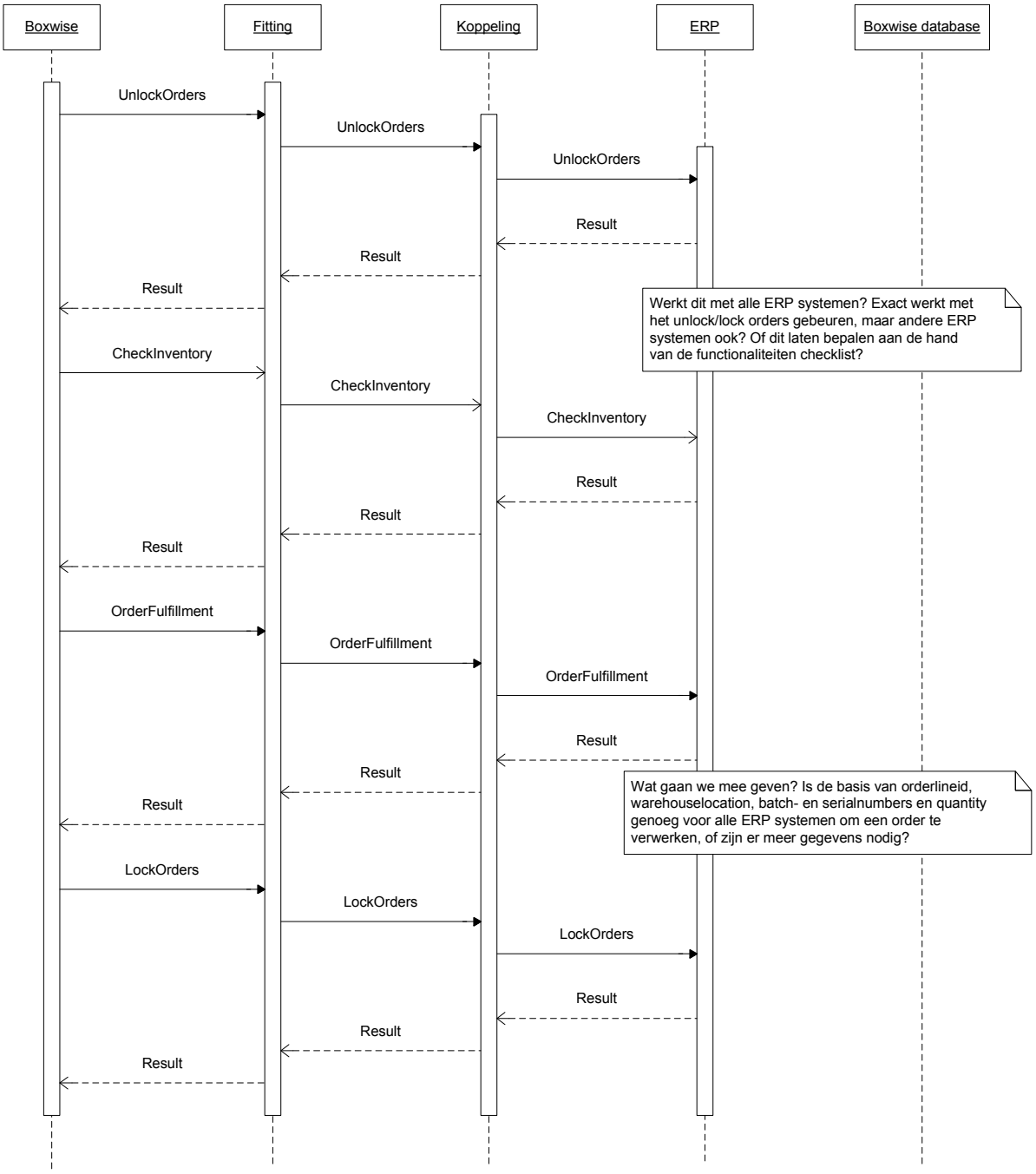
6. Ontvangen goederen | Order verwerken



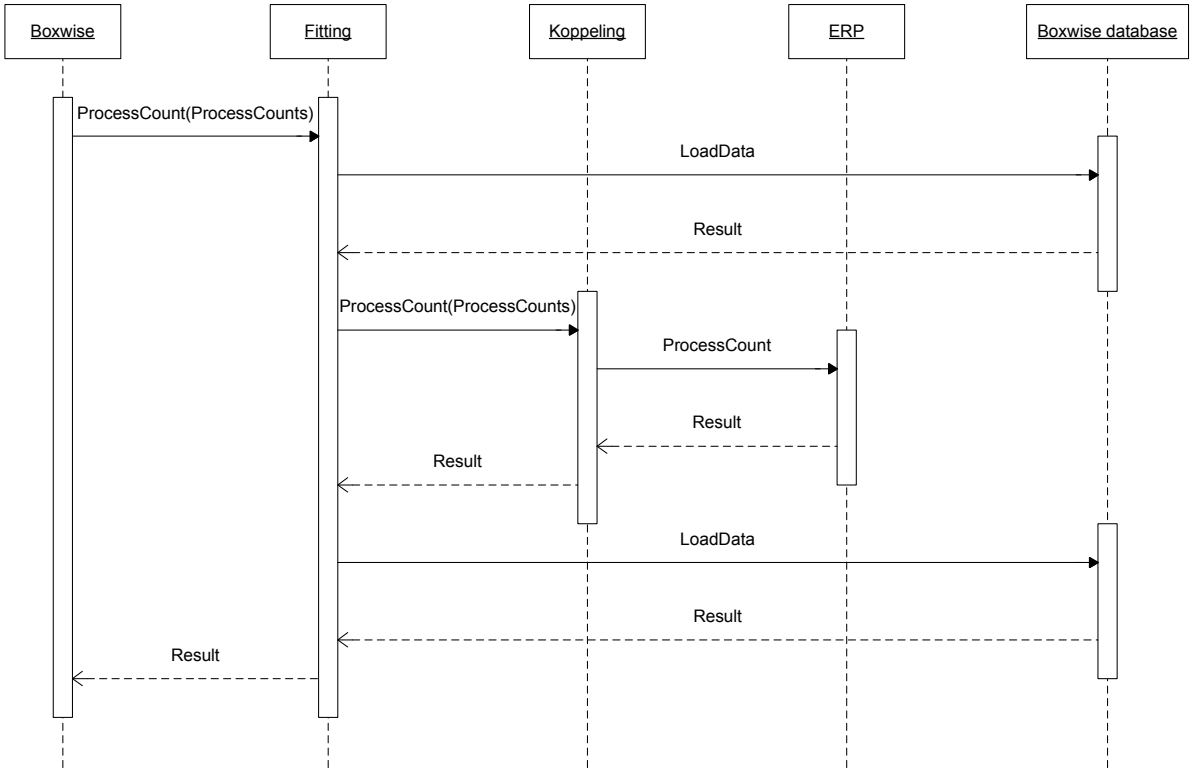
7. Salesorders importeren



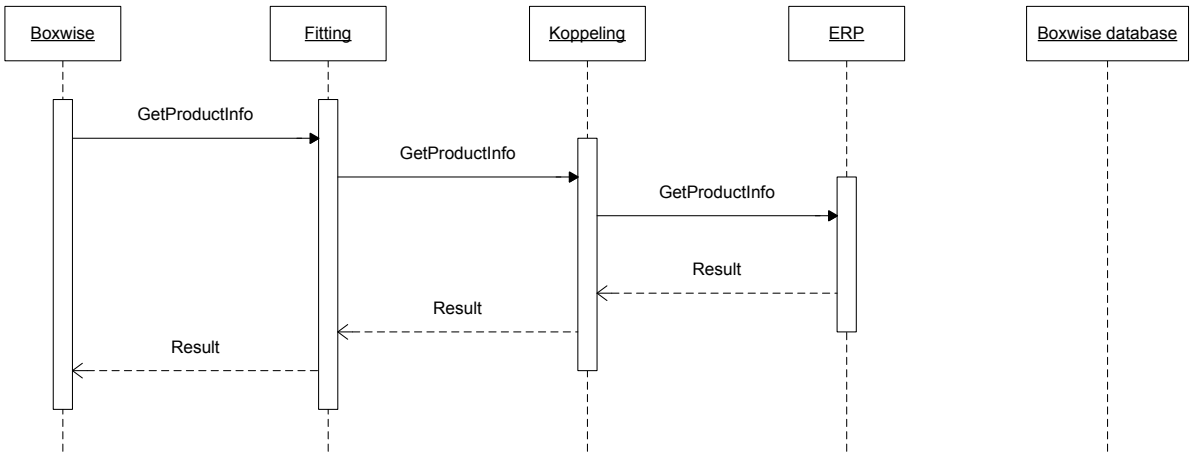
8. Salesorder verwerken



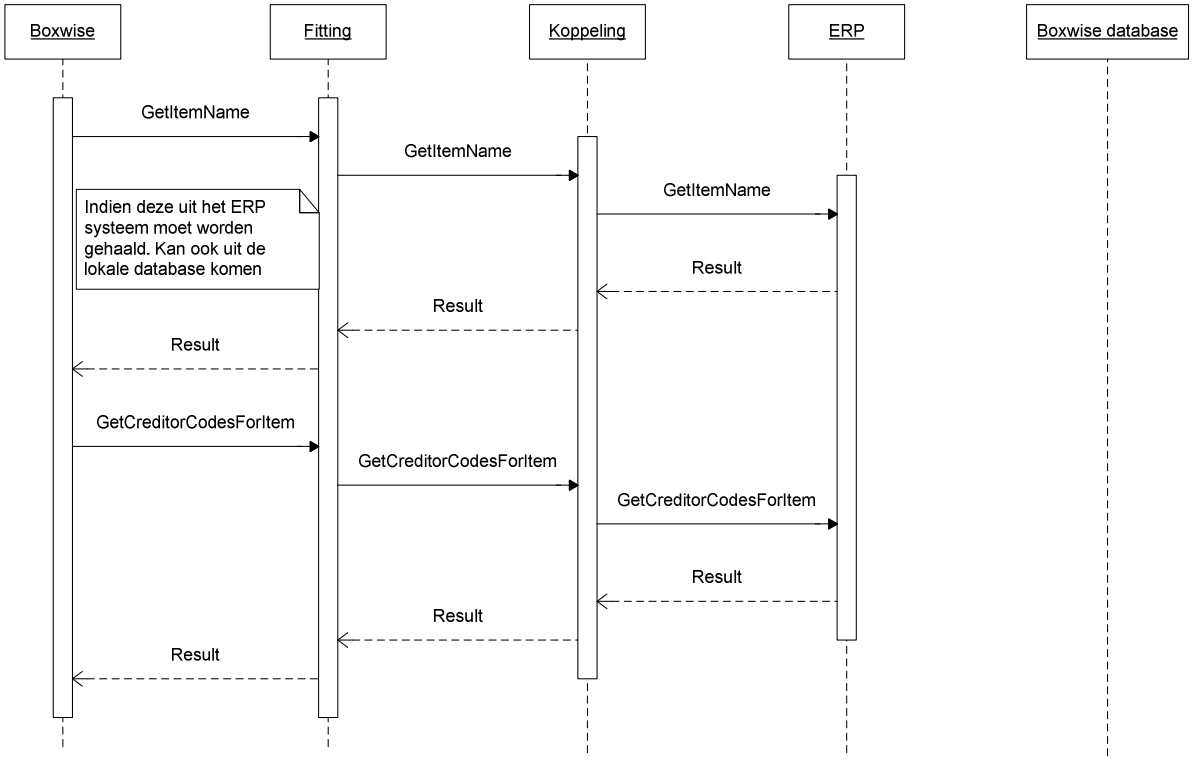
9. Afronden tellingen



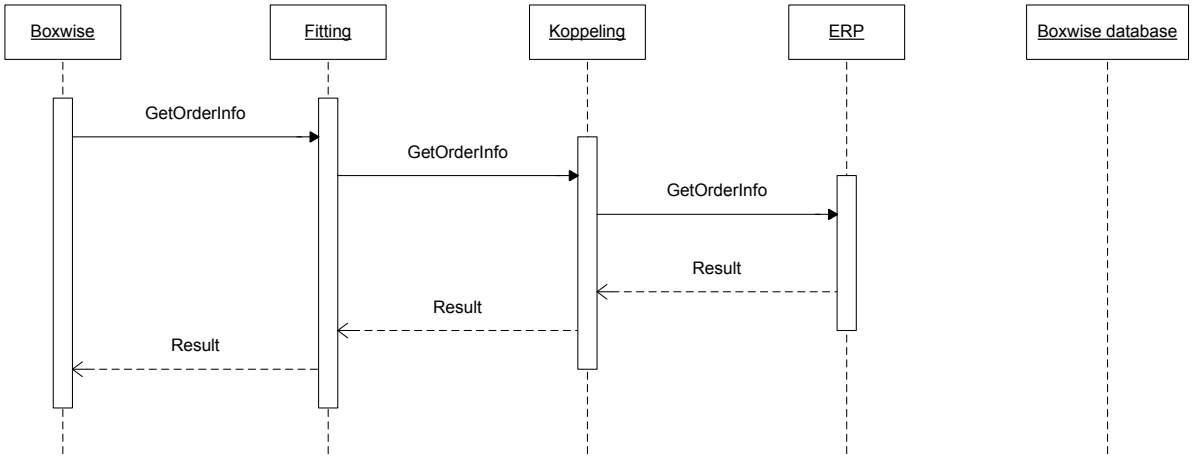
10. Informatie product



11. Barcode wijzigen



12. Informatie order



Bijlage D: Testplan

Testplan

Dit testplan is van toepassing op BOXwise en vooral de externe Exact koppeling (ExactConnection). De Exact koppeling is verantwoordelijk voor twee taken:

- Het ophalen van gegevens uit Exact Globe 2003, om het vervolgens terug naar BOXwise te sturen.
- Het versturen van gegevens naar Exact Globe 2003 waarbij de data wordt geleverd door BOXwise.

Hiervoor zullen er een aantal tests worden gedaan om te garanderen dat de Exact koppeling goed functioneert. Dat will zeggen dat:

- De juiste gegevens worden opgehaald aan de hand van de parameters die door BOXwise worden geleverd.
- De resultaten op de juiste manier worden teruggestuurd naar BOXwise.
- Eventuele fouten juist worden opgevangen, die vervolgens op een nette manier worden getoond.

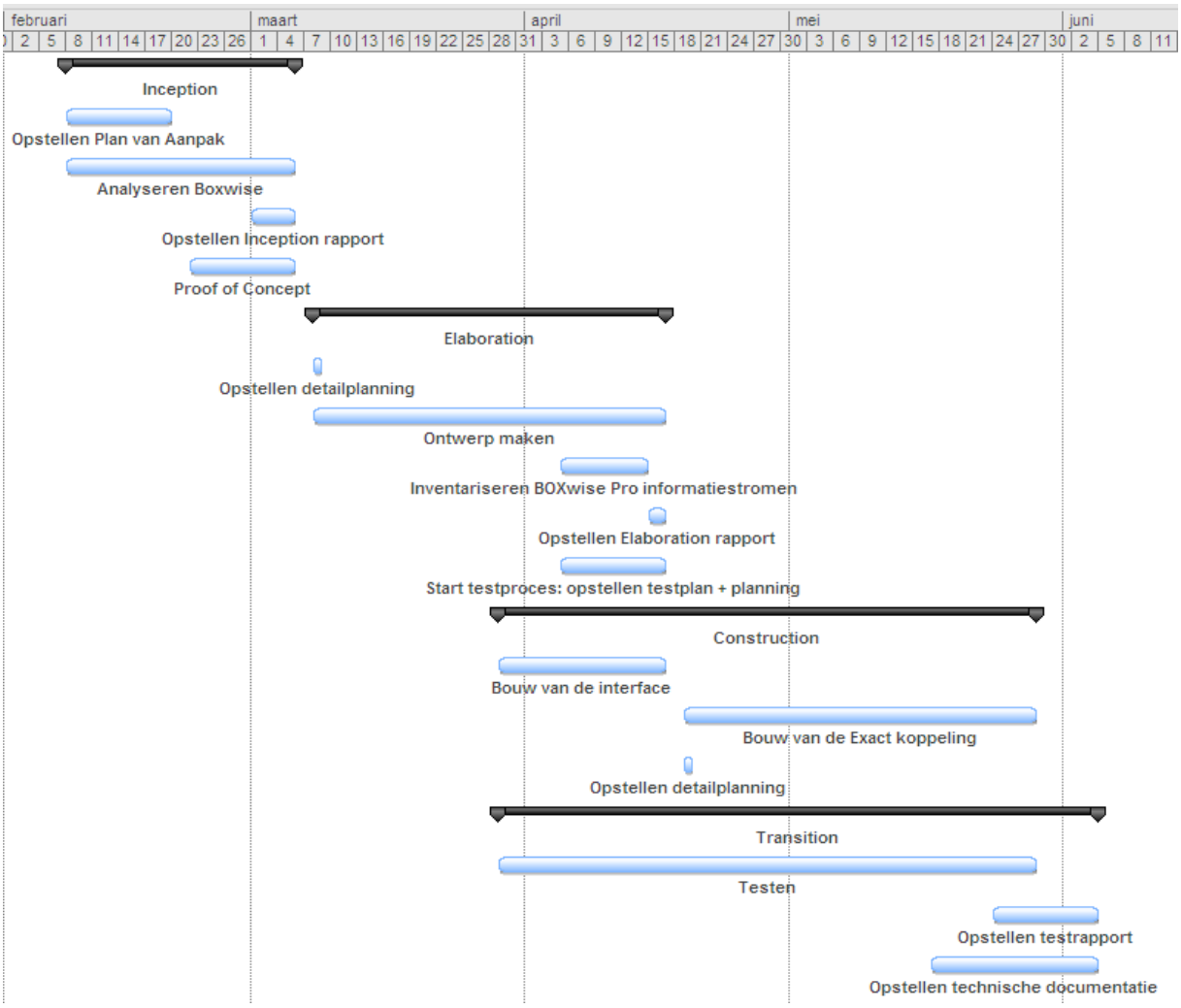
Unit testing

Voor het uitvoeren van de tests zal er gebruik worden gemaakt van unit tests. Hierdoor kan direct de software worden getest. In de namen van de unit tests zal worden beschreven wat er wordt getest. Hiervoor zal de volgende notatie worden gebruikt:

<naam informatiestroom><verwachtte uitkomst>

Planning

Voor dit testproces zal de planning voor de ontwikkeling van de koppeling worden aangehouden. Op deze manier kan de nieuwe software direct worden getest waardoor eventuele fouten direct kunnen worden opgelost.



Testsituaties

Er zijn een aantal testsituaties opgesteld. Dit is gedaan per informatiestroom. Hierdoor kan worden gegarandeerd dat alle informatiestromen juist verlopen.

--- Inkooporders ---

Ophalen leveranciers

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	3 Vendors
Correcte input: Leveranciers van magazijn 1	Warehousecode = 1 FilterText = String.Empty	3 Vendors
Correcte input: Leveranciers van magazijn 1, met filter "Leica"	WarehouseCode = 1 FilterText = Leica	1 Vendor
Foutieve input: niet bestaande wh code	WarehouseCode = %478e	0 Vendors
Foutieve input: ontbrekend warehouseCode	Null	3 Vendors
Foutieve input: filter	WarehouseCode = 1 FilterText = Appel	0 Vendors
Foutieve input (null)	Null	3 Vendors

Ophalen enkele inkooporder

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	Exception
Correcte input: Order 46 ophalen	Id = 46	1 PurchaseOrder
Foutieve input: niet bestaande order	Id = 1	0 PurchaseOrder
Foutieve input: null	NULL	ArgumentException

Ophalen alle inkooporders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	5 PurchaseOrders

Ophalen inkooporders met filter

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	5 PurchaseOrders
Correcte input: Inkooporders uit magazijn 1	WarehouseCode = 1	5 PurchaseOrders
Correcte input: Orders van leverancier 2038 ophalen	VendorNumber = 2038	3 PurchaseOrders van leverancier 2038
Correcte input: Orders ophalen met de filter "TEST"	SearchText = TEST	3 PurchaseOrders
Foutieve input: niet bestaand magazijn	WarehouseCode = &478e	0 PurchaseOrders
Foutieve input: niet bestaande leverancier	VendorNumber = qwerty	0 PurchaseOrders
Foutieve input: filter levert niets op	SearchText = qwerty	0 PurchaseOrders
Foutieve input: ontbrekend WarehouseCode	WarehouseCode = ""	5 PurchaseOrders
Foutieve input: null	Null	5 PurchaseOrders

Ophalen regels van inkooporder

Wat te testen	Waardes	Verwacht resultaat
Correct input: standaard argument	Default	0 PurchaseOrderLines
Correcte input: Regels van order 46 ophalen	PurchaseOrderId = 46	1 PurchaseOrderLines
Foutieve input: Niet bestaande order	PurchaseOrderId = 1	0 PurchaseOrderLines
Foutieve input: Null	Null	0 PurchaseOrders

Verwerken inkooporders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: order X verwerken	PurchaseOrderLines - OrderId = 1111	DataFlowObject.Result = Success
Foutieve input: niet bestaande order verwerken	PurchaseOrderLines - OrderId = -1	ArgumentException
Foutieve input: order verwerken waarbij het magazijn niet bestaat	PurchaseOrderLines - WarehouseCode = H	ArgumentException
Foutieve input: magazijn heeft geen standaard ontvangstlocatie	PurchaseOrderLines - WarehouseCode = 9	ArgumentException
Foutieve input: null	Null	Exception

--- RMA orders ---

Ophalen klanten

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	2 Customers
Correcte input: klanten van magazijn x	Warehousecode = 1	2 Customers
Correcte input: Filter	Filter = Marius	1 Customers
Foutieve input: niet bestaande wh code	WarehouseCode = H	0 Customers
Foutieve input: Filter levert niets op	Filter = &478e	0 Customers
Foutieve input (null)	Null	Alle klanten?

Ophalen enkele RMA order

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	Exception
Correcte input: RMA order 47 ophalen	Id = 47	1 RmaOrder
Foutieve input: niet bestaande RMA order	Id = 0001	Null
Foutieve input (null)	NULL	Exception

Ophalen alle RMA orders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	2 RmaOrders

Ophalen RMA orders met een filter

Wat te testen	Waardes	Verwacht resultaat
Correcte input: RMA orders uit magazijn 1	WarehouseCode = 1	2 RmaOrders
Correcte input: Orders van klant 44135 ophalen	VendorNumber = 44135	3 RmaOrders van klant 2038
Correcte input: Orders ophalen met de filter "Verkeerd"	SearchText = Verkeerd	1 RmaOrders
Foutieve input: niet bestaand magazijn	WarehouseCode = &478e	0 RmaOrders
Foutieve input: niet bestaande leverancier	VendorNumber = qwerty	0 RmaOrders
Foutieve input: filter levert niets op	SearchText = qwerty	0 RmaOrders
Foutieve input: ontbrekend WarehouseCode	WarehouseCode = ""	2 RmaOrders
Foutieve input: null	Null	2 RmaOrders

Ophalen regels van RMA order

Wat te testen	Waardes	Verwacht resultaat
Correct input: standaard argument	Default	0 PurchaseOrderLines
Correcte input: Regels van order 47 ophalen	RmaOrderId = 47	6 PurchaseOrderLines
Foutieve input: Niet bestaande order	RmaOrderId = 1	0 PurchaseOrderLines
Foutieve input: Null	Null	0 PurchaseOrders

Verwerken RMA orders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: RMA order X verwerken	RmaOrderLines - OrderId = 1111	DataFlowObject.Result = Success
Foutieve input: niet bestaande RMA order verwerken	RmaOrderLines - OrderId = -1	ArgumentException
Foutieve input: RMA order verwerken waarbij het magazijn niet bestaat	RmaOrderLines - WarehouseCode = H	ArgumentException
Foutieve input: null	RmaOrderLines = null	NullReferenceException

--- Verkoop orders ---

Ophalen klanten

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	26 Customers
Correcte input: klanten van magazijn 1	Warehousecode = 1	26 Customers (werkt niet, WarehouseCode wordt niet gebruikt, moet misschien wel?)
Correcte input: klanten met filter "Foto"	FilterText = Foto	16 Customers
Foutieve input: niet bestaande wh code	WarehouseCode = &478e	0 Customers
Foutieve input: klanten met filter "&478e"	FilterText = &478^{e}	0 Customers
Foutieve input: null	Null	Exception

Ophalen enkele verkooporder

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Order 22 ophalen	Id = 22	1 SalesOrder
Foutieve input: standaard argumenten	Default	0 SalesOrder
Foutieve input: niet bestaande order	Id = 1	0 SalesOrder
Foutieve input: null	NULL	Exception

Ophalen enkele verkooporder

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	
Correcte input: Order X ophalen	Id = 1	Één order met het Id "1"
Foutieve input: niet bestaande verkooporder	Id = 0001	Leeg resultaat
Foutieve input (null)	NULL	Leeg resultaat

Ophalen alle verkooporders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten		38 PurchaseOrders

Ophalen verkooporders met filter

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Standaard argumenten	Default	38 SalesOrders
Correcte input: Inkooporders uit magazijn 1	WarehouseCode = 1	38 SalesOrders
Correcte input: Orders van klant 44631 ophalen	CustomerNumber = 44631	2 SalesOrders van klant 44631
Correcte input: Orders ophalen met de filter "250"	SearchText = 250	9 SalesOrders
Foutieve input: niet bestaand magazijn	WarehouseCode = &478e	0 SalesOrders
Foutieve input: niet bestaande leverancier	VendorNumber = qwerty	0 SalesOrders
Foutieve input: filter levert niets op	SearchText = qwerty	0 SalesOrders
Foutieve input: ontbrekend WarehouseCode	WarehouseCode = ""	38 SalesOrders
Foutieve input: null	Null	38 SalesOrders

Ophalen verkoop orders voor snapshot

Wat te testen	Waardes	Verwacht resultaat
Correcte input: delegate		Alle verkooporders
Foutieve input: NULL	NULL	NullReferenceException

Ophalen regels van verkooporder

Wat te testen	Waardes	Verwacht resultaat
Correct input: standaard argument	Default	0 PurchaseOrderLines
Correcte input: Regels van order 9 ophalen	Id = 9	5 PurchaseOrderLines
Foutieve input: Niet bestaande order	PurchaseOrderId = 1	0 PurchaseOrderLines
Foutieve input: Null	Null	0 PurchaseOrders

Verwerken verkooporders

Wat te testen	Waardes	Verwacht resultaat
Correcte input: verkooporder X verwerken	SalesOrderLines - OrderId = 1111	
Foutieve input: niet bestaande verkooporder verwerken	SalesOrderLines - OrderId = -1	ArgumentException
Foutieve input: salesorder verwerken waarbij het magazijn niet bestaat	SalesOrderLines - WarehouseCode = null	ArgumentException
Foutieve input: null	SalesOrderLines = null	NullReferenceException

---- Tellingen ----

Één magazijn ophalen

Geldt ook voor GetWarehouseExists. Die roept dezelfde methode aan.

Wat te testen	Waardes	Verwacht resultaat
Correcte input: magazijn 1	WarehouseCode = 1	Één Warehouse
Foutieve input: niet bestaand magazijn	WarehouseCode = H	Null
Foutieve input: NULL	Null	ArgumentException

Controleren of magazijn bestaat

Wat te testen	Waardes	Verwacht resultaat
Correcte input: magazijn 1	WarehouseCode = 1	True
Foutieve input: niet bestaand magazijn	WarehouseCode = H	False
Foutieve input: NULL	Null	ArgumentException

Meerdere magazijnen ophalen (GeWarehouses<...>)

Alle magazijnen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: standaard argumenten	Default	5 Warehouses

Actieve magazijnen

Active = true. Deze wordt in de methode zelf gedefiniëerd.

Wat te testen	Waardes	Verwacht resultaat
Correcte input: alle actieve magazijnen		5 Warehouses

Inactieve magazijnen

Active = false. Deze wordt in de methode zelf gedefiniëerd.

Wat te testen	Waardes	Verwacht resultaat
Correcte input: alle inactieve magazijnen		0 Warehouses

Actieve magazijnen met een locatie code

Active = true. Deze wordt in de methode zelf gedefiniëerd.

Wat te testen	Waardes	Verwacht resultaat
Correcte input: alle actieve magazijnen met een locatie code	WarehouseLocationCode = 1-A	1 Warehouses
Foutieve input: WarehouseLocationCode ontbreekt	Null	ArgumentException
Foutieve input: Niet bestaand WarehouseLocationCode	WarehouseLocationCode = &478e	0 Warehouses

Actieve magazijnen met een standard ontvangst locatie
Active = true. Deze wordt in de methode zelf gedefiniëerd.

Wat te testen	Waardes	Verwacht resultaat
Correcte input: alle actieve magazijnen met een standaard ontvangst locatie	HasDefaultInboundLocation = true	2 Warehouses

Locaties in magazijnen ophalen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: warehouseCode	WarehouseCode = 1	Alle locaties van magazijn 1, 3 Locations
Foutieve input: niet bestaande warehouseCode	WarehouseCode = H	0 Locations
Foutieve input: NULL	NULL	ArgumentException

Controleren of een locatie in een magazijn bestaat

Wat te testen	Waardes	Verwacht resultaat
Correcte input	WarehouseCode = 1 WarehouseLocationCode = 1-A	True
Foutieve input: niet bestaande warehouseCode	WarehouseCode = H WarehouseLocationCode = 1-A	False
Foutieve input: niet bestaande warehouseLocationCode	WarehouseCode = 1 WarehouseLocationCode = &478e	False
Foutieve input: ontbrekend warehouseCode	Null WarehouseLocationCode = 1-A	ArgumentException
Foutieve input: ontbrekend warehouseLocationCode	WarehouseCode = 1 WarehouseLocationCode = null	ArgumentException
Foutieve input: null	Null	ArgumentException

Product verplaatsen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: product binnen een magazijn verplaatsen	WarehouseTransfer - WarehouseCodeFrom: 1 - WarehouseCodeTo: 1 - WarehouseLocationFrom: 1-A - WarehouseLocationTo: 1-B	DataFlowResultEnum.Succes
Correcte input: product naar een ander magazijn verplaatsen	WarehouseTransfer - WarehouseCodeFrom: 1 - WarehouseLocationFrom: 1-A - WarehouseCodeTo: 9 - WarehouseLocationTo: 9-A	DataFlowResultEnum.Succes
Foutieve input: niet bestaand product	WarehouseTransfer - ItemCode = “&478e”	ArgumentException
Foutieve input: niet bestaande locatie	WarehouseTransfer: - WarehouseLocationFrom = “&478e”	ArgumentException Hmm, geen check voor dit...
Foutieve input: negatief aantal	WarehouseTransfer: - Quantity: -1	ArgumentException
Foutieve input: NULL	NULL	ArgumentException

Tellingen verwerken

Wat te testen	Waardes	Verwacht resultaat
Correcte input: Count	Count	Tellingen zijn verwerkt
Foutieve input: niet bestaand product	Count - ItemCode = “123”;	ArgumentException
Foutieve input: niet bestaand magazijn	Count - WarehouseCode = “123”	ArgumentException
Foutieve input: NULL	NULL	NullReferenceException

Item ophalen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: GetItemArguments	GetItemArguments - ItemCode = “2100120021301”	True Item: “Groene appel”
Foutieve input: niet bestaand product	GetItemArguments - ItemCode = “123”	False
Foutieve input: NULL	NULL	ArgumentException

Itemcode van barcode ophalen

Wat te testen	Waardes	Verwacht resultaat
Correct uitvoeren	3489045905378	ItemCode van artikel x
Foutieve input: niet bestaande barcode	8376739569	Foutmelding: item niet gevonden
Foutieve input: null	NULL	NullReferenceException

Items op locatie ophalen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: standaard argumenten	Default	ArgumentException
Correcte input: items op locatie x ophalen	WarehouseCode = “1” WarehouseLocationCode = “1-A”	23 Items
Foutieve input: ontbrekende gegevens	WarehouseCode = “1” WarehouseLocationCode = null	ArgumentException
Foutieve input: niet bestaande locatie	WarehouseCode = “1” WarehouseLocationCode = “1432”	0 Items
Foutieve input: null	Null	NullReferenceException

Locaties van items ophalen

Wat te testen	Waardes	Verwacht resultaat
Standaard argumenten	Default	ArgumentException
Correct uitvoeren	ItemCode = 1630101 WarehouseCode = 1 OnlyDefaultLocations = false	1 Locations (1-A)
Foutieve input: foutieve gegevens	ItemCode = 123 WarehouseCode = 1 OnlyDefaultLocations = false	0 Locations
Foutieve input: ontbrekende gegevens	ItemCode = null WarehouseCode = 1 OnlyDefaultLocations = false	ArgumentException
Foutieve input: null	Null	NullReferenceException

Standaard locatie van een item ophalen

Wat te testen	Waardes	Verwacht resultaat
Correcte input: standaard argumenten	Default	ArgumentException
Correcte input: standaard locatie van item ophalen	ItemCode = 4840290 WarehouseCode = 1 OnlyDefaultLocations = true	1 Location (1-A)
Foutieve input: niet bestaande item	ItemCode = 123 WarehouseCode = 1 OnlyDefaultLocations = true;	Null
Foutieve input: ontbrekende gegevens	ItemCode = null WarehouseCode = 1 OnlyDefaultLocations = true	ArgumentException
Foutieve input: null	Null	ArgumentException

Voorraad item ophalen

Wat te testen	Waardes	Verwacht resultaat
Standaard argumenten (beetje nutteloos)	Default	ArgumentException
Correcte input: voorraad van item ophalen	ItemCode = 2100120021301 WarehouseCode = 1 WarehouseLocationCode = 1-A	1 ItemStockInfo (100 stuks)
Foutieve input: niet bestaande item	Itemcode = 1432 WarehouseCode = 1 WarehouseLocationCode = 1-A	0 itemStockInfo
Foutieve input: ontbrekende gegevens	ItemCode = null WarehouseCode = 1 WarehouseLocationCode = 1-A	ArgumentException
Foutieve input: null	Null	NullReferenceException

Itemidentifications beschikbaar

Wat te testen	Waardes	Verwacht resultaat
Standaard argumenten (beetje nutteloos)	Default	ArgumentException
Correct uitvoeren	ItemCode = 2100120021301 WarehouseCode = 1 WarehouseLocationCode = 1-A	1x ItemIdentifications
Foutieve input: foutieve gegevens	ItemCode = 1673103786 WarehouseCode = 1 WarehouseLocationCode = 10-G	ArgumentException
Foutieve input: ontbrekende gegevens	ItemCode = 2100120021301 WarehouseCode = WarehouseLocationCode =	ArgumentException
Foutieve input: null	Null	ArgumentException

Controleren serienummer van een item

Wat te testen	Waardes	Verwacht resultaat
Correcte input: controleren serienummer	ItemCode = 4840290 SerialNumber = 0001	True
Foutieve input: niet bestaande item	ItemCode = 4840291 SerialNumber = 0001	False
Foutieve input: niet bestaande serienummer	ItemCode = 4840290 SerialNumber = 1234	False
Foutieve input: ontbrekend itemcode	ItemCode = NULL SerialNumber = 0001	ArgumentException
Foutieve input: ontbrekend serienummer	ItemCode = 4840290 SerialNumber = null	ArgumentException
Foutieve input: NULL	ItemCode = NULL SerialNumber = NULL	ArgumentException

Barcode wijzigen

Wat te testen	Waardes	Verwacht resultaat
Standaard argumenten (beetje nutteloos)	Default	Foutmelding: geen itemcode opgegeven
Barcode correct wijzigen	ItemCode = BarCode = Creditor =	Barcode gewijzigd
Foutieve input: ontbrekende gegevens	ItemCode = BarCode = null Creditor =	Foutmelding
Foutieve input: null	Null	Foutmelding: NullReferenceException

Bijlage E: CD met alle producten

Deze CD bevat alle producten die als bijlage bij dit verslag zijn toegevoegd.

	Naam	Soort bestand
1.	**Plan van Aanpak.doc**	Word 2003
2.	**Inception rapport.doc**	Word 2003
3.	**Elaboration rapport.doc**	Word 2003
4.	**Testplan.doc**	Word 2003
5.	**Documentation BOXwise Interface.chm**	Compiled HTML Help File