En beslutstödsmodell för övergång till Visual Basic 2005
- Vilka faktorer påverkar valet?

Kandidatuppsats, 10 poäng, inom Systemvetenskapliga programmet

Framlagd: 06, 2007

Författare: Björn Gustafsson
Johan Frisell

Handledare: Lars Fernebro

Examinatorer: Odd Steen, Kjell Åke Holmberg

Omfång: 80 sidor, avrundat till närmaste tiotal, inkl. bilagor o. dyl.

En beslutstödsmodell för övergång till Visual Basic 2005
-Vilka faktorer påverkar valet?

INF 630 Kandidatuppsats

Omfång: 80 sidor, avrundat till närmaste tiotal, inkl. bilagor o. dyl.

Handledare: Lars Fernebro

Examinator: Odd Steen, Kjell Åke Holmberg

Framlagd: 06, 2007

Institutionen för Informatik, Lunds Universitet

Abstrakt

Uppsatsen behandlar det problem som organisationer har stött på när de vill uppgradera sin utvecklingsmiljö och sin produkt till Visual Basic.NET. Uppsatsen identifierar vilka faktorer det är som påverkar valet av en övergång, positiva respektive negativa. Faktorerna ger organisationer en bättre inblick i problemet och hjälper dem att avgöra huruvida de ska göra en övergång eller ej. För att arbeta fram våra faktorer har vi valt att använda oss av ett induktivt tillvägagångssätt, som mynnar ut i en kvalitativ uppsats. Faktorerna bildar en modell över vad som påverkar valet av övergång samt hur. Uppsatsen grundas i såväl litteratur inom området samt intervjuer med personer som står inför valet av övergång eller ej samt som har genomfört en övergång. Uppsatsen kommer fram till att Visual Basic 2005 påverkar samtliga av de faktorer som är avgörande för valet och att det finns faktorer som är både tekniskt och organisatoriskt anknutna. Uppsatsen kommer även fram till och presenterar specifikt de faktorer som påverkar valet.

Nyckelord: Visual Basic 2005, .NET, uppgradering av programmeringsmiljö, migrering, programmeringstrend,

Presentation av författare:

Björn Gustafsson: 830704 _____

Studerar: Systemvetenskap, Organisation och verksamhetsstödjande system, Lunds Universitet.

Kontakt:
E-post: bjorn.gustafsson@hermes.ics.lu.se
Telefon: +(46) 736 54 23 04

Johan Frisell: 850903 _____

Studerar: Systemvetenskap, Organisation och verksamhetsstödjande system, Lunds Universitet.

Kontakt:
E-post: johan.frisell@hermes.ics.lu.se
Telefon: +(46) 704 82 13 04

Tack:

Vi vill passa på att tacka vår handledare Lars Fernebro för bra feedback och handledning. Vi tackar även de organisationer som ställt upp samt dess anställda som låtit oss skriva vår uppsats med er som underlag.

Innehållsförteckning

Figurförteckning

1 Inledning

Idag går utvecklingen fort inom systemutvecklingsbranschen och marknaden kräver bättre och bättre produkter med allt fler funktioner. För att möta kravet från marknaden krävs det bra verktyg för att kunna utveckla system. Microsoft är ett av dessa företag som levererar verktyg för programvaruutveckling. Ett av deras programmeringsspråk, Visual Basic har länge varit ett populärt verktyg för att utveckla Windows applikationer. Det har funnits många versioner av Visual Basic och 2002 genomgicks det en stor förändring, Visual Basic blev .NET baserat. Många företag ville fortsätta att uppgradera sin Visual Basic miljö men den här gången stötte många på stora problem. .NET innebar stora förändringar i själva arkitekturen hos de program man skulle utveckla vilket medförde problem att konvertera sin produkt. De nya versionerna av Visual Basic som baserade sig på .NET var inte längre lika bakåtkompatibla som tidigare (Gowthaman et. al., 2005). Microsoft är medvetna om att detta är ett problem och har försökt åtgärda problemet men diverse migreringsverktyg. De har även skrivit böcker som ska hjälpa vid själva övergången. Bisbal et. al., (1999) menar även de att organisationer har stött på problem när de ska överföra sina gamla system till nyare versioner. Migreringen till en modernare utvecklingsmiljö blev nu ett mycket större och mer tidskrävande projekt vilket också innebär större kostnader vilket våra intervjuer har visat på. Migrering blev således en kostsam process för organisationer och det är inget som kunder betalar för direkt vilket också kan utläsas i citatet nedan från en av våra intervjurespondenter.

> ”Att konvertera programmet är det ju ingen som betalar för, så där ligger en stor utgift, i utvecklandet.”
>
> Respondent B

Man vill således vara säker på vad det är som kan vara problematiskt vid en övergång men även fördelarna med att byta. Ett av företagen som har stött på det här problemet med migrering är Organisation X. Organisation X vill fortsätta att förnya sin utvecklingsmiljö för att deras produkt ska vara marknadskraftig. Men eftersom migreringen från deras tidigare miljö är så pass omfattande så vill de först göra en analys angående vad som är de faktiska fördelarna och även vart de stora problemen kan uppstå. De vill helt enkelt ta reda på vad det är som påverkar valet av övergång eller ej. Vilka faktorer är det som de ska fokusera på innan de bestämmer sig för att migrera till Visual Basic.NET, positiva respektive negativa.

Bristen i kunskap kring vad det är som påverkas vid en övergång samt vad man bör se till innan man gör en övergång är stor och det finns en efterfrågan att ta reda på vilka dessa faktorer är som påverkar övergången. Vårt problem är således att ta reda på vilka dessa faktorer är som har inverkan på valet av övergång till Visual Basic 2005 (den senaste versionen av Visual Basic.NET) eller ej och vår problemformulering blir: V*ilka faktorer påverkar valet av övergång till Visual Basic och dess kringliggande miljö?* De faktorer vi finner kommer sammanfattas i en beslutstödsmodell som kommer ligga i grund till valet av övergång eller ej som även visar på ett samband faktorerna emellan samt hur de påverkar valet.

Frågan grundar sig i det problem vi har identifierat hos Organisation X. Vi anser att vilka faktorer som ska tas i beaktning vid övergången har varit diffusa för Organisation X. Dock är uppsatsen inte specifikt utformad för Organisation X utan fungerar generellt på andra

organisationer med samma problem. D.v.s. ursprunglig miljö har ingen inverkan på modellen. Att identifiera de faktorer som påverkar valet kan dels identifiera svårigheterna och riskerna men även vilka faktiska fördelar en uppgradering kan innebära. Vi har inte enbart funnit detta problem hos organisation X, detta problem återfinnes även bland andra organisationer bland annat har vi funnit den hos Mandator AB som stod inför samma problem som organisation X, vi tror därför att detta är ett generellt problem som organisationer kan tänkas stå inför.

1.1 Problemformulering

V*ilka faktorer påverkar valet av övergång till Visual Basic och dess kringliggande miljö?*

1.2 Avgränsningar

Vi har en tanke om att importansen av de olika faktorerna kan skilja sig ifrån fall till fall och att eftersom vi ska göra en generell och adaptiv modell så kommer vi inte att försöka lägga något viktighetsaspekt på någon av faktorerna, utan endast presentera varför de är faktorer att ta i beaktning samt hur de hänger samman i modellen. Det är sedan upp till intressenten att anpassa modellen efter sin egen verksamhet och intressen.

Vi avgränsar oss även i den mån att vi endast tar upp de faktorer som vi anser vara av mest importans men gräver man djupare i Visual Basic 2005 och dess kringliggande miljö kan man säkerligen finna mindre funktioner och tjänster som kan ha inverkan på valet.

1.3 Syfte

Vårt huvudsakliga syfte med uppsatsen är att utveckla en beslutsstödsmodell som kan vägleda företag och organisationer huruvida de bör uppgradera sin rådande mjukvaruplattform till Visual Basic 2005 och dess kringliggande miljö. Ett underliggande syfte med uppsatsen är att det inte ska spela någon roll från vilken utvecklingsmiljö man kommer ifrån utan uppsatsen skall vara tillämpbara oavsett bakgrund. Intressenten ska efter läsandet förstå hur dessa faktorer som påverkar valet hänger samman samt hur de associerar med Visual Basic 2005 samt vilka det är. Uppsatsen ger även en viss inblick i Visual Basics egenskaper.

1.4 Disposition

Vi har försökt att utforma vår disposition så att arbetet håller en röd tråd samtidigt som det har en bra struktur och uppfyller kraven för en akademisk skrift. Arbetet öppnas med en inledning och presentation av problemet som ni precis har tagit del av. Vidare presenteras vårt tillvägagångssätt, vårt metodval samt information angående vår informationsinsamling och analys. Efter metodgenomgången följer en genomgång och presentation av vår fakta. Presentationen består av tre kapitel. Det första kapitlet är en produktpresentation, det andra kapitlet är vår litteraturgenomgång och det tredje kapitlet är vår presentation av våra intervjuer – empiri. Efter genomgången och presentationen följer vår diskussion, där vi formar vår modell.

Uppsatsen avslutas med en konklusion innehållandes vår slutsats, metodutvärdering, vidare forskningsförslag samt lite kritik på uppsatsen.

2 Metod

Vår problemformulering i denna uppsats är att finna vilka faktorer det är som påverkar övergången till Visual Basic 2005 och dess kringliggande miljö. För att kunna uppfylla det syftet så ska vi leta efter dessa faktorer på tre nivåer. Dels ska vi söka efter egenskaper hos Visual Basic 2005 som kan ha inverkan på valet av övergång eller ej, dels ska vi leta efter faktorer i själva produkten som sådan som kan ha inverkan och dels ska vi leta faktorer på ett mer organisatoriskt plan, faktorer som såväl påverkar organisationen och som påverkas av organisationen. Samtliga av faktorerna kommer att eftersökas såväl i litteratur som i intervjuer.

Anledningen till att vi söker efter faktorer på dessa tre nivåer är att vi har en tanke om att det är på det sättet en övergång kan påverka. Tanken grundar sig i våra tidigare erfarenheter och kunskaper framförallt från vår systemvetenskapliga utbildning.

När vi granskat vad som kan ha inverkan på valet på de olika nivåerna, både i litteratur och i intervjuerna så ska vi försöka finna de verkliga faktorerna genom att diskutera vad vi funnit och hur det påverkar valet. D.v.s. vad är det företag fokuserar på när man gör en övergång till Visual Basic 2005. Vi ska också försöka finna samband mellan de olika faktorerna för att kunna kategorisera dem och förstå vad som mer övergripande påverkar valet. Det gör vi för att komma ett steg närmre en modell över faktorerna och för att förstå hur det vi har funnit under analysen har inverkan på valet. Vidare ska vi försöka sammanfoga samtliga faktorer och kategorier till en modell och skapa en klar modell över vad som påverkar valet och hur.

Vårt tillvägagångssätt lyser igenom vår disposition där både litteraturgenomgång och empirikapitlet är indelat efter de tre nivåerna på vilka vi har sökt efter faktorer.

2.1 Kvalitativ metod

Man särskiljer på kvantitativ och kvalitativ forskning. Vi utför en kvalitativ undersökning då vi anser att det ger oss en djup och rik bild av problemområdet. Vi vill ta del av andra personers tankar kring vad som kan ha inverkan på valet. Bryman (2002) citerar Lofland & Lofland (1995) ”... direkt samspel är den mest omfattande betingelsen för att kunna ta del av det som sker i en annan människas medvetande”. Vi får mer av det här samspelet när vi använder oss av en kvalitativ studie och anser det därför vara den metod som lämpar sig bäst. Vi vill kunna hantera problemet ur möjliga framtida användare av vår modells ögon. Ni kommer självklart även finna tillvägagångssätt som tyder på att det är en kvalitativ studie såsom kvalitativa -, semi - strukturerade intervjuer. Vi använder oss av ord och inte kvantiteter vilket är en stor skillnad tillvägagångssätten emellan (Bryman, 2002) samt att kvalitativ metod försöker skapa en mer nyanserad och detaljerad bild av problemet (Hartman, 1998). Dock inte sagt att dessa är de enda skillnaderna.

2.2 Undersökningsmetod

När det kommer till att välja en undersökningsmetod för vår studie kommer vi inte enbart arbeta utefter en metod. Vi kommer att använda oss dels av en induktiv strategi (se fig. 2.1) då vi kommer att göra observationer på Visual Basic 2005 för att få fram resultat om vilka faktorer det är som påverkar valet av en övergång. På samma sätt kommer vi att använda intervjuerna, vi ska observera vad respondenterna anser ha inverkan på deras val för att få resultat angående vilka faktorer det är som påverkar valet. Dessa resultat sammanfogas sedan och diskuteras för att skapa en teori kring vilka faktorer som påverkar valet vilket ska leda fram till vår tänkta modell.

Observation/Resultat ⇨ Teori

Fig. 2.1 Induktivt tillvägagångssätt (Bryman, 2002, s.)

Enligt Bryman (2002) förknippas den induktiva strategin ofta med ett kvalitativt tillvägagångssätt, vilket vi också använder. Dock kommer vi även att arbeta efter ett deduktivt tillvägagångssätt när vi till våra intervjuer bygger frågorna på kunskap som vi tagit del av i litteratur. Deduktion enligt Bryman (2002) innebär att man tar sin teori utför observationer vilket leder fram till ett resultat. Man kan likna det vid fig 2.1 med modifikation då teorin är längst till vänster i figuren. Genom att arbeta på med de båda tillvägagångssätten leder det oss in på något som kallas Abduktion. Abduktion är enligt Alvesson & Sköldberg (1994) en undersökningsmetod som är en kombination av en induktiv ansats samt en deduktiv ansats. Man använder sig här båda delar, där en analys av empirin kan kombineras med, studier av tidigare teori i litteratur.

2.3 Informationsinsamling

Vår studie grundar sig främst i expertintervjuer och litteraturinsamling där våra expertintervjuer står för primär data. Litteraturinsamlingen kommer att användas för att skapa kunskap om ämnen och bekräfta våra för föreställningar, men även för att möjliggöra en tolkning av den information som återfinns i expertintervjuerna. Primär data är data som inte är hämtat från någon annan källa utan som är skapade av personerna bakom uppsatsen. Primära data har för avsikt att tillföra ny data till uppsatsen. (Miles & Huberman, 1994; Yin, 2003). Primära data har vi använt både för att finna nya faktorer samt jämföra med sådan vi redan funnit i litteratur. Således menar vi på att litteraturen både styrker och hjälper till vid analysen av empiri. Empirin i sin tur har samma förhållande till litteraturen, den hjälper till vid analys och kan styrka den fakta vi funnit i litteraturen.

När vi först samlar in information kommer vi att tolka och sätta oss in i den information vi har funnit. Sedan ska vi arbeta med informationen och försöker utvinna vad som är relevant och intressant för vår frågeställning. Utefter den information vi får fram fortsätter vi sedan informationsinsamlingen med våra nya kunskaper i bagaget för att ytterliggare komma närmre ett svar på vår frågeställning. Den här processen ska upprepas om och om igen tills vi känner att

vi har den information vi behöver för att utföra vår diskussion.

2.3.1 Litteraturinsamling

Vi ska som tidigare nämnt använda oss av så väl akademisk litteratur samt mer tekniskt inriktad litteratur eller facklitteratur. Facklitteraturen ligger främst i grund för vår kunskap kring produkten Visual Basic 2005 med kringliggande miljö. Det är här vi letar efter egenskaperna hos Visual Basic 2005 som sedan med hjälp av annan litteratur har analyserats för att se om de har någon inverkan på valet.

För att granska och analysera de olika egenskapernas och faktorernas inverkan på valet ska vi använda oss av akademisk litteratur då vi tror att denna inte är lika partisk i bedömningen. Den akademiska litteraturen kommer främst att eftersökas i olika databaser såsom ELIN och ACM.

Vi ska ha en så kritisk syn som möjligt på all litteratur vi läst för att skapa en så trovärdig och heltäckande modell som möjligt. I de fall vi kommer att använda oss av litteratur från Microsoft och MSDN kommer vi försöka ha en kritisk syn på materialet då vi har en tro om att de kan vara partiska i sin bedömning av produkten.

2.3.2 Val av studieobjekt och respondenter

Respondenterna vi väljer att intervjua är experter inom området vi söker information, vilket gör att vi utför en form av expertintervjuer (Lundahl & Skärvad, 1999). Vi anser att våra intervjuobjekt har en bred branschkännedom och kan därför ge oss information som vi tidigare inte träffat på i vår litteratur.

Vi kommer att använda oss av Organisation X för att försöka finna vad företag har för tankar kring faktorer vid byte till Visual Basic 2005 och kringliggande miljö. Organisation X ska bidra med möjliga nya faktorer och även i viss mån bekräfta vad vi har funnit i litteratur. Inom Organisation X ska vi försöka finna möjliga faktorer på två nivåer inom organisationen. Den ena parten är systemutvecklare (respondent B) och den andra parten har en mer organisatorisk post (respondent A). Man kan tänka sig en möjlig skillnad i tankarna kring faktorer de två parterna emellan. De här intervjuerna är av slaget expertintervjuer då de inte har gjort någon egentligen övergång men ändå har stor kunskap angående vad det är som påverkar deras val.

Förutom Organisation X ska vi även använda oss av Martin Sällberg (i följande kapitel nämnd som respondent C) som tidigare arbetat som anställd på Mandator, senare Semcon. Martin Sällberg har arbetat med övergång till .NET miljö tidigare och har således stor kunskap om de olika faktorerna som påverkade han och hans organisations val. Vi har således respondenter både från folk som sitter i den sits där man vill uppdatera, och från dem som har gjort det. Intervjun med Martin Sällberg bidrar med primär data till uppsatsen då han har utfört en migrering och vet vad som påverkade valet och också vad konsekvenserna blev.

En mer fullständig beskrivning av respondenterna och deras organisationer återfinns innan intervjuerna i slutet av arbetet under rubriken ”Presentation av respondenter samt deras organisationer”.

Som säkerligen noterat ovan så kommer vi att använda oss av representativa studier, d.v.s. vi har själva valt vilka företag som passar in för att representera verkligheten. Vi är väl medvetna om att det kan finnas brister i ett sådant arbetssätt såsom bias, men anser att för att kunna göra en rikare och mer djupgående analys (kvalitativ studie) så är det en nödvändig avgränsning.

2.3.4 Intervjuform och analysmetod

Våra intervjuer är av det kvalitativa slaget. Anledning till det är att vi vill fånga intervjupersonens egna uppfattningar och synsätt. Tanken är att vi som intervjuare inte ska bestämma riktningen av intervjun utan att den ska föras av intervjupersonen, intervjuerna ska vara flexibla. Enligt Bryman (2002) uppfylls dessa krav av en kvalitativ intervju. Dock ska vi inte använda oss av en helt ostrukturerad intervjumetod, utan snarare en semistrukturerad. Vi kommer att utforma en intervjuguide med olika ståndpunkter som ska beröras under intervjuernas gång men även en del frågor är utformade i förväg. Vi hoppas att på det här sättet kunna få ut så mycket information ur intervjun som möjligt och samtidigt motverka bias.

Intervjuerna kommer att ske vid fysiska möten och ska bli inspelade. När intervjuerna är avslutade kommer vi transkribera dem ordagrant i den mån det är möjligt (där ljudet är upptagligt). Det transkriberade materialet stavningskontrolleras och vi korrigerar små fel såsom alltför stor grad av talspråk. Sedan skickas intervjun till den intervjuade så de får granska den och godkänna den. De får om de vill utesluta något eller förklara något. Det är även möjligt att vi kommer att göra en uppföljning på svar som vi inte riktigt förstår eller vill ha mer information kring.

Intervjuerna kommer vara uppdelade i två omgångar där den fösta omgången är med organisation X och den andra omgången med Martin Sällberg. Detta görs för att vi ska kunna bygga vår andra intervju på information vi kunnat utröna ur våra första intervjuer och på detta sätt arbeta mer iterativt enligt Bryman (2002). Vi kommer även efter första intervjuomgången söka vidare i litteraturen om det vi funnit och på detta vis gå in i nästa intervjuomgång med mer kött på benen. När det här är avklarat ska vi koda dem efter ett kodningsschema som bygger på egenskaper samt faktorer som vi funnit i vår litteratursökning inom ämnet. Men eftersom ett syfte med intervjuerna är att finna nya faktorer som vi har missat ska vi även försöka att koda efter möjliga nya faktorer. När den första kodningen är gjord jämför vi vårt kodningsresultat och diskuterar möjliga nya faktorer och skillnader kodningarna emellan. När det är färdigt och vi är överens sammanställer vi kodningen och markerar ut de nya faktorerna som vi har stött på. Kodningen ska vi göra för att vi på ett effektivt sätt ska kunna utvinna information ur intervjuerna. Kodningsschemat kommer att bifogas med varje intervju.

2.3.5 Intervjuguide

För att få svar på vår frågeställning och tillgodose vårt syfte med uppsatsen har vi valt att ställa frågor som dels rör organisationen och dess olika nivåer, dels faktorer som Visual Basic bidrar med samt vilka egenskaper Visual Basic har som har inverkan på en övergång.

I intervju 1 som motsvarar bilaga 1 tas följande ämnen upp:

- Visual Basic 2005 och dess egenskaper: Här eftersöker vi information som är mer specifik för Visual Basic 2005 på ett tekniskt plan.
- Teknisk nivå: Här strävar vi efter hur och vad tekniken har för inverkan på en övergång eller ej. Spelar möjligtvis Visual Basics egenskaper in på ett beslut eller ej? Ett exempel på en möjlig diskussion denna punkt kan leda till är huruvida effektiviteten av Visual Basic 2005 spelar in i övervägandet eller ej.
- Organisatorisk nivå: Här letar vi efter möjliga faktorer på som kan spela in på en övergång. Man skulle kunna tänka sig att där fanns en ekonomisk aspekt av en övergång, vilket vi genom detta ämne vill försöka fånga.

Inför intervjuomgång 2 har vi ändrat våra ämnen en aning, detta gör vi för att resultaten ska få ett bättre djup, samt för att få en starkare anknytning till vår frågeställning. I bilaga 2 ser man de specifika frågorna. Nedan kommer vi att beskriva varför vi tar upp dem och ge dem återkoppling till frågeställningen och syftet.

- Som det framgår ur bilaga 2 presenterar vi något som kallas VBVSF vilket är en förkortning för Visual Basic, Visual Studio Faktorer. Detta ämne väljer vi att behandla då det är just detta vi söker information om för att svara på vår frågeställning.
- OF som står för organisatoriska faktorer har även den anknytning till frågeställningen, här försöker vi utröna om de finns faktorer på den organisatoriska nivån
- Vi har sedan sammanställt faktorer vi kunnat utröna i tidigare intervjuer. Vi väljer här att prata om dessa för att se om det är något som även denna respondent tycker är viktigt.
- Vi försöker till sist fånga information om hur de olika faktorerna hänger samman.

2.4 Etik

Etik och etiska frågor är något som kan uppkomma när forskning bedrivs. Det rör sig framförallt om hur individen som skall studeras behandlas, hur den information som erhålles genom olika former av undersökningar ska behandlas för att inte skada individen eller organisation man arbetar med kort sammanfattat som konfidentialitet. (Bryman, 2002)

Vi har ett antal gånger under arbetets gång ställts inför olika etiska problem, de har framförallt handlat om hantering av känslig information. För att arbeta med detta har vi hela tiden hållit våra respondenter klara med våra avsikter och berättat att deras intervjuer kommer att bli inspelade för att sedan transkriberas. Vid färdigställandet av transkriberingen skickade vi ut denna till

respondenten för deras godkännande. Respondenten hade här möjlighet att komma med synpunkter eller förslag om att radera känslig information.

Somliga av de organisationer samt respondenter vi haft kontakt med har bett oss ej använda deras riktiga namn. Namn som kommer att användas för igenkännelse är: respondent A, B, C samt organisation X. Vi kommer att beskriva deras roller inom organisationen samt vilken typ av organisation det rör sig om så att läsaren ändå ska få en bra inblick i hur organisationen fungerar och varför vi valt dem som respondenter. Vi väljer att göra detta av de enkla skäl att personerna och organisationerna vill hålla sig anonyma så att inga företagshemligheter sprids via Internet då denna uppsats kommer att bli en allmän handling. Informationen om hur man ska arbeta med etik är hämtat ut Bryman (2002).

2.5 Validitet och reliabilitet

Det har visat sig finnas en del svårigheter med att använda dessa två ovanstående begrepp inom kvalitativ forskning (Bryman, 2002). Vi har valt använda LeCompte & Goetz (1982) termer för validitet och reliabilitet som finns presenterade i Bryman (2002) s. 257. De använder sig av fyra begrepp, extern reliabilitet, intern reliabilitet, intern validitet och extern validitet.

Extern reliabilitet står för i vilken utsträckning studien går att upprepa. (Bryman, 2002). Att utföra denna studie med exakt samma respondenter och i samma miljö kan nog te sig ganska svårt då våra respondenter troligtvis har en annan bild av övergången efter genomförandet av studien än vad de har nu. Förhoppningsvis eftersom de har intresserat sig för ämnet och själva aktivt sökt efter fakta kring ämnet men även för att vi troligtvis kommer påverka deras syn och väcka en del tankar hos dem. Vilket också är ett problem som Bryman (2002) belyser då han säger att det är ”omöjligt att ”frysa” en social miljö”. Däremot har vi en tanke om att det finns många fler organisationer som sitter i samma sits som man skulle kunna göra samma studie på för att se om man får samma resultat. Av den anledningen anser vi vår externa reliabilitet vara relativt hög.

Den interna reliabiliteten står för i vilken mån forskarna är ense om hur de ska tolka deras resultat (Bryman, 2002). För att öka den interna reliabiliteten i uppsatsen ska vi koda intervjuerna vi genomför var för sig, för att sedan jämföra våra resultat och mäta hur pass väl de stämmer överens. I övrigt när det gäller resultat och diskussion kommer vi inte presentera någon information som vi inte båda är överens om. När vi efter kodning på eget håll jämförde våra resultat kom vi fram till en intern reliabilitet på 72 % vilket vi anser vara en bra siffra med tanke på att vi även kodade mot att identifiera nya faktorer. För att få fram siffran tog vi antalet markeringar som vi hade samma delat med det totala antalet markeringarna vi gjort.

Den interna validiteten i vilken grad våra observationer stämmer överens med vår teori (Bryman, 2002). För att öka den interna validiteten i vår uppsats så kommer vi att använda oss av öppna intervjuer där vi försöker skapa en så bra och sann bild av vad som påverkar övergången som möjligt. På samma sätt ska vi även försöka motverka bias i högsta mån för att få en teori som väl stämmer överens med de observationer vi har utfört.

Den externa validiteten är i vilken grad man kan använda våra resultat i en annan situation (Bryman, 2002). Vi har tidigare nämnt ska tidigare miljö inte ha någon inverkan på modellen utan den ska vara anpassad för organisationer som överväger en övergång till Visual Basic 2005 generellt. Uppsatsen är inte heller anpassad för organisation X specifikt utan går att använda även på andra organisationer.

Vi har även arbetat med respondentvalidering enligt Bryman (2002). Som vi nämnde ovan ville vi försäkra oss om att vi hade förstått vad våra respondenter hade velat förmedla i deras intervjuer. Vi vill även ge dem en chans att censurera delar de inte vill att vi ska presentera, med avseende på företagshemligheter eller annan känslig information. För att arbeta med detta skickar vi ut den transkriberade intervjun till respektive respondent så de själva får korrigera och lämna förslag på vad som får presenteras i själva uppsatsen.

2.6 Målgrupp

Vår uppsats riktar sig till informatikstudenter eller organisationer med en allmän insikt i området samt grundläggande kunskap om systemutveckling. Vi har en ordlista som kan underlätta möjliga svårigheter med ord och förkortningar.

3 Produktpresentation

Under det här kapitlet finner ni en produktpresentation. Produktpresentationen gjordes för att vi skulle kunna identifiera olika egenskaper hos Visual Basic 2005 och dess kringliggande miljö som hade inverkan på valet. Vi har även valt att ta med en kort beskrivning av produkten i uppsatsen för att ge läsaren en liten inblick i vad Visual Basic 2005 egentligen är samt öka förståelsen för kommande kapitel. Informationen är främst hämtad från Microsofts beskrivning av produkten och även vad vi själva har för erfarenhet efter användning.

3.1 Visual Basic

Visual Basic 2005 är den senaste versionen av Microsofts programmeringsspråk Visual Basic. Visual Basic är från början en dialekt på BASIC (*Beginner's All-purpose Symbolic Instruction Code*) och är utvecklat av Microsoft.

> "Visual Basic is a tool for productively building type-safe and object-oriented applications. It allows developers to create a wide range of Windows, Web, mobile, and Office applications built on the .NET Framework."
>
> MSDN(2007 c)

Visual Basic 2005 är således ett komplett objektorienterat programmeringsspråk som stödjer hög återanvändbarhet via t.ex. arv till skillnad från dess föregångare som inte var baserade på .NET (Hayes, 2002). En av Visual Basics grundtankar är att möjliggöra lätt programmering, språket ska vara lätt att lära sig och lätt att använda enligt Microsoft. Visual Basic 2005 används för att utveckla såväl Windowsapplikationer som webb, mobil och kontorsapplikationer. Visual Basic är också vad man kallar för ett Rapid Application Development (RAD) verktyg och kan användas för att snabbt utveckla produkter (Harkness et. al. (2007), Zubeck (1997), MSDN (2007 d). Visual Basic 2005s syntax liknar till stor del de tidigare versionerna av Visual Basic men sedan införandet av Visual Basic.NET så är även Visual Basic en del av Visual Studio.

3.2 Visual Studio

Visual Studio 2005 som är den nuvarande versionen av Visual Studio, det är inte något specifikt för just Visual Basic utan innefattar såväl C#, J#, Visual Basic, Visual C++ och webbspråk såsom ASP.NET. Visual Studio är en utvecklingsmiljö som tillhandahåller verktyg som hjälper till vid utvecklingen av applikationer med de olika programmeringsspråken. Kompilering, texteditor och debugg funktioner är några av de funktioner som återfinns i Visual Studio. Visual Studio är vad man kallar för en IDE (integrated development environment). Visual Studios största kännetecken är dess visuella programmerings möjligheter. Den tillhandahåller färdiga

komponenter som kan använda för att utveckla gränssnitt såsom knappar, behandla trådar (threads) och liknande. Tanken är att man snabbt ska kunna skapa exempelvis en knapp utan att behöva skriva flera rader kod. Man kan således säga att Visual Basic är en del av Visual Studio och kan dra nytta av dess funktioner och verktyg. I fig. 3.1 visas hur Visual Basic befinner sig i Visual Studio och använder sig utav dess funktioner och egenskaper. Väljer man Visual Basic 2005 som programmeringsspråk får man således även tillgång till de funktioner som finns tillgängliga i Visual Studio.

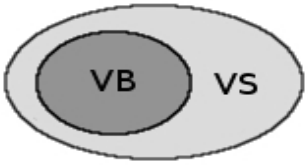

Fig. 3.1 Visual Basics förhållande till Visual Studio (Egen)

3.3 .NET

Microsofts .NET är en plattform som tillhandahåller en rad tjänster. .NET i sig är inget programmeringsspråk men är grunden för många programmeringsspråk. .NETs grund är .NET Framework, det är här körningen sker samt klassbiblioteken återfinns. .NET initiative är en annan del av .NET som innehåller protokoll och liknande för kommunikation över Internet. Vid första anblicken kan .NET ha stora likheter med Java och UCSD Pascal med sina färdiga klassbibliotek men även deras fokus på mobila enheter och Web Services har stora likheter (Zerzelidis & Wellings, 2005). Faktum är att Microsoft faktiskt har hämtat många av deras idéer därifrån. (Evjen et. al., 2005) Fig. 3.2 visar på .NETs innehåll och struktur.

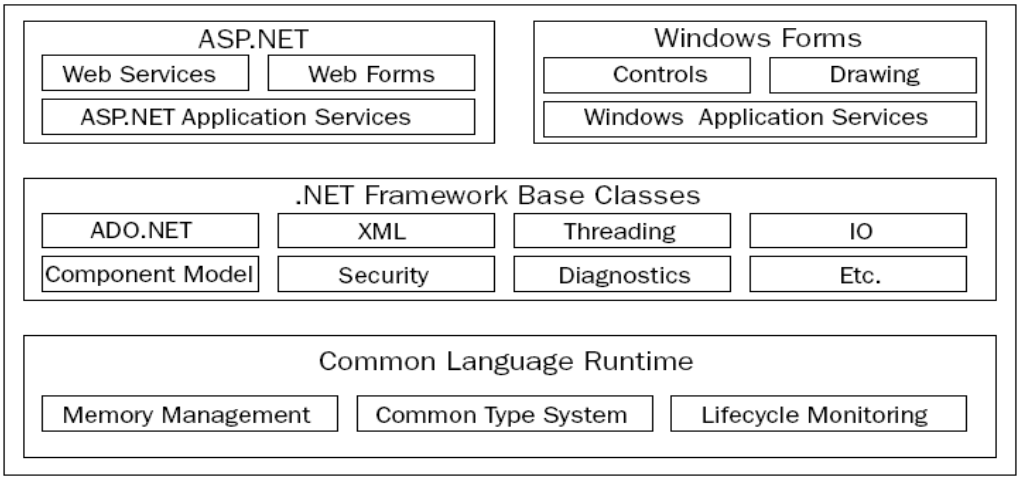

Fig. 3.2 .NET (Evjen et. al., 2005 s. 4)

I botten återfinns “Common Language Runtime” (CLR). CLR är där själva “motorn” hos .NET ligger. Minneshantering, hantering av datatyper m.m. Här översätts alla datatyper till ett standard språk som exempelvis möjliggör arv mellan olika språk. Det är således möjligt att i Visual Basic

2005 ärva från en klass skapad i C#. Mittensegmentet innehåller de olika klasserna som är tillgängliga för de språken som använder sig av .NET. Klasserna behandlar saker såsom databas kopplingar, nätverk, datamanipulering m.m. Dessutom är klasserna standarder vilket kan medföra hög kompatibilitet med annan programvara (Evjen et. al., 2005). Det översta lagret är gränssnittet. Ena sidan är webbgränssnitt (ASP.NET) och andra sidan är Windowsgränssnitt (Windows Forms). Det här lagret förmedlar således den information som processas i de undre lagren.

Väljer man Visual Basic 2005 som programmeringsspråk så väljer man också .NET som arkitektur eller plattform som många väljer att kalla det. .NET medför en hel del möjligheter och begränsningar. Det är också här många av de egenskaper som Visual Basic och Visual Studio har grundar sig. CLR möjliggör exempelvis integrering av flera programmeringsspråk i samma applikation så länge de bygger på .NET. Det tillhandahåller även ADO.NET som är en XML baserad koppling med databaser. .NET möjliggör även trådning och levererar hög säkerhet för källkoden (MSDN, 2007 a). Visual Basic 2005 erhåller således mycket funktionalitet från .NET och man kan därför fortsätta på tidigare presenterad bild över förhållandet mellan Visual Basic och Visual Studio där Visual Studio tillhandahöll många funktioner och möjligheter till Visual Basic och lägga till även .NET (fig. 3.3).

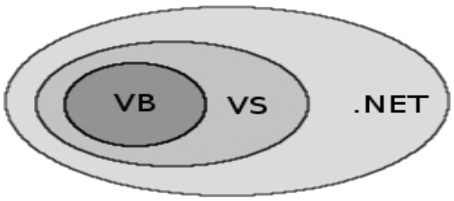

Fig. 3.3 Visual Basic och Visual Studios förhållande till .NET (Egen)

I fig. 3.3 är det .NET som tillhandahåller funktioner och möjligheter till Visual Basic och Visual Studio. Utan .NET klasser för multitrådning hade inte Visual Studio kunnat hantera trådar och inte heller Visual Basic hade kunnat användas för att skriva program som använder sig av trådar. Om du väljer Visual Basic 2005 så måste du även använda dig av .NET men .NET kan knappast sägas vara någon egenskap hos Visual Basic men är ändå en viktig del av valet. .NET i sig har nämligen en del konsekvenser. Vi kommer att återkomma till vad för inverkan .NET har på valet av övergång eller ej senare.

4. Litteraturgenomgång

Under det här kapitlet kommer vi att presentera vad vi har funnit i litteratur som kan ha inverkan på valet av övergång eller ej. Under metodkapitlet beskrev vi hur vi letade faktorer på tre plan. Dels letade vi i egenskaperna, dels i själva produkten och dels på ett mer organisatoriskt plan. På samma sätt har vi delat upp litteraturgenomgången.

4.1 Egenskaper som inverkar på valet

När vi gjorde en granskning i litteratur av de egenskaper som Visual Basic 2005 besitter fann vi att en del av dem hade positiv eller negativ inverkan på utvecklingen. Under den här delen av kapitlet ska vi presentera de egenskaper vi funnit ha inverkan och som således också kan påverka valet av övergång. Notera att det alltså inte är en komplett genomgång av Visual Basic 2005s egenskaper utan endast de som vi har funnit ha inverkan på valet.

4.1.1 Visuellt Programmeringsspråk

I vår produktpresentation i kapitel 3 visade vi på att Visual Basic 2005 använder sig av Visual Studio och dess inbyggda komponenter för att utveckla applikationer. Visual Studio använder sig av något som kallas visuell programmering. Vi har i litteraturen som refereras under det här kapitlet funnit att det kan ha vissa konsekvenser och således inverkan på valet av övergång eller ej.

4.1.1.1 Visuella programmeringsspråk och dess inverkan

Motsvarigheten till Visual Programmering Languages eller visuella programmeringsspråk förkortat VPL är textbaserade språk. De textbaserade språken är starkt influerade av hårdvarumarknaden men likväl i talspråket och det matematiska språket finner man stora likheter. De textbaserade språken har dock ett par svagheter. Textbaserade språk är ofta komplexa och svåra, fokus måste därför läggas på syntaxen hos programmeringsspråket vilket hindrar utvecklaren från att vara kreativ. (Ahmed, 1999)

De visuella programmeringsspråken har sin grund i de visuella gränssnitten som i dag återfinns hos majoriteten av vår mjukvara. Med visuella programmeringsspråk menas att man har ersatt användandet av ord och siffror med bilder eller visuella animationer. Man slipper således skriva mycket av den kod som ligger bakom ett visuellt moment, t.ex. en knapp (Shu, 1988). Ahmed (1999) har även visat på fyra kännetecken hos VPL enligt nedan.

- Det behandlar endast sådana mekanismer som är direkt knutna till själva applikationen. Man visualiserar således inte loopar, listor, olika objekts användningsområde (scoop) m.m. utan snarare knappar, fönster och objekten i sin helhet.
- Visuella programmeringsspråk är konkreta. Med det menar Ahmed (1999) att det ofta använder sig av samma teknik för flera funktioner. Såsom ”drag and drop” för att skapa såväl checkboxar som knappar. Motsvarigheten i ett textbaserat språk hade haft helt skilda kommandon.
- Visuella programmeringsspråk visar på väldigt tydliga relationer mellan objekt.
- Direkt feedback. Du kan med en gång se vad som händer vid implementeringen av ett visst objekt eller t.o.m. en variabel. Man behöver inte anropa någon funktion för att se hur resultatet blir.

Det finns ett flertal fördelar med att använda ett visuellt programmeringsspråk även om text är att föredra i vissa lägen där bild inte fungerar exempelvis vid deklarering av variabler. Det finns framförallt två stora aspekter med visuell programmering som påverkar valet. Den första är den faktorn att det gör programmeringsspråket mindre komplext och lättare att förstå vilket var ett problem med de textbaserade. Visuell programmering syftar till att förenkla programmeringen (Ahmed, 1999). Det faktum att man ser relationerna och inte behöver läsa sig till exakt hur t.ex. olika objekt relaterar till varandra ligger i grund för denna motivering. Det råder dock viss tvivel huruvida visuella eller textbaserade programmeringsspråk är lättast att förstå. Green et al. (1992) menar att det är lättare att förstå de textbaserade språken medan Ahmed (1999) menar motsatsen. Svaret vilken av typerna som är lättast kan således vara kontext och utvecklarberoende.

Att designa mjukvara är en svår uppgift och det behövs verktyg för att simplifiera denna process. Man bör sära på utvecklandet av själva funktionerna och designen. En kombination av textbaserad programmering för utvecklandet av funktionerna och visuell programmering för utvecklandet av designen är en bra lösning (Song, 1994). Textbaserad programmering och visuell programmering har olika fördelar och kompletterar varandra på olika plan av utvecklingen. Man kan lätt hantera den grafiska programmeringen och samtidigt hantera variabler på ett textbaserat sätt som har visat sig vara väldigt svårt att göra i ett strikt visuellt programmeringsspråk (Song, 1994). En kombination är således att föredra, där respektive del används i den situation där de bäst lämpar sig (Navarro-Prieto & Cañas, 2001).

Den andra huvudsakliga aspekten är att visuell programmering har visat sig vara mer tidseffektiv än textbaserad programmering. Programmerare som använder ett visuellt programmeringsspråk utför mer än de som använde ett rent textbaserat språk (Baroth & Hartsough, 1994). De ska även öka hastigheten på programmeringen då man inte behöver kompilera och köra programmet för att se visuella förändringar (Ahmed, 1999). Snabba utvecklingstider leder även till fler fördelar. Ottersten & Berndtsson (2002) belyser vikten av att testa så tidigt i utvecklingsfasen som möjligt när det gäller designen av mjukvara. På så sätt integreras användaren i utvecklandet på ett naturligt sätt och risken för att slutprodukten inte är till användarnas belåtenhet i slutändan minskar. Visuell programmering möjliggör snabb framtagning av prototyper. Tidig testning och framtagning av prototyper möjliggör att man kan rätta fel tidigt i utvecklingsprocessen vilket minskar kostnaderna för projektet i slutändan (Ahmed, 1999).

Ytterligare en aspekt som man gärna vill se hos programmeringsspråk är felfaktorn. Det finns

tester som visar på att visuella programmeringsspråk har lägre felfaktor än de textbaserade. Nedan (tabell 4.1) finner ni resultaten från en studie gjord på ett helt visuellt programmeringsspråk, Forms/3 som bygger på formler för celler och formulär samt två textbaserade språk, Pascal som är ”blockbaserat” och OSU-APL som bygger på listhantering (Pandey & Burnetts, 1993). Testerna innehåller två problem som ska lösas med vart och ett av de tre programmeringsspråken. Problemen fokuserar på matris- och vektorhantering och ska fånga konstruktionen på språken. Nedan i tabell 4.1 finner ni de sammanställda resultaten från testerna.

	completely correct	nearly correct	conceptually but not logically correct	incorrect
Pascal	45	6	25	44
Forms/3	88	9	9	14
OSU-APL	64	6	8	42

Tabell 4.1 Test på felfaktorn hos VPL (Pandey & Burnetts, 1993)

Som framgår från tabell 4.1 ovan så minskar felfaktorn vid användandet av det visuella programmeringsspråket Forms/3 gentemot de andra språken.

4.1.1.2 Visual Basic som visuellt programmeringsspråk

Visual Basic är ett visuellt programmeringsspråk tack vare sin IDE Visual Studio. Det är dock vad Ahmed (1999) skulle kalla ett delvis visuellt programmeringsspråk. Visual Basic innehåller en del som är visuell. Där kan man hantera fönster, knappar, rullistor och annan grafik. De underliggande funktionerna däremot, är fortfarande textbaserade. D.v.s. man skapar t.ex. en variabel eller en loop med hjälp av text. Den skapade loopen möjliggör sedan t.ex. ifyllnad av den rullista som man skapade visuellt. Nedan finner ni några bilder tagna ur Visual Studio 2005 för att visa hur det är uppbyggt med kombinationen av visuell programmering och textbaserad programmering.

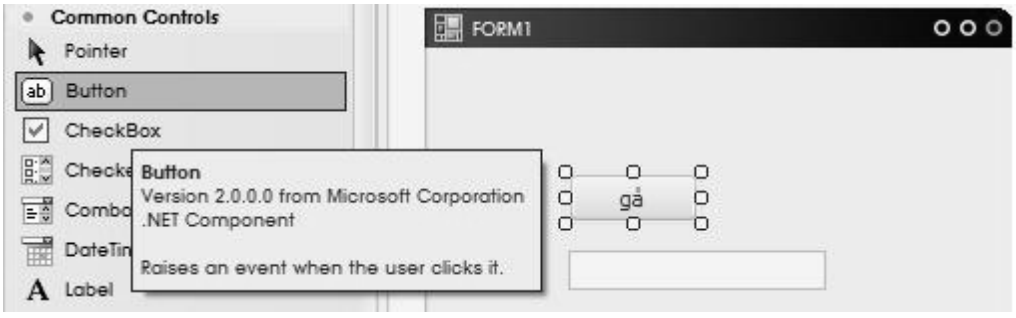

Fig. 4.1 Visuell Programmerings ”drag n’ drop” (egen)

I fig. 4.1 kan man lägga till en knapp visuellt med s.k. ”drag and drop”. I denna del av programmet kan man även ändra namn, text, storlek, positionering m.m. på knappen. Det här sker således visuellt.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim tmp As Integer
    tmp = CType(TextBox1.Text, Integer)
    aPerson.walk(tmp)
End Sub
```

Fig. 4.2 lägger till funktion till en knapp (egen)

I fig. 4.2 anger vi knappens funktion, d.v.s. vad knappen ska göra. Den här delen sker således med hjälp av text.

Vi har således funnit att Visual Basic 2005 är ett VPL och att visuell programmering är en egenskap hos Visual Basic 2005. Vi har även funnit en rad konsekvenser med att använda ett VPL. Vi kommer senare i arbetet (kap. 4.1.2) även belysa Rapid Application Development (RAD) som till stor del bygger på VPL och i sin tur medför några ytterligare konsekvenser. Man bör utvärdera Visual Basic som ett VPL och se de fördelar respektive nackdelar som det möjligtvis kan bidraga med till ens organisation innan man väljer att göra en övergång. Vi har ovan presenterat några av de konsekvenser som det kan ha, men det utesluter inte att det finns flera aspekter som är mer specifika för en organisation. Dock finns det konsekvenser med att använda ett VPL som visat ovan och det kan därför ha inverkan på valet av övergång eller ej.

4.1.2 Rapid Application Development

Ovan beskrev vi Visual Basic 2005 och dess egenskaper som VPL. En av de största möjligheterna var att det gick att utveckla produkter snabbt. Den här snabba utvecklingen leder dock inte endast till något positivt. Microsoft kallar Visual Studio för ett Rapid Application Development verktyg (RAD verktyg). RAD är egentligen en metod (Avison & Fitzgerald, 2006) men det finns verktyg som möjliggör denna metod. Denna typ av verktyg har en del fördelar och nackdelar som vi ska visa på nedan. Fördelarna och nackdelarna har vi funnit i mångt och mycket vara detsamma som hos metoden RAD men de kan skilja sig i abstraktion och till viss del även helt och hållet. Den här uppsatsen och detta kapitel behandlar dock RAD endast som ett verktyg och inte en metod. Eftersom RAD inte är den enda metoden som kan användas om man har Visual Basic som utvecklingsverktyg så hade vi annars behövt belysa fördelar och nackdelar med samtliga möjliga metoder, vilket vi inte har funnit ha någon inverkan på valet av övergång eller ej.

4.1.2.1 RAD och dess inverkan

Det finns både fördelar och nackdelar med att använda sig av RAD tekniker. En av de främsta nackdelarna må vara bristen i kontroll och dynamik i RAD verktygen. RAD verktyget måste vara väl anpassat efter projektets mål och kriterier. RAD verktyg har t.ex. svårt att anpassas till situationer där man måste skapa stora komplexa och unika system där utvecklaren måste ha stor kontroll över strukturen. (Zubeck, 1997)

Zubeck (1997) tar upp en metafor om Schweiziska urverk och system där det skulle uppstå problem att skapa ett system med så invecklad arkitektur som ett Schweiziskt urverk med ett RAD verktyg om det inte fanns en samling korrekt utformade kugghjul för varje enskild klocka som ska byggas. Utvecklaren tappar således fördelarna med RAD verktyget om han inte kan använda sig av dess färdiga funktioner. Verktyget kommer tvärtemot att motarbeta utvecklaren då han får det svårt att själv bygga de kugghjul som behövs, han har svårt att kontrollera den underliggande strukturen. Den underliggande strukturen är av stor vikt för systemet. Om något är fel i systemets grund så är det svårt att vidareutveckla systemet (Hough, 1993). Det finns även en delad mening om huruvida RAD är en bra teknik för nya utvecklare eller inte. RAD verktyg är lätta att ta till sig i alla fall om man håller sig till de färdiga komponenterna och teknikerna som finns integrerade. Ska man dock börja skapa egna objekt och komponenter blir det lite mer komplext och inte riktigt lika lätt för en ny programmerare att använda (Zubeck, 1997). RAD verktyg har även visat sig vara dåligt kompatibla med ”ovanlig” hårdvara och mjukvara. Sådana produkter som helt enkelt inte har tagits i beaktning vid utvecklandet av själva verktyget i sig. RAD verktygen tappar då sin fördel gentemot t.ex. C språken eftersom mycket tid måste läggas på att kringgå dessa problem och att utvecklingstiden då inte blir lika kort. De kan även ha liknande problem med olika typer av DBMS (Database management systems) beroende på vem som har utvecklat RAD verktyget (Zubeck, 1997). Det är även viktigt att identifiera projektets mål, och att det stämmer väl överens med vad RAD verktyget kan åstadkomma. Om inte det görs måste mycket tid läggas åt att tvinga verktyget att agera på ett sätt som det från början inte är tänkt att göra (Zubeck, 1997). Om man utvecklar system i en isolerad miljö, kanske i form av en komponent till ett redan befintligt system så finns det en fara med att använda RAD verktyg. Viktigt är då att man fokuserar på att standardisera utvecklingen. Det råder även delade meningar om huruvida RAD tekniker kan användas vid utvecklandet av system med krav på hög säkerhet p.g.a. svårigheterna att bibehålla kontrollen över sin utveckling (Zubeck, 1997) . Så för att sammanfatta de kritiska momenten med RAD så kan det möjligen uppstå problem med system som ska; hantera stora kvantiteter av data, utveckla system med krav på hög säkerhet, använda sig av ”ovanlig” hårdvara och vissa DBMS, utvecklas i en isolerad miljö, använda komponenter som inte återfinns i RAD verktyget. Vi sammanfattar dessa i att RAD verktyg kan innebära problem om man ska utveckla system där utvecklaren behöver ha stor kontroll över underliggande struktur och där han inte kan använda sig av de färdiga komponenterna som finns presenterade i verktyget.

Trots de många nackdelarna med RAD så har vi även identifierat minst lika många fördelar. Ett problem är dock att dess fördelar inte är så väl dokumenterade (Zubeck, 1997). Den största fördelen vi har hittat är dock det faktum att utvecklingstiden kan minskas. Att man kan utveckla system snabbt medför en rad olika fördelar förutom det faktum att det kan minska utvecklingskostnaderna. En stor fördel är att möjligheten att utveckla ett system eller en komponent som ett direkt svar på ett krav från en organisation, antingen som svar på en organisatorisk förändring, användarbehov eller behov som inte kan bli förutspådda lång tid i

förväg.(Hough, 1993) Utvecklingen går snabbt och det krävs snabba systemutvecklingstekniker för att hålla jämn takt. Med en kort utvecklingstid kan man leverera produkter som kan utvecklas med tiden. Faran med system som utvecklats under lång tid är att deras målorganisation har förändrats under utvecklingstiden och att deras behov inte längre är desamma. Det här kan även sänka underhållskostnaderna eftersom systemet blir mer anpassat efter situationen och färre korrigeringar måste göras. Risken för att s.k. ”runaway project” minskar, d.v.s. system som drar över på såväl tid som pengar (Hough, 1993). Kortare utvecklingstid medför även närmare kontakt med användaren (Subramanian & Zarnich, 1996). Snabb utveckling medför möjligheter att snabbt ta fram prototyper. Att snabbt kunna ta fram prototyper kan dels underlätta för integrering av användaren i utvecklingsprocessen samt utveckla projekt med oklara mål och hög innovativitet. (Hough, 1993)

4.1.2.2 Visual Basic 2005 som ett RAD verktyg

Som vi har nämnt tidigare under produktpresentationen så är Visual Basic (numera p.g.a. av Visual Studio) ett RAD verktyg (Harkness et. al. (2007), Zubeck (1997), MSDN, (2007 a). Visual Basic möjliggör snabb utveckling bl.a. med hjälp av den tidigare nämnda VPL tekniken och dess integrerade verktyg. Man kan exempelvis på samma sätt som man la till en knapp lägga till trådhantering. Men även klassbibliotek hos .NET medför många färdiga komponenter som snabbt går att implementera. Eftersom RAD verktyg har en del konsekvenser så är det enligt oss en egenskap att studera när man väljer att göra en övergång till Visual Basic 2005 med kringliggande miljö. Det kan innehålla en rad fördelar såsom snabb utveckling, protoyping m.m. Men det finns även risker såsom kontroll på strukturen och liknande. Man bör således kontrollera och överväga de fördelar som RAD verktyg har mot nackdelarna och se om det är bra lämpat för organisationens krav på utvecklingsmiljön.

4.1.3 Objektorientering

Under produktpresentationen i kapitel 3 nämnde vi att Visual Basic 2005 är ett objektorienterat programmeringsspråk. I litteraturen har vi funnit att det finns en rad konsekvenser med att använda objektorienterade språk, därför anser vi det vara av vikt att presentera fördelarna och nackdelarna med objektorientering och vad de kan ha för inverkan på valet av migrering eller ej. Nedan finner ni en kort resumé av vad objektorientering innebär samt vad det har för konsekvenser och varför det är en egenskap att ta i beaktning efter vad vi har funnit i litteratur.

4.1.3.1 Objektorientering och dess inverkan

Objektorientering är numera ett välkänt begrepp inom programmeringskretsar. Men precis som i fallet med RAD så är objektorientering såväl en metod som en programmeringsteknik. Man kan använda objektorienterade metoder såsom OOAD med modelleringstekniken UML såväl som objektorienterade programmeringsspråk, t.ex. Visual Basic 2005. För att definiera objektorienterade programmeringsspråk använder vi oss av ett citat:

> ”Object Oriented Programming is a method of implementation in which programs are organised as a co-operative collection of objects, each of which represents an instance of some class, and whose classes are all embers of a hierarchy of classes united via inheritance relationships.”
>
> Booch (1991)

Objektorientering bygger alltså på objekt som alltid är en instans av en klass. Dessa klasser i sin tur är relaterade till varandra via t.ex. arv. Ett objekt är en abstraktion av ett verkligt objekt (Avison & Fitzgerald, 2006). I fig. 4.3 finns ett schema för att förenkla beskrivningen något.

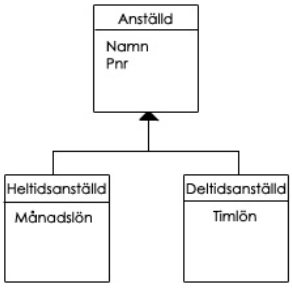

Fig. 4.3 Arvsprincipen (egen)

Det objekt som lagrar data, utför funktioner och relaterar till andra objekt (Yourdon & Argila, 1996). Vi ska inte göra någon djupare beskrivning av objektorientering då vi anser att det skulle leda oss från vårt verkliga syfte med uppsatsen men en kort beskrivning kan ändå vara till hjälp för förståelsen av denna faktor.

Det faktum att man kan ärva ner attribut och kod från klasser är en av objektorienteringens största fördelar. Man talar här om återanvändande av kod. Vi återgår till exemplet med de anställda. Här slipper man helt enkelt skriva om den kod som finns i klassen anställd i klasserna heltidsanställd och deltidsanställd eftersom de klasserna ärver samtliga av anställdklassens kod. (Avison & Fitzgerald, 2006). Det här kan även vara användbart för rent databaserade objekt. Ett fönster i Windows är ett exempel på ett objekt. Varje specifik typ av fönster ärver i sin tur av en superklass för fönster så de agerar på samma sätt. Det hindrar inte specifika fönster att innehålla alldeles egna funktioner, men exempelvis kanske samtliga fönster ska innehålla maximera, minimera och stäng funktionerna och dessa kan då ärvas från superklassen för fönster. Vill vi sedan ta bort funktionen för att maximera fönster på samtliga fönster räcker det med att vi tar bort den i superklassen. Eftersom de underliggande fönsterklasserna ärver av superklassen så kommer maximerafunktionen att försvinna från samtliga fönster. Samma sak kan man göra med tidigare presenterade scenario med anställda. Objektorienterade programmeringsspråk levereras ofta med ett färdigt bibliotek av klasser som kan användas vilket medför att man slipper skapa kod för många funktioner och liknande som redan är integrerade i språket. Många menar även att detta i framtiden kommer utvecklas till att man kan köpa redan färdiga objekt som passar ens behov och således korta ner utvecklingstiden (Avison & Fitzgerald, 2006). Dessa färdiga objekt ska således vara väl testade och problemfria, sammantaget ska det leda till en snabbare och

problemfriare utveckling, vilket även leder till färre kostnader. (Avison & Fitzgerald, 2006, Balevičius et. al., 2006). Anledningen till det är bl.a. att arkitekturen är lättare att följa. Av samma anledning så ökar även underhållbarheten på programvaran (Balevičius et. al., 2006).

Objektorientering är dock inte endast guld och gröna skogar, det har sina baksidor. Det har främst visat sig finnas nackdelar när man pratar om system i större skala. Objektorienterad programmering kan ha svårigheter att behandla återanvändning när det kommer till system i större skala (Ossher & Tarr, 1999). För att objektorientering ska vara användbart kräver det en klar identifiering och undersökning av målorganisationen samt planering (Avison & Fitzgerald, 2006), det här är framförallt viktigt vid större projekt (Ossher & Tarr, 1999). Vidare problem kan vara att samtliga utvecklare måste utgå från samma modell när de utvecklar, de kan således inte skapa en modell av problemet som passar deras situation. En objektorienterad modell måste stämma överens med hur det verkligen ser ut som vi har nämnt ovan och kan således inte skilja sig utvecklare emellan, samtliga utvecklare måste alltså ha full förståelse om hur relationerna i modellen fungerar (Ossher & Tarr, 1999). Ovan nämnde vi att objektorientering är bra för att det är lätt att integrera nya delar i ett system. Det råder däremot ett visst tvivel om huruvida det här stämmer eller ej när det gäller större system. Ossher & Tarr (1999) menar att det kan vara svårt att bygga ut system utan att behöva göra ändringar i det redan befintliga systemet. Det finns försök till att lösa det här problemet, och en kraftfull metod inom objektorienteringen är att använda sig av mönster. Utvecklaren följer mönster i sin utveckling som möjliggör att systemet blir lättare att underhålla (Gamma et al., 1993). För att förklara begreppet mönster använder vi oss av exemplet MVC. MVC står för Modell-View-Controller och bygger på att man särar på själva modellkomponenten där alla objekt och dess metoder finns lagrade med gränssnittet och de faktiska funktionerna. Man kan således byta ut hela gränssnittet utan att behöva ändra bland sina objekt.

Mönster kan dock även vara komplicerade och invecklade att förstå och bör därför endast användas när de verkligen behövs (Gamma et al., 1993). Mönster är inte heller en komplett lösning på de problem vi har tagit upp tidigare utan många problem kvarstår (Ossher et al., 1996).

Det finns dock ytterligare en nackdel med objektorienterade programmeringsspråk, de använder mycket CPU-tid. En studie gjord på procedural FORTRAN och C++ visade att program skrivna i FORTRAN var 6.5 gånger så snabba som samma program skrivet i C++ vid körning. (Balevi˘cius et. al., 2006)

4.1.3.2 Visual Basic 2005 som objektorienterat programmeringsspråk

Visual Basic har även med tidigare versioner försökt gå mot en objektorienterad programmering men har i och med Visual Basic 2005 gått mot ett verkligt objektorienterat språk. Ett tecken på detta är ovan nämna arvsprincipen. I tidigare versioner av Visual Basic har man använt sig av objekt och klasser i viss mån, men arvsprincipen har aldrig tidigare integrerats (Evjen et. al., 2006 & Harkness et. al., 2007). Visual Basics nya underliggande ramverk .NET 2.0 är en ny form av Windows API som är även det objektorienterat. Dock är det inte endast arvsmöjligheterna som gör Visual Basic 2005 till ett objektorienterat språk. Abstraktion, polymorphism, och enkapsulation är ytterliggare möjligheter som karakteriserar objektorientering och återfinns i Visual Basic 2005 (Harkness et. al., 2007).

Visual Basic 2005 är således ett verkligt objektorienterat programmeringsspråk. Vi har också identifierat en rad för- och nackdelar med objektorientering. Vi hävdar således att det här är en viktig egenskap att fokusera på efter vad vi har funnit i litteraturen.

Precis som med många tidigare punkter är objektorientering varken en positiv eller negativ faktor. Man måste se till sin egen kontext och säkerställa om objektorientering är något man söker eller ej. Faktorn objektorientering är däremot ett faktum då den kan påverka utvecklingen, positivt eller negativt.

4.1.4 Syntax

När vi skrev vår produktpresentation i kapitel 3 presenterade vi Visual Basic som ett lätt språk att ta till sig och använda exempelvis tack vare VPL egenskapen. Vi ställde oss då frågan vad det är som gör det lättare. Vi fann en del litteratur om huruvida syntaxen hos ett språk har inverkan, skillnader i syntaxen som gör att ett språk är bättre eller lättare än ett annat och vilka dessa skillnader är. I våra intervjuer fann vi ytterliggare information om att syntaxen kan vara av importans i vissa situationer. Vi har således funnit att syntaxen kan ha inverkan på valet av övergång eller ej och under det här kapitlet ska vi presentera varför.

4.1.4.1 Syntax och dess inverkan

Vi har funnit en hel del litteratur angående syntax påverkan på utvecklingen. Men även i våra intervjuer har vi funnit att syntax kan han inverkan på valet.

Man kan fokusera på tre punkter på ett språks syntax. Skrivbarhet (writeability), läsbarhet (readability) samt säkerhet (reliability). De här tre punkterna har således påverkan på valet av programmeringsspråk. (Ghezzi & Jazayeri, 1987)

Med skrivbarhet menas helt enkelt hur pass lätt språket är att skriva i. Det finns flera saker hos en syntax som kan ge den hög skrivbarhet. En viktig faktor är hur pass lätt språket är att lära sig och skriva. Med lätt menar vi inte bara att det ser lätt ut det kan även vara saker såsom möjligheter att ”overloada” metoder för att slippa memorera många metodnamn och funktioner, återanvändning av kod o.s.v. Tvivel råder även huruvida man ska ha många möjligheter att uttrycka samma sak eller få för att minska komplexiteten hos språket. Skrivbarheten är av importans då hög skrivbarhet leder till att utvecklaren kan fokusera på de logiska och faktiska problemen snarare än hur man ska lösa dem rent programmeringsmässigt. (Ghezzi & Jazayeri, 1987) Vi har tidigare varit inne på Visual Basic 2005s skrivbarhet, man stödjer exempelvis overloading, återanvändning samt automatisk generering av kod för exempelvis knappar. Däremot anser vi att det är en smaksak om huruvida man anser skrivbarheten vara hög eller inte, och låter därför det vara osagt. Vi konstaterar däremot att man bör titta på skrivbarheten.

Ytterliggare ett effektivitets perspektiv på programmeringsspråk är läsbarhet. Med läsbarhet menas hur pass enkelt språket blir att läsa och följa. Läsbarheten är ofta starkt kopplat till

skrivbarheten, hög skrivbarhet leder således oftast även till hög läsbarhet. (Ghezzi & Jazayeri, 1987). För att förklara begreppet mer ingående finner ni ett utdrag från Visual Basic 2005 nedan.

```
Public Class Person
        Private name As String
        [...]
        Public Sub SetName(ByVal tmpName As String)
            name=tmpName
        End Sub

        Public Function GetName() As String
            Return name
        End Function
        [...]
End Class
```

Ovan ser ni en klass som heter person med attributet namn. Ni finner även två metoder som används för att ange samt hämta namnet på en person. Det är således ganska lätt att läsa vad klassen kan göra samt vad de olika metoderna har för funktion då man använder sig av endast ”vanlig” text – det är hög läsbarhet, om man kan engelska. Kan du inte engelska är det kanske väldigt svårt att läsa och läsbarheten väldigt låg. Man bör således kolla på syntaxens läsbarhet för att se att den stämmer överens med ens krav. Läsbarhet är viktigt för att man ska kunna återgå till sin kod, bygga om och bygga ut den. Det kan även underlätta förståelsen för de funktioner man har skapat, både för andra programmera men även för en själv om det är ett stort projekt. (Ghezzi & Jazayeri, 1987**)**

Säkerhet är relaterat till både skrivbarheten och läsbarheten. Desto lättare det är att skriva med en syntax desto lättare är det att veta att man har gjort rätt. Samma sak gäller läsbarheten – desto lättare vi kan läsa vad vi har gjort, desto lättare är det att utröna huruvida man har gjort rätt eller ej. (Ghezzi & Jazayeri, 1987)

Olika språk har olika syntax, och kan således ha olika skrivbarhet, läsbarhet och säkerhet. Syntaxen är därför något man bör ta i beaktning när man väljer programmeringsspråk efter vad vi har funnit i litteratur.

4.2 Faktorer identifierade i produkten

Som sagt i vårt metodkapitel (kapitel 2) så har vi letat efter faktorer även i själva produkten. Efter att ha studerat produkten och dess kringliggande miljö i litteratur har vi funnit att vissa saker kan ha inverkan på valet. Under den här delen av kapitlet ska vi presentera vad vi har funnit i själva produkten Visual Basic 2005 och dess kringliggande miljö som kan ha inverkan på

valet.

4.2.1 Kompatibilitet

Vi har funnit efter att ha granskat Visual Basic 2005 i litteratur att den kan ha vissa kompatibilitetsaspekter som vi ska presentera nedan.

Program som är kompilerade i .NET kan exempelvis endast köras på Windowsplattformar (Harkness et. al., 2007), vilket begränsar kompatibiliteten. Vi har även visat under produktpresentationen i kapitel 3 att .NET innehåller en rad nya funktioner som ska öka kompatibiliteten med såväl hårdvara som olika typer av DBMS. Windows har även utvecklat språket CLR som gör att deras olika .NET baserade språk är kompatibla med varandra. Man kan således se att det finns både negativ kompatibilitet i form av begränsning av plattformen man kör sin utvecklade produkt på samt en satsning i att göra språket kompatibelt med olika typer av DBMS och programmeringsspråk.

4.2.2 Migreringsverktyg

För att minska migreringskostnaderna så har man tagit fram olika verktyg som automatiskt ska konvertera ursprunglig kod till Visual Basic 2005 kod och då korta ner migreringstiden (Gowthaman et. al., 2005). Dessa verktyg är inget som alltid finns tillgängligt men i vissa fall finns det. Det finns således inget generellt verktyg som Microsoft tillhandahåller från alla programmeringsspråk till Visual Basic 2005. Det är inte heller bara Microsoft som tillhandahåller dessa verktyg. Vi har funnit att det finns tredjepartstillverkare som tillhandahåller olika verktyg för olika konverteringar och t.o.m. för olika stadier i migreringen (Good, 2002). Det krävs att man gör efterforskning för att se om möjligheten finns för just det unika fallet för att på så vis minska migreringstiden. Microsoft tillhandahåller dock verktyg för att konvertera från Visual Basic 6.0 till Visual Basic 2005 vilket finns integrerat i Visual Studio 2005. Man bör således inspektera sitt ursprungliga programmeringsspråk och se vad möjligheterna för migrationsverktyg är. Dock påvisar Microsoft (Microsoft Corporation, 2005) och även annan litteratur (Gowthaman et. al., 2005) att koden måste analyseras och omarbetas innan migreringsverktygen går att använda. Träning i själva migreringsverktyget kan komma att behövas så att det används på bästa möjliga sätt vilket Bergey et. al. (2001) belyser. Men det är inte bara själva omskrivningen av kod som är problemet vid migreringen utan även planering och struktureringen av den som är problematisk (Bisbal et. al., 1999). Det måste finnas en klar strategi vid övergången till .NET (Microsoft Corporation, 2005). Det vanligaste är dock att man inte använder sig av ett migreringsverktyg vid övergången till .NET. Detta kan dels bero på att det inte finns ett verktyg från det språk man kommer från eller p.g.a. att där finns en tro om för stor komplexitet i själva migreringsverktyget (Gowthaman et. al., 2005). Gowthaman et. als. (2005) studie visar dock på att det kan finnas en skillnad i migreringstid om man använder sig av ett migreringsverktyg eller inte. Användandet av ett migreringsverktyg kan påverka migreringstiden positivt.

4.3 Organisatoriska faktorer

I vårt metodkapitel (kapitel 2) så nämnde vi att vi sökte efter faktorer även på en organisatorisk nivå. Dessa faktorer eftersöktes inte endast i kring miljön för Visual Basic 2005 utan också kring migrering och modernisering av arbetsmiljöer och tekniker i allmänhet. Vi försökte därefter se om det fanns något som även var användbart i vår frågeställning om Visual Basic 2005. Vi fann i litteratur en del intressanta saker som kan ha inverkan på valet. Under den här delen av kapitlet kommer vi att visa på de faktorer vi har funnit ha inverkan på valet när vi sökte på organisatorisk nivå i litteraturen, d.v.s. hur organisationen påverkas och hur organisationen påverkar valet av övergång eller ej.

4.3.1 Personal

När vi utförde våra litteraturgranskningar lyckade vi identifiera fakta som behandlade personalen och hur denna kan ha inverkan på valet av en övergång till Visual Basic 2005 och dess kringmiljö eller inte. Enligt Hovland et. al., (2001) är Visual Studio en utvecklingsmiljö som tillhandahåller de senaste och mest sofistikerade teknikerna, vilket skulle möjliggöra en personlig utveckling för utvecklarna på organisationen som överväger en övergång. Harkness et. al. (2005) beskriver fördelarna med Visual Studio 2005 och påvisar på vilket sätt man kan anse att detta utvecklingsverktyg faktiskt är i framkant vad gällande teknik. Att kunna erbjuda sin personal ett stimulerande och tekniskt utmanande arbetssätt är viktigt för att bibehålla sin personal. Brist på att kunna göra detta är tämligen negativt för ett tekniskt företag då de i huvudsak förlitar sig på spetskompetens (Avison & Fitzgerald, 2006). Det är av Brill & Levine (2005) bevisat att det i människans natur ligger ett behov av att utveckla och lära sig nya saker, de kallar behovet för ”growth” och syftar till behovet av att utvecklas.

4.3.2 Konkurrens

Efter att ha granskat litteraturen på ett organisatoriskt plan fann vi även information i litteraturen om hur Visual Basic 2005 och dess kringliggande miljö kan ha en direkt inverkan på organisationens möjligheter till att konkurrera. Vi ska under det här delkapitlet presentera närmre vad vi har funnit. När vi pratar om konkurrens i det här kapitlet är det inte de konkurrensmöjligheter Visual Basic 2005 samt dess kringliggande miljö kan ge oss på en teknisk basis, d.v.s. nya möjligheter till funktioner etcetera. Utan den konkurrensfördel man kan få genom att endast använda sig av en uppdaterad utvecklingsplattform samt miljö, en sorts informell konkurrens. Att arbeta med en utvecklingsmiljö som är uppdaterad och ligger rätt i tiden samt är välkänd för kund inbringar en form av seriositet och eliminerar de konkurrenter som inte utvecklar i .NET (Blom & Ljung, 2004). Blom, J. & Ljung, J (2004) har i sina studier visat på att där finns krav från kunder på att använda tekniskt avancerade plattformar och miljöer så som .NET. Kunderna kan känna sig säkra i en utveckling på .NET plattform vad gällande teknik och funktionalitet då Microsoft är en stabil leverantör vars .NET plattform troligtvis inte inom en snar framtid kommer upphöra att vara konkurrenskraftig, vilket in för kund även kan

innebära en viss trygghet och seriositet (Blom, J. & Ljung, J., 2004), (Hovland, 2001). Vidare konkurrensaspekter som lokaliserats i litteraturen är den att företag som ligger tekniskt långt fram skapar konkurrensfördelar som säkerställer organisationens position på marknaden samt dess rykte av att vara ett tekniskt kompetent företag (Hatch, 2002). Hatch (2002) är inte den enda som konstaterat denna aspekt, även Falk & Olve (1996) beskriver i sin bok ”IT som strategisk resurs” att många företag idag konkurrerar med hjälp av deras grad av teknisk utveckling. Blom et al., (2004) visar igenom sina studier i form av intervjuer med företag som sysslar med utveckling huruvida de känner sig manade att använda sig av den senaste tekniken för att inför kund visa att de besitter den senaste kunskapen. Detta visade sig också vara en faktor vid övervägandet om de skulle gå över till .NET. Vi har vidare genom Blom et. als. (2004) intervjuer med utvecklingsföretagen kunnat utröna att utvecklingen som sker i Visual Studio på en .NET plattform anses på marknaden vara hållbara och pålitlig då Microsoft levererar helhetslösningar som är väl kompatibla med varandra.

4.3.3 Inlärning

I den litteratur vi tagit del av har vi funnit en del fakta om inlärningens påverkan vid övergången till Visual Basic 2005 samt dess kringmiljö. Vi kommer nedan att presentera den fakta vi har funnit.

Hur anpassar sig en ny programmerare till ett nytt språk, vilka effekter det kan ge och vad man bör tänka på? Detta är något som vi anser vara viktigt att ta reda på innan man gör en övervägning om migrering till ett nytt språk samt en ny utvecklingsmiljö då vi funnit belägg för detta i den litteratur som behandlar ämnet t.ex. Scholts & Wiedenbeck (1993).

Då en erfaren programmerare sätter sig ner vid en ny miljö har denna person en väldigt hög nivå av tidigare kunskaper om hur problem bör lösas i form av konceptuella modeller, taktiker, strategier samt planering. Dessa kunskaper underlättar självklart arbetet om man jämför med en novis inom programmering som ligger på samma nivå som den erfarna programmeraren när man enbart ser till syntaxen inom det nya språket, dock besitter inte den oerfarna programmeraren samma erfarenheter vad gäller strategier och egna modeller. (Scholts & Wiedenbeck, 1993)

Det som kan innebära problem för en erfaren programmerare är att denne har förutfattade meningar om hur vissa operationer är konstruerade och förutsätter således att de är konstruerade på ett liknade sätt även i det framtida programmeringsspråket. Detta kan innebära att det tar längre tid för utvecklaren att lära sig uppbyggnaden av det nya språkets då han först måste komma ur sitt tidigare tankesätt.(Scholts & Wiedenbeck, 1993)

Erfarna programmerare börjar med att bygga upp en mental bild av problemet som ska lösas, denna bild guidar dem genom arbetet och fram till en tänkt lösning. Problemet som uppstår när en erfaren programmerare skapar denna typ av mental bild är att den är starkt kopplat till det föregående programmeringsspråket, vilket möjligtvis inte stämmer överrens med sättet att lösa ett visst problem i den nya utvecklingsmiljön (Sholts & Wiedenbeck, 2003). Programmerare förlorar oftast sina rutinmässiga mönsterlösningar som denne besatt i tidigare utvecklingsmiljö. (Guindon, Krasner & Curtis, 1987)

När man talar vidare om inlärningen står det vidare klart att programmerare behöver hjälp vid uppstartandet av ett nytt språk då det tar längre tid att lära sig själv kontra att ha någon som kan tillhandahålla assistans (Scholtz & Wiedenbeck, 1993). Assistans som ges från kollegor innebär en betydande reduktion av den tiden det tar att lära sig ett nytt språk (Scholts & Wiedenbeck, 1993). Således kan det vara bra om någon i organisationen har kunskap om utvecklingsmiljön/språket man ska migrera till.

Vi har identifierat problem programmerare stöter på när det kommer till själva planeringen av en lösning i ett nytt programmeringsspråk. Man menar att planeringen på en strategisk nivå troligtvis är densamma och behöver därför inte bearbetas vid ett byte mellan miljöer men att planering av själva implementering skiljer sig väldigt mycket, det har visat sig att det är just här en hel del av den totala inlärningstiden spenderas då det oftast innebär en omfattande genomsökning i dokumentation (Scholtz & Wiedenbeck, 1993). Det som även kan innebära problem är att man i forskning återfunnit oviljan att lära sig planeringen och problemlösningen i det nya språket om man på ett väl fungerade sätt kan implementera ett gammalt sätt att lösa problemet i den nya miljön. Detta resulterar i att det nya språkets alla fördelar inte kommer till sin rätt och miljön blir inte så kraftfull som den möjligtvis kan vara och vad är då meningen med övergången? Utvecklarna börjar i flertalet av fallen i studien utförd av Scholtz & Wiedenbeck (1991) med att leta om det finns en liknande lösning i det nya språket som de använt sig av i det äldre språket och väljer här oftast bort dokumentationen om det nya språket som möjligtvis kunde givit dem en kraftfullare och bättre lösning.

4.4 Kapitelsammanfattning

Vi kommer nu att sammanfatta den fakta vi presenterat under kapitel fyra detta görs för att lättare få en överblick från kapitlet samt på ett bättre sätt kunna få med sig informationen till fortsatt läsning. Först inleder vi med att beskriva de egenskaper hos Visual Basic 2005 som kunde påverka valet, sedan sammanfattas de faktorer vi fann både hos produkten samt på den organisatoriska nivån.

4.4.1 Egenskaper hos produkten

VPL: Vi har lyft fram fakta som säger att Visual Basic 2005 kan användas som ett visuellt programmeringsspråk även kallat VPL. Detta innebar att man t.ex. har knappar man kan dra in i applikationen istället för att själv sitta och skriva kod som möjliggör en viss typ av knapp. Det påstås även att det skulle gå snabbare att programmera på detta vis. Dock så visualiseras inte allt, loopar och listor får man fortfarande skriva manuellt

RAD: Vidare har vi presenterat något som kallas RAD som står för rapid applikation development. Microsoft med flera kallar sitt Visual Studio ett RAD-verktyg som möjliggör snabb programmering. Här kan dock problem uppstå, kontrollen minskar om man använder sig av RAD-tekniker. RAD har en stark koppling till VPL då VPL är en av teknikerna som möjliggör RAD.

OO: Objektorientering är även det något som bli har belyst i tidigare text. Vi belyser fördelar med OO via arv, enkapsulation samt de färdiga klassbiblioteken som de objektorienterade språken tillhandahåller. Dock finns där en del problem t.ex. risken av hög koppling mellan olika klasser och objekt, vilket inte alltid är bra att ha när man utvecklar applikationer.

Syntax: Denna punkt sammanfattar vi genom tre punkter som har inverkan på valet, skrivbarhet, läsbarhet, säkerhet. Dessa kan inverka på hur lätt resp. svårt man tycker språket är att förstå samt skriva.

4.4.2 Faktorer identifierade i produkten

Kompatibilitet: Vi visar här på de möjligheter Visual Basic 2005 har med att kopplas samman med andra programmeringsspråk samt olika databaser.

Migreringsverktyg: Verktyg som kan hjälpa till att konvertera gammal kod till Visual Basic 2005. Anledningen till att man använder sig av denna typ av verktyg är för att minska konverteringstiderna. Studier har dock visat på att skillnaden mellan att använda sig av ett verktyg och skriva om koden från början är väldigt liten om man ser ur ett rent tidsperspektiv.

4.4.3 Organisatoriska faktorer

Personal: Denna faktor ser till vilken inverkan personalen har på en möjlig övergång. Vi finner här att personalen strävar efter personlig utveckling och en ny miljö bidrar till denna utveckling.

Konkurrens: Vi visar här på hur företag som använder sig av en uppdaterad miljö har en strategis fördel på marknaden då de framstår som tekniskt avancerade och kunniga om de använder sig av den senaste tekniken.

Inlärning: Denna faktor påverkar programmeraren sett till inlärning, en ny miljö innebär att man måste lära sig ett nytt sätt att programmera vilket tar tid, samt leder inledningsvis till effektivistesminskningar.

5 Empiri

Som vi har nämnt under metodkapitlet (kapitel 2) så använder vi oss främst av expertintervjuer vilka inte ger oss några empiriska data. Men våra intervjuer har ändå gett oss information om vad det är som påverkar valet av övergång eller ej. Vi anser att det här är viktigt att belysa då det är viktigt att ta reda på vad det är som organisationer söker hos en övergång. Vad som eftersöks i en produkt för att en övergång ska vara motiverad. Respondent C däremot har gjort en övergång och viss information som återfinns i hans intervju (Bilaga 3) kan ses som empirisk data. Under det här kapitlet ska vi presentera vad vi fann i intervjuerna. Vi har använt oss av samma upplägg som under litteraturanalysen för att bibehålla strukturen på uppsatsen.

5.1 Egenskaper som inverkar på valet

5.1.1 VPL

Under vår litteratur genomgång presenterade vi egenskapen VPL och hur den kunde ha inverkan på valet. VPL är även identifierat i våra intervjuer till viss del. I intervjun med respondent C framgår det att han tyckte de grafiska funktionerna för att rita gränssnitt i Visual Studio var en förbättring mot tidigare versioner av Visual Basic. Han anser att miljön var effektivare och innehöll många bra funktioner som kunde påverka hans utveckling positivt. Han ansåg att det innebar en ökning i produktiviteten att använda det nya verktyget och att han efter inlärning av det kunde arbeta snabbare än vad han gjort tidigare. Respondent C anser således att miljön var en förändring som kunde ha inverkan på valet.

5.1.2 RAD

Visual Basic är ett RAD verktyg och under analysen av litteratur fann vi att det hade vissa konsekvenser och kunde således ha inverkan på valet. I våra intervjuer är det inga som pekar på RAD specifikt, när man frågar dem angående begreppet så har de inte hört talas om det. Men om man läser mellan raderna så finner man att exempelvis respondent C eftersöker färdiga komponenter som man kan integrera i sin produkt. Finns det då olika komponenter som tillhandahålls av Visual Basic 2005 och dess kringliggande miljö som man kan använda så anser vi att man använder Visual Basic 2005 och dess kringliggande miljö som ett RAD verktyg. Man eftersöker i intervjuer således inte ett RAD verktyg specifikt men man eftersöker till stor del de funktioner som vi funnit i vår litteraturgranskning vara karakteriserande för RAD verktyg.

5.1.3 Objektorientering

Till skillnad från ovanstående egenskaper som intervjuobjekten inte nämnt ordagrant men som vi

ändå kunnat utröna efter att ha läst lite mellan raderna vara något som eftersöks så är objektorientering något som intervjuobjekten ordagrant eftersöker. Respondent C eftersöker objektorientering dels för att den tidigare nämnda arvsprincipen. Han ansåg även möjligheten att använda arvsprincipen vid gränssnittsutveckling var något väldigt positivt. Respondent C eftersöker arvsprincipen eftersom det kan öka hans utvecklingshastighet. Även respondent B eftersöker objektorientering då han anser att det medför snabbare och effektivare programmering tack vare dess återanvändbarhet.

Intervjuobjekten eftersöker också objektorientering p.g.a. dess inverkan på underhållbarheten. Respondent B tycker det är av importans hur lätt deras produkt är att underhålla och ser objektorienteringen som ett sätt att öka underhållbarheten. Respondent C ansåg att efter övergången tvingades han att programmera mer objektorienterat vilket ökade strukturen på hans kod och gjorde den lättare att underhålla vilket stämmer överens med vad respondent B eftersträvar. Respondent B anser också att han utveckling kommer bli lättare om språket är strikt objektorienterat till skillnad från hans nuvarande språk som inte är objektorienterat fullt ut.

Objektorientering är således något som intervjuobjekten ovan eftersöker då det kan medföra förbättringar i deras utveckling. I det här fallet är således objektorienteringen något som man ser positivt. Men respondent A**,** som har en mer organisatorisk post snarare än teknisk, är inte helt säker på att det kommer innebära någon större skillnad.

> ”[...] men jag vet inte om det lönar sig att göra det nu när koden finns där. Det är möjligt att det inte lönar sig att jobba mer på den punkten, men hade man kunnat göra det från början kan man tänka sig att det såg lite annorlunda ut! Men det är mer en teknisk fråga, jag tror inte vi mår illa av hur det ser ut nu.”
>
> Respondent A

> ”[...] nya möjligheter som jag inte är helt säker på att vi kommer utnyttja, just vad gällande klassprogrammeringen.”
>
> Respondent A

Denna ser det som att programmet fungerar, samt att kunden märker inte hur pass objektorienterat man har programmerat. Han har således inte fokus på programmeringsaspekter och ser därför inte objektorientering som något måste, han fokuserar istället på kunden och ser det som att objektorienteringen inte innebär någon skillnad för dem.

5.1.4 Syntax

I litteraturen fann vi att syntax kunde ha en viss inverkan på övergången. Även i våra intervjuer har vi funnit en del information om syntaxens inverkan på valet. Respondent C menar att läsbarhet har inverkan på strukturen och det är viktigt att språket är konsekvent så det blir att i efterhand läsa och följa. Läsbarhet var även något som visades i litteraturen möjligtvis kunna ha inverkan på valet. Men vi har även funnit i våra intervjuer med respondent B och respondent A att de har valt att övergå till senare version av samma språk p.g.a. att syntaxen har varit lik och att övergången således skulle bli mer smärtfri. Respondent B menar även på att det faktum att syntaxen har varit för svår att konvertera har lett till att de tidigare valt att inte göra en övergång.

Respondent A säger också i sin intervju att han anser konverteringssvårigheterna hos syntaxen är den viktigaste faktorn vid övergång för honom. Om man överväger en migrering från ett tidigare språk till Visual Basic 2005 eller egentligen vilket språk som helst, så kan det således vara nödvändigt att kolla på hur pass lik syntaxen är sitt tidigare språk för att minska inlärningstiden, som vi nämnt tidigare ha inverkan på valet. Det här perspektivet på syntaxen kan dels vara av importans vid en övergång men kan kanske även för att bevara de rutinmässiga tekniker och knep som programmeraren har för att lösa specifika problem med hjälp av sitt tidigare programmeringsspråk. Det här fenomenet har vi beskrivit i tidigare kapitel. Nedan finner ni ett exempel på samma loop i Java, Visual Basic 6 samt Visual Basic 2005.

Java:

```
[...]
for ( int i=1; i<6; i++ ) {
        System.out.println(i);
}
[...]
```

Visual Basic 6:

```
[...]
For i = 1 To 6
        Debug.Print i
Next
[...]
```

Visual Basic 2005:

```
[...]
For i=1 To 6
        Console.WriteLine(i)
Next
[...]
```

Skillnaderna är således färre mellan Visual Basic 6 och 2005 än mellan Java och Visual Basic 2005. Vi hävdar således att en person som tidigare har kodat i Visual Basic 6 har lättare att lära sig och konvertera sin kod till Visual Basic 2005 än en person med ett förflutet i Java.

Syntax är således något som man har fokuserat på hos våra intervjuobjekt men de fokuserar inte endast på det som vi funnit i litteratur utan ser även till hur pass lik syntaxen är sitt tidigare programmeringsspråk. Något som vi tidigare varit inne på under kapitlet inlärning.

5.2 Faktorer identifierade i produkten

5.2.1 Kompatibilitet

I litteraturanalysen visade vi att Visual Basic 2005 och dess kringliggande miljö påverkade kompatibiliteten både positivt och negativt. Kompatibilitet är även en stark faktor vi har funnit i intervjuer. I intervjun med respondent C så framgick det att ett av hans grundmål med en migrering var att deras produkt skulle vara framtidssäker och kompatibel med kommande versioner av t.ex. Windows. Respondent C visade även på en annan typ av kompatibilitet som var av vikt för valet, det var viktigt att produkten var kompatibel med olika färdiga komponenter och verktyg som hjälper till i utvecklingsprocessen som finns på marknaden. Språkstöd var ytterligare en kompatibilitets aspekt som kunde ha stor inverkan på valet. Respondent A anser det även vara viktigt att se hur pass kompatibel produkten är med Internetlösningar och Internetförbindelser.

5.2.2 Tekniska möjligheter

I intervjuerna finns det en förhoppning om att Visual Basic 2005 ska innehålla tekniska förbättringar jämfört med tidigare versioner. Respondent A anser att det bör finnas olika nya tekniska möjligheter med att uppgradera som inte har varit möjliga med tidigare versioner.

> ”[...] så det finns ju anledning till att se på möjligheterna, vad man kan få nytt så att säga. Det tycker jag är mest spännande egentligen.”
>
> Respondent A

Respondent A anser således att de tekniska möjligheterna jämfört med tidigare miljö är en viktig faktor vid valet huruvida han ska genomföra en migrering eller ej. Även i intervjun med Respondent C så framgår det han att såg många nya tekniska möjligheterna och finesserna som något positivt vid övergången. Respondent A och respondent C söker dock något olika tekniska möjligheter. Respondent A vill ha nya möjliga funktioner i sin slutgiltiga produkt för användaren, medan respondent C snarare söker nya tekniska möjligheter i utvecklingen, som i och för sig inte behöver vara irrelevant för slutanvändaren.

5.2.3 Migreringsverktyg

När vi genomförde våra intervjuer med de olika respondenterna var en av punkterna som framkom själva migreringsverktyget.

Samtliga respondenter vill minska migreringstiden så mycket som möjligt. De hoppas bl.a. på att det ska finnas migreringsverktyg som kan förkorta tiden och hjälpa dem vid en övergång. Respondent A säger i sin intervju att han väntar på någon ”smart person” att uppfinna ett verktyg

som de kan använda för att migrera deras kod och ser det nästan som ett krav för att göra en övergång. Anledningen är att minska kostnaderna för omskrivningen som respondent B säger:

> ”Att konvertera programmet är det ju ingen som betalar för, så där ligger en stor utgift, i utvecklandet.”
>
> Respondent B

Här visar intervjuobjektet att önskan om migreringsverktyg finns där ur ett rent kostnadsperspektiv. Vidare i intervjun med respondent B påstås det även att han tror det kommer att behövas träning i migreringsverktyget för att det ska gå att använda. I intervju med respondent B förklarar han att när de skulle utföra migreringen till .NET tittade de på användandet av just ett sådant verktyg men att verktyget inte ens kunde klara av att konvertera de enklaste små programmen. Detta kunde dock ha med väldigt specifika egenskaper i deras kod att göra, men han lägger ingen större tilltro till denna typ av verktyg.

5.3 Organisatoriska faktorer

5.3.1 Personal

Liksom i litteraturanalysen fann vi i intervjuerna att det fanns olika personalrelaterade aspekter med en övergång. Vi kan här se att en övergång till Visual Basic 2005 och dess kringmiljö påverkar personalen inom organisation på olika sätt. Respondent B på Organisation X menar att han vill utvecklas och detta är en av orsakerna till varför han vill gå över till Visual Basic och dess kringmiljö. Vidare genom intervjun finner vi information som påverkar personalaspekten på ett organisatoriskt sätt då vilket påvisas genom detta citat.

> ”Jag tror att det blir lättare att få hit folk om vi byter till 2005”
>
> Respondent B.

Respondent B identifierar här möjligheten att attrahera ny personal till organisationen, vid en övergång, vilket troligtvis är av vikt för en organisation. Respondent B beskriver att arbetet som utförs i dagsläget har pågått i samma utvecklingsmiljö i 7 år vilket enligt respondent B börjar bli lite tråkigt, dessutom ser han som utvecklare alla de brister den nuvarande utvecklingsmiljön har, men med tanke på dess höga ålder har leverantören slutat leverera ”buggfixar”, vilket kan vara enerverande för en utvecklare som hela tiden måste dras med samma fel utan att kunna göra något åt det, vilket även respondent B konstaterar. På frågan huruvida det blir trevligare att gå till jobbet svarar respondent B enligt följande.

> ”Ja naturligtvis, det är dels därför jag är intresserad av att gå över till Vb2005, som jag sa det är lite tradigt att köra i samma miljö fortfarande. Jag har kört den i 7 år nu så det börjar bli lite tradigare. Man behöver alltid lite nytt, utvecklas.”
>
> Respondent B

Vi kunde även genom intervju med respondent C utröna att personalen fann det väldigt viktigt att byta miljö då de ville vara säkra på att de hängde med i utvecklingen och inte tappade attraktion på arbetsmarknaden om de i framtiden ville byta jobb. Han förklarade det på följande sätt.

> ”Även som utvecklare kände vi att vi inte ville sitta kvar i det gamla för då har vi inga möjligheter att få jobb om 10 år”
>
> Respondent C

Respondent C förklarade att hade inte en övergång till .NET blivit genomförd i det företaget han arbetade på för tillfället hade troligtvis en hel del av personalen avgått.

5.3.2 Konkurrens

En aspekt som vi har funnit i intervjuer med respondent C är den att till .NET släpps hela tiden uppdateringar och nya komponenter som kunden kan ta del av, vilket vi även fann att det var av vikt för respondent A som hade förhoppningar om nya funktioner i .NET som de i sin tur skulle leda till nya funktioner till kunden. För honom låg den största vikten av en migrering vid vilka nya funktioner han kunde erbjuda kunden.

Respondent C menar på att det är bra att kunna visa för kund att man ligger långt fram i utvecklingen och att man använder sig av de senaste verktygen. Han anser att detta är en viktig del att ta med i betänkandet om en övergång eller ej.

Då vi intervjuade respondent B visades det att det kan finnas ett annat syfte med Visual Basic och dess kringmiljö än just de tekniska bitarna.

> ”Om man kollar på andra företag och deras produkter så stoltserar de ofta med att de har programmerat i .NET miljö. Även om det inte har någon betydelse för användaren direkt.”
>
> Respondent B

Respondent B klargör här att .NET miljön fungerar som en form av marknadsföringsfaktor vilket de försöker nå ut till potentiella kunder med, men han menar också på att det används enbart av denna faktor.

5.3.3 Inlärning

Även i intervjuerna identifierade vi att inlärningen kan ha inverkan på valet av övergång eller ej. Vi kan genom alla intervjuer se att denna punkt återkommer. Vi börjar med att presentera vad respondent A på Organisation X tror. Vid frågan om huruvida hans personal behöver någon form av inlärning i det sättet att programmera och tänka vad gäller programmering svara han att det troligtvis behövs någon form av inlärning men att hans utvecklare lär sig detta på ett par dagar.

Liknande tankar presenteras också i intervjun med respondent B på Organisation X. Han beskriver det som att det troligtvis enbart behövs en snabb uppdatering av den nya syntaxen. I intervjun med respondent C berättade han hur de gick tillväga när det handlade om inlärningen av det nya språket. Deras alternativ var att skicka ett antal utvecklare på kurs där de skulle lära sig det nya språket, deras kunskap skulle sedan användas vid internutbildning. Den här lösningen fungerade bra, där kunskap införskaffades innan själva migreringen satte igång. Intervjuerna behandlade även inlärningen av själva migreringsverktyget var vida en av utvecklarna på Organisation X beskrev det som att man antingen kunde hyra in en konsult som visade på hur verktyget skulle användas alternativt att man tog en snabb kurs.

6 Faktorer

Vi har nu ett litet grepp om vad Visual Basic, Visual Studio och .NET är för något samt skaffat oss en viss inblick i vilka egenskaper de besitter samt övriga aspekter som kan ha inverkan på valet. Denna fakta har vi funnit i både intervjuer och i litteratur. Men vi har inte skapat någon struktur i hur de påverkar valet samt vad de har gemensamt. I det här kapitlet ska vi gå lite djupare och undersöka vilka faktorer det är som påverkar valet samt på vilket sätt de påverkar. Faktorerna är dels identifierade ur egenskaperna hos Visual Basic 2005 och dess kringliggande miljö men även ur själva produkten samt på ett mer organisatoriskt plan. Dessa olika typer av faktorer presenteras i separata underkapitel.

6.1 Faktorer funna hos egenskaper

När vi utförde vår granskning av egenskaper hos Visual Basic 2005 och kringliggande miljö i både kapitel 4 och 5 fann vi en rad egenskaper som hade inverkan på valet, både i litteratur och i våra intervjuer. Dessa faktorer var VPL, RAD, objektorientering och språkets syntax. Men vad är det i de här egenskaperna som är de egentliga faktorerna? Att språket är visuellt, objektorienterat m.m. är egentligen inte någon faktor, det är vad visuell programmering och objektorientering ger som är de egentliga faktorerna och som påverkar valet men de har som visats under kapitel 4 och 5 visat på att det innebär konsekvenser, såväl positiva som negativa. Man kan exempelvis i våra intervjuer med Organisation X utläsa att utvecklarna ser objektorientering som en fördel medan personer på en mer organisatorisk nivå inte ser den egentliga fördelen och därför inte anser det vara av samma importans. Det är således viktigt att identifiera dessa faktorer för att förstå hur utvecklingen kommer att påverkas av egenskaperna hos Visual Basic 2005 vid övergången. Helt enkelt, vad det är i respektive egenskap hos Visual Basic 2005 och dess kringliggande miljö som gör att egenskapen har inverkan på valet.

6.1.1 Utvecklingstid

En av egenskaperna vi fann i både kapitel 4 och 5 var VPL. VPL hade flera positiva aspekter. En av de aspekterna var hastigheten i vilket man kunde utveckla applikationer och produkter. En av de aspekter med VPL som gör det till en faktor är att det kan öka utvecklingshastigheten. Nära knutet till VPL låg även RAD. RAD verktygs syfte är också att utveckla produkter snabbt och utvecklingstid är en central del i varför RAD är en faktor. Även objektorientering kan kopplas till utvecklingstiden. Återanvändning som är en stor sida av objektorienteringen finns till viss del för att slippa skriva om stora massor av kod vilket ökar hastigheten i utvecklandet. Man kan även integrera redan färdiga klasser vilket ytterligare snabbar upp utvecklandet. I kapitel 5 visar vi på att respondent B i sin intervju anser utvecklingstiden som väldigt viktigt och menar på att de har väldigt höga krav på utvecklingstiden som måste vara kort. Likaså gör respondent C i sin intervju och poängterar att den nya utvecklingsmiljön möjliggjorde snabbare utveckling vilket

var en positiv aspekt som har inverkan på valet av övergång eller ej.

Många av de egenskaper som vi har märkt spelar in på valet av övergång eller ej i kapitel 4 och 5 har således med utvecklingstid att göra. Vi kan därför konstatera att utvecklingstid är något man gärna tittar på innan man gör en övergång. Kortare utvecklingstid är något positivt hos en programmeringsmiljö. Snabb utveckling möjliggör att man snabbt kan skapa applikationer och komponenter på basis av användarnas efterfrågan, det kan också minska kostnaderna för utvecklingen. Ett flertal av Visual Basic och dess kringliggande miljös egenskaper har också för avsikt att korta ner utvecklingstiden, vilket kan ses som ett ytterligare tecken på att utvecklingstid är en viktig faktor i dagens programvaruutveckling och således även vid valet av övergång eller ej.

6.1.2 Komplexitet

I kapitel 4 fann vi även att VPL påverkar komplexiteten i utvecklingen. Vi skrev att VPL till stor del var till för att simplifiera utvecklingen. Man slapp förstå exakt hur knappar skapades och man kunde lätt se hur objekten relaterade till varandra. Man använder samma kommandon för att utföra ett flertal saker, en knapp läggs till på samma sätt som en radioknapp vilket gör språket lättare. Hade detta gjorts med hjälp av text hade man behövt skriva helt skilda saker. Det var dock inte helt säkert att VPL gjorde det enklare, snarare var det så att man borde kombinera visuell programmering med textbaserad. Vi kan dock fastslå att VPL påverkar komplexiteten i utvecklingen och att det är en del av vad som gör VPL till en faktor att ta i beaktning. I både kapitel 4 och 5 fann vi även att syntaxen kan ha inverkan på komplexiteten. Syntaxens två egenskaper läsbarhet och skrivbarhet är mått på hur komplex syntaxen är. Syntaxens komplexitet belyses alltså som en aspekt och är således en del av varför syntaxen egentligen är en faktor. Vidare har även objektorientering inverkan på komplexiteten. Det kan både vara en positiv och negativ inverkan. Dels blev arkitekturen lättare att följa i vissa fall, men i större projekt kunde objektorientering innebära svårigheter då samtliga utvecklare måste ha en helhetsbild av relationerna vilket kan bli komplext. Även RAD fann vi ha inverkan. Dels kan de färdiga komponenterna minska komplexiteten, men studier som vi har presenterat har även visat på att RAD kan innebära högre komplexitet om man vill skapa egna objekt som inte finns redan integrerade i verktyget. Även Microsoft själva hävdar att minska komplexiteten är en av huvudtankarna bakom de senare versionerna av Visual Basic som är baserade på .NET (Hilson, 2001).

Egenskaperna hos Visual Basic 2005 och dess kringliggande miljö är således i vissa fall faktorer p.g.a. att de påverkar komplexiteten i utvecklingen. Komplexitet är alltså en faktor som man tar i beaktning vid valet av programmeringsspråk. Man vill att det ska vara lätt att använda sig av språket och miljön eftersom det leder till att fokus kan läggas på själva problemlösningen istället för hur man ska lösa det i själva miljön (Ghezzi & Jazayeri, 1987). Låg komplexitet leder även till lägre utvecklingstid (Harkness et. al., 2007) vilket vi har visat ovan vara en faktor vid valet av övergång eller ej. Att Microsoft marknadsför Visual Basic och Visual Studio som ett lätt verktyg är också tecken på att det är något som man har i baktanke när man väljer sitt programmeringsspråk och har således inverkan på valet av övergång eller ej.

6.1.3 Underhållbarhet

Objektorienteringen har även haft inverkan på underhållbarheten och vidareutvecklingen. Arvsprincipen gav möjligheter att integrera objekt lätt i sin befintliga struktur. Strukturen är även lätt följa enligt många författare och även intervjuobjekt vilket påverkade underhållbarheten positivt. Det visade sig även vara lätt att utföra en ändring som innebar förändring i flera delar av systemet då man endast behövde ändra på ett fåtal ställen. Objektorientering kan dock även ha inverkan på ett negativt sätt när man pratar om system i större skala. Uppgifter fanns att det då var svårt att vidareutveckla systemet utan att behöva ändra mycket i den redan befintliga koden. En bra struktur och användning av mönster kunde dock råda bot på problemet. RAD hade den negativa faktorn att det tappade kontrollen lätt över underliggande struktur. Den underliggande strukturen var viktig för framtida uppdateringar och således kan underhållbarheten påverkas negativt vid användandet av RAD verktyg. Men den kunde även påverkas positivt p.g.a. RADs inverkan på utvecklingshastigheten. Snabb utveckling leder till nära kontakt med användaren och snabb framtagning av prototyper vilket gör att systemet kräver mindre underhållbarhetskostnader. Syntaxens kännetecken såsom läsbarhet hade även inverkan på underhållbarheten. Om det exempelvis var hög läsbarhet kunde man lätt underhålla och bygga ut sin kod.

Underhållbarhet är alltså ytterligare en faktor som man fokuserar på vid valet av övergång eller ej som också återfinns hos Visual Basic 2005 och dess kringliggande miljös egenskaper. Man vill åstadkomma hög underhållbarhet och det finns egenskaper hos Visual Basic 2005 och dess kringliggande miljö som påverkar denna faktor såväl positivt som negativt.

6.1.4 Felfaktorn

Med felfaktor menar vi risken för att fel uppstår och hur programmeringsmiljön kan påverka denna aspekt. VPL kunde ha en viss inverkan på felfaktorn hos programutvecklingen. Vi presenterade studier som gjorts där man jämfört felfaktorn mellan textbaserade och visuella programmeringsspråk och det visade sig att de visuella kunde ha en lägre felfaktor. Möjligheten att testa på ett smidigt och snabbt sätt minskade även risken för att den utvecklade produkten skulle innehålla fel. Objektorienteringens möjlighet att återanvända kod kan också minska felfaktorn då det blir färre ställen där misstag kan begås. Även möjligheten att integrera färdiga väl testade komponenter kan minska felfaktorn. Syntaxens egenskaper skrivbarhet och läsbarhet påverkar också felfaktorn. Om det är lätt att skriva koden är det mindre risk för att man gör fel, samma sak gällde läsbarheten. Hög läsbarhet gjorde det lättare att kontrollera sin kod vilket minskade felfaktorn på slutgiltiga produkten.

Låg felfaktor är alltså något man vill åt när man väljer ett programmeringsspråk och är alltså av importans vid valet av övergång eller ej. Visual Basic 2005 och dess kringliggande miljö hade inverkan på felfaktorn både positivt och negativt och det bör därför ses som en faktor vid valet av övergång eller ej.

6.2 Faktorer funna i Visual Basic som produkt

Vi har funnit efter att ha studerat litteratur och intervjuer att det finns faktorer som härstammar i själva produkten. Dessa faktorer påverkas således av Visual Basic 2005 och dess kringliggande miljö men kanske inte är någon egenskap såsom exempelvis objektorientering.

6.2.1 Kompatibilitet

Vi har funnit att kompatibilitet är något som man eftersöker i våra intervjuer under kapitel 5. Vi har också i litteratur visat att det finns vissa kompatibilitets aspekter med Visual Basic 2005 som kan vara värda att se över under kapitel 4. Kompatibilitet är således något som man anser vara viktigt och fokuserar på när man väljer sin programmeringsmiljö och har inverkan på valet och därför en faktor. Det är .NET som har inverkan på kompatibiliteten hos Visual Basic. Som sagt så är det här samtliga av de klassbiblioteken som sköter t.ex. kommunikationen med databaser återfinns. Det är även .NET som översätter den skrivna koden till CLR kod, vilket har inverkan på kompatibiliteten med andra språk såsom C#. .NET kanske inte kan ses som en egenskap hos Visual Basic 2005 men det ingår trots allt i valet när man väljer Visual Basic 2005. Kompatibiliteten är alltså något som påverkas av Visual Basic 2005 och dess kringliggande miljö och man bör kontrollera vilken nivå av kompatibilitet man eftersöker och om Visual Basic 2005 med .NET kan uppfylla dessa krav.

6.2.2 Tekniska möjligheterna

Vi har i intervjuer sett en efterfrågan efter nya möjligheter såväl hos slutprodukt som i själva utvecklingen hos Visual Basic 2005. Men vi har inte funnit det i vår litteraturgranskning. Nya tekniska möjligheter är helt kontextberoende, d.v.s. man kan inte avgöra om Visual Basic 2005 medför nya tekniska möjligheter innan man vet vad man migrerar ifrån. Det tror vi också är anledningen till att vi inte har hittat någon information angående denna faktor i litteraturen. Men finns det några nya möjligheter så har det för somliga en positiv inverkan på valet och för vissa är det t.o.m. ett krav, därför är det en viktig faktor som påverkar valet. Respondent C hade funnit olika möjligheter som han ansåg positiva vid övergången. Dessa möjligheter låg hos Visual Basic 2005 och dess kringliggande miljö. Även respondent A har en önskan om att Visual Basic 2005 och dess kringliggande miljö ska medför nya tekniska möjligheter. Denna faktor har således ungefär samma förhållande till Visual Basic 2005 som ovanstående punkten kompatibilitet. Den påverkas inte utav Visual Basics egenskaper men kanske direkt av dess kringliggande miljö och språket som sådant. .NET erbjuder möjligheter att använda sig av flera programmeringsspråk via CLR och har ett nytt klassbibliotek som kan erbjuda nya möjligheter.

6.2.3 Migreringsverktyg

En faktor som vi har funnit främst hos intervjuobjekt är möjligheten att använda ett migreringsverktyg för att korta ner själva tiden för omskrivningen av koden. Denna faktor är något som påverkar organisationen och kanske inte egentligen härstammar ur någon teknisk

egenskap, därför har vi placerat den under denna rubrik.

Vi kan klart se att där finns en önskan i migreringsprojekt att förkorta migreringstiden i antal arbetade timmar. Detta är troligtvis en ren konsekvens av att man vill hålla kostnaderna nere så mycket som möjligt. Detta faktum konstateras även i de intervjuer vi genomfört. En intressant aspekt att ta upp är att vi i kap 4 funnit att det vanligaste sättet att gå från ett programmeringsspråk till ett annat är att inte använda sig av ett migreringsverktyg. Detta kan dels bero på att där inte finns något verktyg mellan de språk man vill konvertera eller att verktygen är för komplexa och ger inte det resultat man eftersöker. Respondent C berättar att de ej använde sig av ett migreringsverktyg när de utförde sin konvertering från Visual Basic 6.0 till Visual Basic 2005, de fann att verktyget inte kunde utföra de mest elementära uppgifterna. Faktum kvarstår att migreringsverktyg är något som är efterfrågat hos våra intervjuobjekt. Om det finns eller inte påverkar valet om övergång eller ej och bör därför ses som en faktor som spelar in i valet. Man kan vid första anblicken se denna faktor som aningen irrelevant och ifrågasätta den som en faktor, men faktum är att den har haft stor inverkan på framförallt respondent A som sitter på en beslutsfattande position och ser till själva kostnaderna i migreringen. Man bör dock se den faktiska skillnaden i att använda verktyget och inte låta sig bli lurad även om det finns tillgängligt, då det inte alltid är säkert att det betyder en förkortad migreringstid.

6.3 Organisatoriska faktorer

Vi har nu presenterat faktorer som antingen härstammar utifrån egenskaperna i produkten eller från produkten som sådan. Men vi har även en sista typ av faktorer som är mer organisatoriskt anknutna. De påverkas till stor del av Visual Basic 2005 men de härstammar inte därifrån. Dessa är personal, konkurrens och inlärning. Vi anser alltså att dessa skiljer sig från de övriga rubrikerna då de påverkar och påverkas av organisationen.

6.3.1 Personal

Vi har i litteraturen, samt genom intervjuerna tagit del av en faktor som kan komma att spela en stor roll vid övervägandet av att använda sig av Visual Basic och Visual Studio 2005 som utvecklingsmiljö och språk. Den faktorn vi väljer att kalla personal gick ej att placera som en ren teknisk faktor utan kan möjligtvis ses mer som en generell faktor som ligger på det organisatoriska planet. Denna del av modellen syftar till inverkan Visual Basic och Visual Studio 2005 har på humankapital inom en organisation. Faktorn har framkommit efter utförd analys av kap5.

Vi tror att utvecklarnas professionalism ökar då de sitter i en ny och uppdaterad miljö där de kan använda sig av den senaste tekniken, detta är vidare konstaterat genom respondent B på Organisation X som menar på att han vill utvecklas och detta är en av orsakerna till varför han vill gå över till Visual Basic och dess kringmiljö. Vi har i tidigare kapitel presenterat fakta som säger att Visual Basic 2005 och dess kringmiljö ligger långt fram vad gäller de tekniska

aspekterna, vilket skulle möjliggöra personlig utveckling för de utvecklare som migreringen påverkar. Vi har även presenterat fakta som visar att en ny och uppdaterad miljö skulle kunna fungera som ett dragplåster för rekrytering av ny personal vilket bör vara av relevans för de flesta organisationer. Ytterligare en aspekt som rör personalen är att de anställda hela tiden som ovan nämnt vill utvecklas. Ger då inte organisationen möjlighet till det finns en risk att personal söker sig till en annan organisation som kan tillfredställa deras behov som utvecklare. Det här visar sig även i intervjun med respondent C, som tror att om inte en migrering blivit genomförd i den organisation han jobbade för hade en del personer valt att söka sig till nya organisationer.

Genom att ha lokaliserat fakta i både intervjuer och i litteraturen känner vi att detta är en faktor som man bör ta med i övervägandet vid en migrering, då den kan leda till förbättringar för organisationen i fråga.

6.3.2 Konkurrens

Faktorn konkurrens har även den kunnat utrönas i kap 5 genom intervjuer vi genomfört med våra olika företag, och liksom faktor personal är det inte en teknisk faktor utan ligger på ett organisatoriskt plan. För att kunna förstärka vad vi funnit i intervjuerna och försöka placera konkurrens som en faktor, har vi med hjälp av litteratur som presenteras i kap 4 försökt ge denna faktor legitimitet och trovärdighet. När vi pratar om konkurrens under detta kapitel är det inte de konkurrensmöjligheter Visual Basic 2005 samt dess kringliggande miljö kan ge oss på en teknisk basis utan här handlar konkurrensen om de fördelar varumärket för med sig.

I våra intervjuer har vi funnit att personer i de olika organisationerna tycker det är av relevans att kunna erbjuda sina kunder nya funktioner. Vi har i tidigare kapitel fört resonemanget om att Visual Basic 2005 samt dess kringmiljö är tekniskt avancerad och kan troligtvis erbjuda kunderna nya tekniska möjligheter. Vi tros efter detta tidigare resonemang därför kunna dra slutsatsen att en användning av Visual Basic 2005 samt dess kringliggande miljö anses vara konkurrenskraftig på marknaden. Ytterligare fakta som presenteras i intervju med respondent C är att det är bra att inför kund kunna visa att man ligger långt fram i utvecklingen och att man använder sig av de senaste verktygen. I Blom et. als. (2004) studier presenteras fakta som säger att utveckling som sker i Visual Studio på en .NET plattform anses på marknaden vara hållbara och pålitliga då Microsoft levererade helhetslösningar som är väl kompatibla med varandra. Något som vi inte klart funnit i litteratur men delvis i intervjuer är aspekten av att åka snålskjuts på ett stort företags namn. Väljer man att arbeta med produkter som kommer från Microsoft tar man troligtvis del av det stora företagets fördelar genom sitt välkända varumärke på marknaden. Detta kan kanske vara av intresse för mindre företag som får gratis marknadsföring genom att inför kund säga att man arbetar med .NET. Det som återfinns i intervjuer som har koppling med detta är när respondent B beskriver det som att vissa organisationer stoltserar med att de programmerat i en .NET miljö, även fast det inte har någon direkt betydelse för slutkunden.

Efter att vägt in fakta som står skrivet ovan tycker vi att denna faktor är grundad nog för att ta med som en väsentlig del av modellen som vi presenterar. Om man väljer att bortse från denna faktor missar man troligtvis något som kan betydelse för organisationen.

6.3.3 Inlärning

Vi anser efter att ha tolkat våra intervjuer samt granskat tidigare studier att inlärning är en faktor att ta in i beräkningen när man väljer att göra en migrering. Vi tar här upp och behandlar denna faktor och försöker påvisa sambandet mellan våra intervjuer i kap 5 samt det som står skrivet i litteraturen under kap 4. Nedan för vi en diskussion om de fakta vi använder oss av för att stärka våra resonemang och varför vi väljer att väga in denna faktor i valet av övergång till Visual Basic och Visual Studio 2005 eller inte.

I litteraturen samt i empirin ovan presenterat att inlärningen har inverkan på valet av övergång eller ej. Men även Visual Basic kan ha inverkan på inlärningen vilket i sig kan ses som en faktor att ta i beaktning inför en övergång. I kap 4 presenterar vi fakta som menar på att om man kan minska inlärningstiden kan man även minska kostnaderna för migreringen, vilket är positivt. Vi har visat att Visual Basic och dess egenskaper har inverkan på komplexiteten vilket i sin tur kan påverka inlärningstiden, om språket är lätt att använda så kan inlärningstiden minska.

Vi nämnde tidigare att utvecklare har konceptuella modeller för hur denne ska utveckla produkter, så om Visual Basic 2005 till stor del liknar föregående språk i sin struktur så minskar inlärningstiden detta resonemang bygger vi på efter dess genomgång i kap 4. Objektorientering är ett sådant exempel. Man kan se objektorienteringen som ett sätt att strukturera sin utveckling med objekt och liknande om tidigare språk också är objektorienterat så kan således utvecklaren bibehålla stor del av sina konceptuella modeller och strukturer när han övergår till Visual Basic 2005.

Kunskap om det nya utvecklingsverktyget samt språket anser vi vara viktigt att organisationen har införskaffat sig innan de utför migreringen, tidigare studier har visat att med assistans från kollegor går det snabbare att lära sig kontra att själv försöka sitta och leta i litteratur. Vidare fann vi i vår litteratur svårigheten att släppa tidigare tankesätt man arbetat utefter, vilket i det nya språket kunde leda till effektivitetsförluster då man inte utnyttjade dess funktionalitet till fullo. Även detta faktum kan man undvika på ett lättare sätt om organisationen som ska genomföra migreringen besitter kunskap om det nya språket och dess funktioner. Då inlärningen är förknippad till en direkt kostnad vid migreringen bör man väga in denna faktor då man gör valet, huruvida personalen besitter den kunskap som behövs eller om man behöver hyra in konsulter, eller skicka befintlig personal på utbildning. Visual Basic 2005 hade också egenskaper som kunde påverka inlärningstiden och således också kostnaderna för migreringen. Vi anser att denna faktor bör tas i beaktning och väljer därför att väga in den i vår modell

7 Utformning av modellen

Vi har nu funnit en hel del faktorer som har inverkan på valet av övergång eller ej. Faktorerna härstammar från både egenskaper hos produkten såväl som från produkten själv men även från organisatoriska påtryckningar. Det som däremot är gemensamt för samtliga av de faktorer vi presenterat under kapitel sex är att de alla påverkas av Visual Basic 2005, antingen från dess egenskaper eller från själva produkten. Således vill vi placera Visual Basic 2005 och dess kringliggande miljö i form av Visual Studio och .NET längst till vänster i modellen (fig. 7.1). Där återfinns även de egenskaper som vi har funnit haft inverkan på valet, och som har legat till grund för och påverkat vissa av faktorerna. Näst till höger återfinns således de faktorer som vi har identifierat. Där samtliga påverkas av Visual Basic 2005, dess kringmiljö samt egenskaper. Fig. 7.1 är modellen är modellen så här långt.

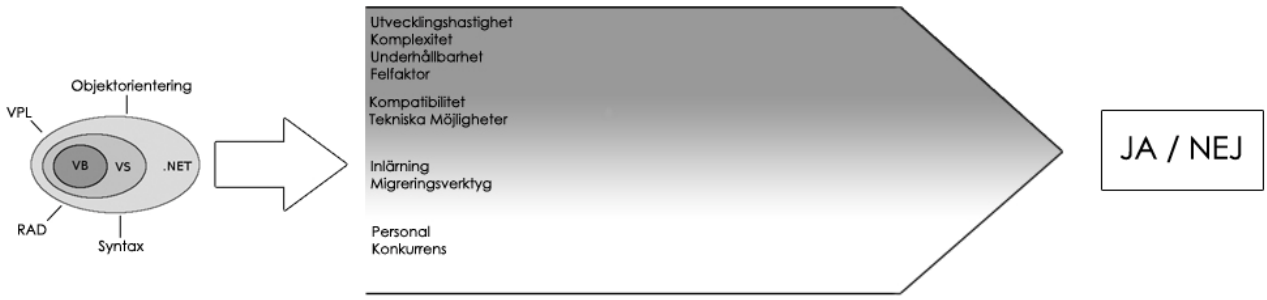

Fig. 7.1 Modell steg ett (egen)

För att utforma en modell vidare, och se vad som ännu mer övergripande påverkar valet så vill vi särskilja de olika faktorerna och se om det finns något sätt att kategorisera dem och se hur de påverkar valet, och hur de hänger samman. Nedan kommer vi presentera dessa ytterligare kategoriseringar steg för steg och under varje steg presentera vilka faktorer som hör hemma där samt varför.

7.1 Effektivitet

Många av de faktorer som härstammar från egenskaper hos Visual Basic 2005 påverkar effektiviteten. Komplexitet, utvecklingshastighet, felfaktor och underhållbarhet menar vi är faktorer som påverkar effektiviteten i utvecklandet. Det är faktorer som man eftersöker för att kunna skapa felfriare och fungerade produkter snabbt. Faktorn utvecklingshastighet handlade om hur Visual Basic 2005 och kringliggande miljö påverkade i vilken hastighet man kunde utveckla vilket enligt oss är ett effektivitets perspektiv. Komplexitet påverkade utvecklarnas kreativitet men även utvecklingshastighet. Vi anser således att låg komplexitet ökar effektiviteten i utvecklingen. Att lätt kunna underhålla sin produkt är också ett tecken på effektivitet. Visual Basics egenskaper fann vi hade inverkan på underhållbarheten, att snabbt och lätt kunna underhålla sin produkt anser vi vara ett tecken på effektivitet. Även att minska risken för att fel uppstår påverkar effektiviteten i utvecklingen. Egenskaper såsom VPL, RAD, objektorientering och syntaxen påverkar alltså i långa loppet effektiviteten i utvecklandet via utvecklingstid,

komplexitet, felfaktor och underhållbarhet. Huruvida produkten medför effektiv utveckling har således stark inverkan på valet. Den här kategorin grundar vi således i att de egenskaper hos Visual Basic 2005 och kringliggande miljö som man efterfrågar påverkar effektiviteten. Ser man på våra intervjuer finner man att det här är en faktor som främst utvecklare ser till. Även om personer på beslutsfattande position vill ha en effektiv utvecklingsmiljö så har de svårare att se hur egenskaperna hos Visual Basic 2005 påverkar effektiviteten. Effektivitet är däremot något som påverkar valet och sammanfattar faktorerna, komplexitet, utvecklingshastighet, felfaktor och underhållbarhets inverkan på valet av en övergång eller ej. Får man lägre komplexitet, högre underhållbarhet, högre utvecklingshastighet och lägre felfaktor p.g.a. olika egenskaper hos Visual Basic 2005 så påverkas effektiviteten positivt i utvecklandet. Vi tycker också att det är viktigt att belysa denna del av faktorerna för folk på organisatorisk nivå då det dels har stark inverkan på utvecklarna men även för att det ökar effektiviteten och således kan minska kostnaderna för utvecklingen. I fig. 7.2 har vi implementerat effektivitet i modellen framför utvecklingshastighet, komplexitet, underhållbarhet och felfaktor.

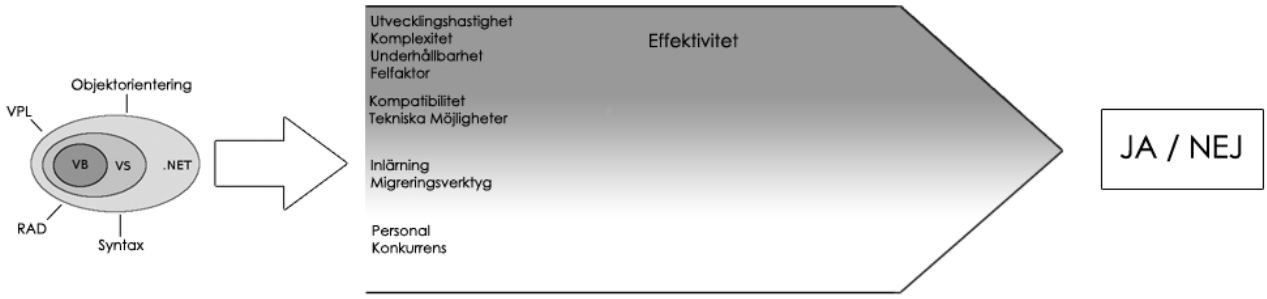

Fig 7.2 Modell steg två (egen)

7.2 Möjligheter

Utöver effektivitet fann vi även att möjligheter var något som var viktigt för valet av övergång eller ej. Vi identifierade en faktor som var tekniska möjligheter där intervjuobjekten sökte nya tekniska möjligheter med en övergång, både mot kund och för utvecklare. Denna punkt ligger nära effektivitet men skiljer sig i den mån att den inte påverkar effektiviteten direkt. Nya tekniska möjligheter behöver inte påverka effektiviteten i utvecklandet det innebär bara att man kan utveckla nya funktioner och tjänster till kund. På samma sätt anser vi även att kompatibilitet påverkar möjligheterna. Nya möjligheter att ansluta exempelvis nya DBMS leder kanske till att man kan utveckla sin produkt och leverera nya tjänster. Nya komponenter kan integreras vilket kan medföra möjligheter att utveckla tjänster som inte tidigare var möjligt. CLR möjliggjorde integrering av exempelvis C# kod i sin Visual Basic 2005 kod vilket kanske erbjuder möjligheter att utveckla saker som inte var möjligt med enbart Visual Basic 2005. Möjligheter är något som man både på beslutsfattande och utvecklarnivå eftersträvar. Men möjligheterna man söker kan ibland skilja sig något åt då beslutsfattare är mer intresserade av vilka möjligheter det innebär för slutkund och utvecklare mer fokuserar på vilka möjligheter som de får i utvecklandet. Vi sammanfattar således tekniska möjligheter med kompatibilitet under punkten möjligheter som vi anser ha inverkan på valet av övergång eller ej och presenterar modellen så här långt i fig. 7.3.

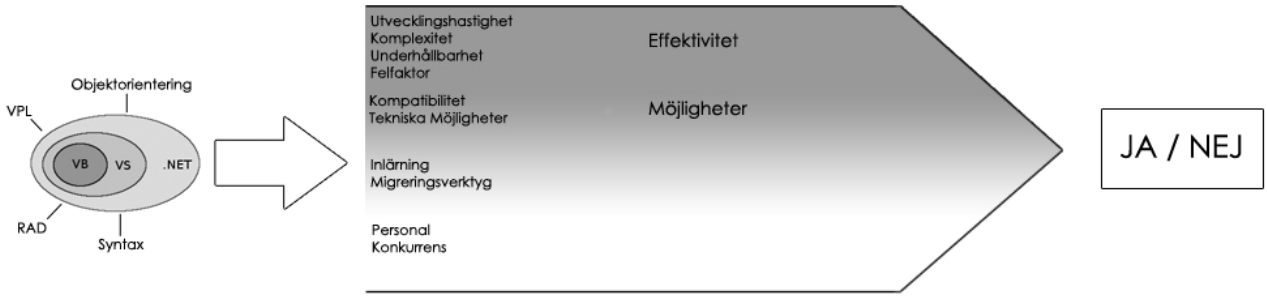

Fig. 7.3 Modell steg tre (egen)

7.3 Migreringskostnader

Då vi i tidigare kapitel 6 skrivit om inlärning och migreringsverktyg och hur de är faktorer som har inverkan vid övervägandet om att övergå till Visual Basic 2005 och dess kringmiljö har vi även lyckats utröna en gemensam nämnare för dem båda, denna nämnare är kostnad. Vårt problem, migreringen, grundar sig i att en migrering kostar pengar. Faktorn inlärning har med kostnad att göra då den ser till hur personalen måste införskaffa sig vidare kunskap inom det område man vill migrera till, i detta fall Visual Basic 2005. Att införskaffa sig denna kunskap tar oftast tid vilket i sig innebär löner som ska betalas ut samt förlorad produktionseffektivitet under inlärningen. Om det krävs kurser för att lära sig miljön som skall brukas, innebär även dessa en kostnad för organisationen. Inlärningen påverkar således kostnaden för migreringen negativt. Men eftersom vi har visat att Visual Basic 2005 kan påverka inlärningen så påverkas således kostnaden för övergången vilket har inverkan på valet av att göra en övergång eller ej. Inlärningen är således en faktor som är negativ men den kan också påverkas av Visual Basic 2005 och är således en faktor som har med migreringskostnader att göra.

När vi vidare ser till migreringen och hur den är kopplad till kostnad lägger vi i kapitel 5.2.3 om migreringsverktyg fram ett citat från respondent B som beskriver migreringen som en direkt kostnad, då man han menar på att det är ingen som direkt betalar för att man ska skriva om koden. Det är något utvecklingsföretaget måste bekosta själva, man ser också till den tid det kommer ta att göra en migrering. Det är här övervägandet och önskan om ett fungerande migreringsverktyg vägs in, då man hoppas på att migreringsverktyget ska minska tiden för migrering vilket minskar kostnaden för migrering. Migreringsverktygen påverkar således kostnaden för övergången i likhet med inlärningen.

Eftersom själva problemet med en övergång är kostnaden, så tror vi att en kategorisering av faktorerna som kan påverka kostnaden för migreringen är viktiga att särskilja. Faktorerna inlärning och migreringsverktyg sammanfattas således under migrationskostnader och modellen ser ut enligt fig 7.4.

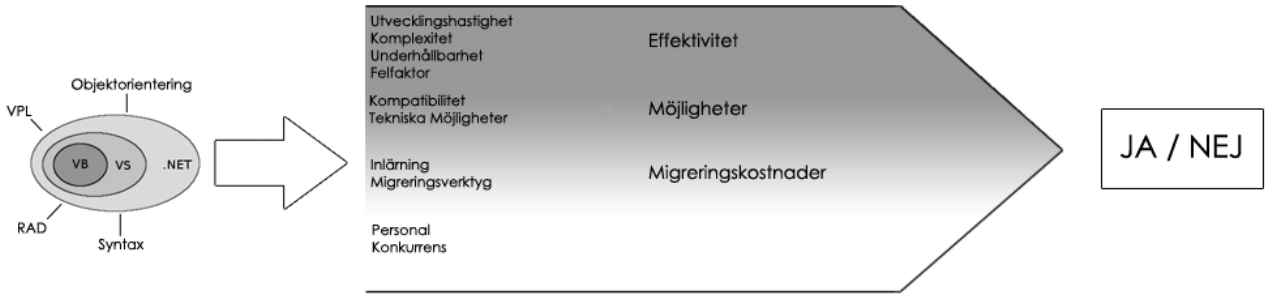

Fig. 7.4 Modell steg fyra (egen)

7.4 Affärspolitiska aspekter

Faktorerna konkurrens och personalaspekter är de faktorer vi hitintills inte har berört. Vi har mellan dessa två faktorer funnit ett samband som vi tycker särskiljer dem från de andra faktorerna. Vi anser dem påverka organisationens affärspolitiska aspekter. Med affärspolitiska aspekter menar vi organisationen möjlighet att påverka sin marknadsposition. Vi har funnit att Visual Basic 2005 med dess kringliggande miljö har inverkan på ett företags möjlighet att konkurrera. När vi under konkurrenskapitlet pratade om konkurrens menade vi den informella konkurrensen, d.v.s. den typ av konkurrens som själva produkten gav i sig. Organisationen fick således stärkt marknadsposition endast genom att använda sig av Visual Basic 2005 även om de inte skulle använda sig av några nya funktionaliteter hos produkten. Även personalen påverkades av ett byte utöver det att deras utvecklingsverktyg förändrades. De stärkte sin egna marknadsposition då de kunde hålla sig i framkant inom sitt yrke. De kände även att deras nuvarande jobb skulle bli mer intressant och de kunde tänka sig att stanna på samma arbetsplats under längre tid om de fick någon förändring i sin arbetsmiljö, något som visade sig vara viktigt för tekniska organisationer. På samma sätt kunde även arbetsgivaren konkurrera om den personal som finns ute på arbetsmarknaden bättre om de hade en modern och inspirerande arbetsmiljö att erbjuda.

Personalaspekter och den informella konkurrensen har således många likheter då de påverkar företagets möjligheter att stärka sin marknadsposition. Till viss del kan man tycka att även andra faktorer som nya tekniska möjligheter påverkar ett företags marknadsposition, och visst är det så men det kräver att man först implementerar dessa nya möjligheter. Faktorerna personalaspekter och konkurrens har en direkt påverkan på företagets marknadsposition. Vi lägger således till affärspolitiska aspekter i fig. 7.5 och utvecklar modellen ett steg till.

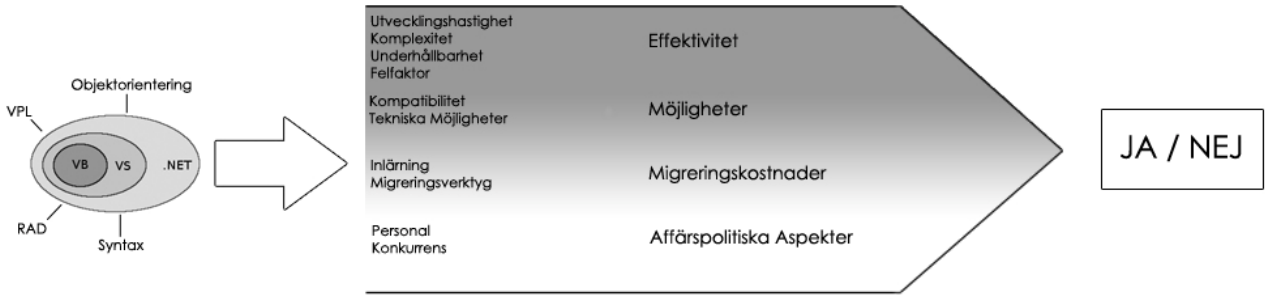

Fig 7.5 Modell steg fem (egen)

7.5 Teknik och Organisation

Vi har nu tagit fram ytterligare kategorisering av våra faktorer och påvisat samband mellan dem och hur de påverkar valet. När vi granskar våra intervjuer efter att ha gjort denna kategorisering så tycker vi också att det är just dessa kategorier som lyser igenom intervjuobjektens krav. Även om de pratar om ganska specifika och djupa egenskaper och faktorer så är det just de här fyra kategorierna som påverkar deras val i grund och botten. Även i litteratur är det de här kategorierna som återfinns. Det är högre effektivitet, fler möjligheter, möjliga sätt att påverka migrationskostnaden samt olika affärspolitiska aspekter som man efterfrågar när man ska göra en övergång till Visual Basic 2005 med kringliggande miljö. Men vi tycker att en viktig del fattas i modellen så här långt. När vi har gjort vår analys, både på litteratur och på intervjuer så har vi funnit att det egentligen finns två olika sätt som Visual Basic 2005 och dess kringliggande miljö påverkar valet, en teknisk del som innefattar de faktorer som påverkar effektiviteten och de nya möjligheter som en övergång innebär. Men det finns även faktorer som inte är direkt tekniskt anknutna. De här faktorerna återfinns under migrationskostnader och affärspolitiska aspekter. Det är fortfarande produkten Visual Basic 2005 som teknisk produkt som ligger i grund även för de här organisatoriska faktorerna, och de tekniska specifikationerna har också inverkan på exempelvis inlärningen vilket visats i tidigare kapitel. Men faktorerna inlärning, migreringsverktyg, personalaspekter och den konkurrensfördel som Visual Basic 2005 innebär är snarare organisatoriska aspekter än tekniska då de påverkar organisationen vid en övergång och inte den tekniska utvecklingen vilket de resterande faktorerna gör. Det är ingen tydlig gräns mellan de olika faktorerna och de nya möjligheterna och tekniska faktorerna påverkar också de organisatoriska i viss mån men även vice versa är tänkbart. Nya tekniska möjligheter kan t.ex. påverka personalen inom organisationen. Vi kommer därför inte dra en heldragen linje mellan de organisatoriska och de tekniska faktorerna och kategorierna i modellen. Vi anser däremot att det kan förtydliga modellen för läsaren och kanske även höja användbarheten i den då det blir lättare att urskilja hur de olika faktorerna påverkar valet samt deras inverkan vid övergången. Vi placerar av samma anledning inte dessa två kategorier av faktorer framför några andra utan snarare i bakgrunden av modellen då de snarare indikerar typen av faktor och inte egentligen vad som påverkar valet i respektive faktor som tidigare presenterade kategorier har gjort. I fig 7.6 finner ni modellen efter att vi har implementerat teknik och organisation kategorierna.

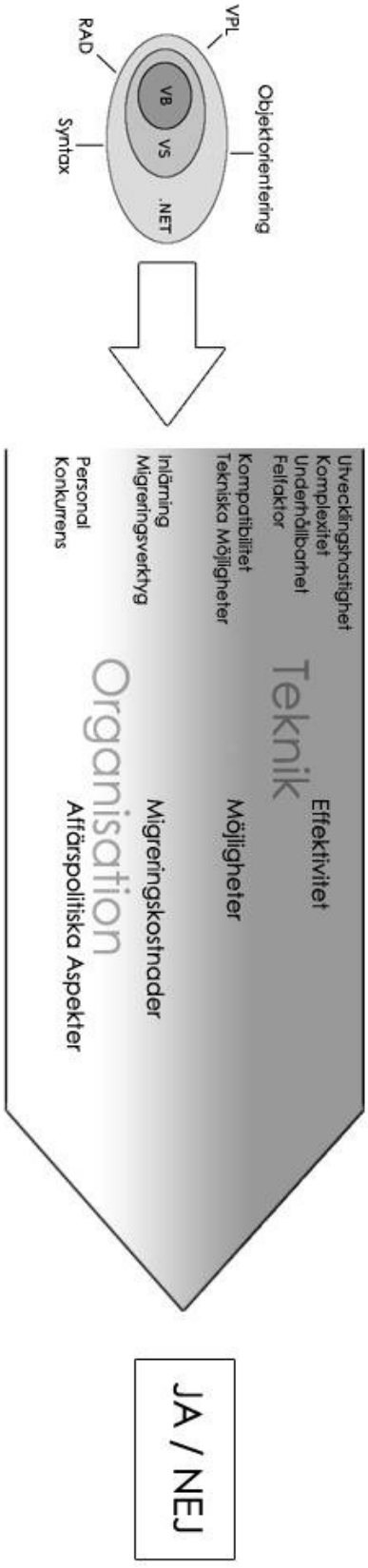

Fig. 7.6 Modell steg sex (slutgiltig) (egen)

8 Konklusion

8.1 Slutsats

Vår frågeställning var att ta fram vilka faktorer som påverkar valet av övergång samt hur de hängde samman. Vår slutsats är främst vår modell i fig. 7.6. Modellen innehåller de faktorer som påverkar valet samt hur vi har funnit att de hänger samman vilket också är vår slutsats då det besvarar vår frågeställning. Modellen blev utformad som så att längst till vänster placerar sig Visual Basic 2005 med dess kringliggande miljö i form av Visual Studio 2005 och .NET. Egenskaperna VPL, RAD, objektorientering och syntax fann vi inte vara direkta faktorer utan återfinns snarare i Visual Basic 2005 och dess kringliggande miljö. Visual Basic 2005 och dess kringliggande miljö påverkade sedan de faktorer som återfinns ett steg till höger i modellen, utvecklingshastighet, komplexitet, underhållbarhet, felfaktor, kompatibilitet, tekniska möjligheter, inlärning, migreringsverktyg, personal och konkurrens. Dessa i sin tur var faktorer som påverkade effektiviteten, möjligheterna, migrationskostnaderna och affärspolitiska aspekter. Sammantaget fann vi att de olika faktorerna kunde kategoriseras i tekniska och organisatoriska faktorer något så när och vi fann att de var på dessa två plan det fanns faktorer som påverkade valet av övergång eller ej.

Men vi ska också under det här kapitlet belysa vad vi mer djupgående har kommit fram till som kanske inte visas i modellen.

Det var som sagt Visual Basic som produkt, Visual Basics kringliggande miljö, t.ex. .NET och dess egenskaper som låg i grund för faktorerna. Dessa olika ”delar” av Visual Basic 2005 påverkade och skapade dock olika faktorer. Man kan till viss del härleda en faktors bakgrund och till viss del från dess inverkan på valet. De faktorer som påverkar effektiviteten härstammar exempelvis ur egenskaperna hos Visual Basic 2005.

Det här håller dock inte fullt ut då faktorerna under migreringskostnader härstammar ur både egenskaperna och själva produkten. Dock kan man se ett visst mönster i att faktorernas ursprung påverkar dess inverkan på valet då det stämmer i 3 fall av 4. Vi fann också att det egentligen finns två typer av faktorer, dels de tekniska faktorerna och dels de organisatoriska faktorerna som båda härstammar från Visual Basic 2005 men som påverkar valet på olika plan.

Faktorerna som har presenterats i modellen är funna i både litteratur och i intervjuer och till största del så finner man samma faktorer när man granskar dem. Det finns alltså en ganska klar bild om vad det är som påverkar valet. På djupare nivå finns det dock en viss skillnad, exempelvis RAD verktyg är det ingen av respondenterna som eftersöker specifikt trots att vi har funnit att det är en egenskap som kan ha konsekvenser, men snabb utveckling som RAD verktyg kan bidra med eftersöks. De som överväger en övergång har således ganska bra kunskap om vad det är som påverkar deras val och det stämmer också överens med litteraturen. Dock har de svårt att sätta fingret på vad exakt det är som frambringar dessa faktorer hos Visual Basic 2005. Det är också dessa samband som vår modell visar på.

8.2 Metodutvärdering

Vi anser att vårt kvalitativa tillvägagångssätt och vår blandade abduktion strategi har gett oss ett bra djup i vår informationsinsamling som legat i grund för analysen vi har utfört för att finna faktorerna. Kombinationen av intervjuer och litteratur har gett oss en bra inblick i vad som spelar in på valet av övergång eller ej. Vi hade dock önskat att fler intervjuer hade kunnat genomföras men p.g.a. av uteblivna intervjuobjekt och arbetsmarknadens brist på tid så fick vi tyvärr ingen möjlighet till det. Vi hade gärna sett att ytterligare intervjuer hade gjorts med folk som redan har migrerat då de ger oss mer empirisk. Fler intervjuer hade kunnat ge oss högre validitet och reabilitet i arbetet samt möjligtvis kunnat utvinna i ytterligare faktorer som vi inte funnit. Dock anser vi att de intervjuer vi genomförde hade hög relevans inom ämnet och gav oss mycket information, mycket tack vare dess ostrukturerade form. Frågeställningen medförde svårigheter i upplägget och strukturen av arbetet då vi fick det svårt att särskilja på analys och diskussion och samtidigt bibehålla den röda tråden genom arbetet. Vi tycker dock att det upplägg som vi har använt oss av har varit den mest lämpliga kombinationen och gett en bra röd tråd samtidigt som strukturen bibehållits.

Vi har även under arbetets gång tänkt på hur arbetet skulle ha sett ut om vi använt oss av ett kvantitativt tillvägagångssätt. Man hade t.ex. kunnat samla empirisk data från organisationer som gjort en övergång där de fick utvärdera .NET och försöka få svar på vad de ansåg vara bra respektive mindre bra såväl med slutresultatet som med själva övergången. När vi ser hur vi arbetade med den deduktiva strategin där vi först läste in oss på litteratur som vi sedan utformade våra intervjuer med hjälp utav, gav detta oss en bra grund att bygga vidare teorier på. Den abduktiva tillvägagångssättet som vi faktiskt arbetat utefter anser vi vara dem mest lämpade metoden för denna uppsats, då de andra enskilt inte kunde förse oss med ett tillräckligt tillvägagångssätt. Självklart är det inte omöjligt att användandet av enbart en metod åt gången hade givit oss andra resultat, man kan därför ifrågasätta huruvida vårt tillvägagångssätt är det mest lämpliga för denna form av studie.

Vi hade i början ett strikt induktivt tillvägagångssätt men insåg att detta inte var tillräckligt då vi var tvungna att grunda våra intervjuer på tidigare teorier i litteratur. Vi kunde heller inte arbeta strikt deduktivt då vi ville fortsätta utforma vår egen teori efter de intervjuer vi genomfört. Vi kom därför fram med det abduktiva tillvägagångssättet.

När vi ser till vårt litteraturval anser vi att vi har gjort en bra filtrering av information som väl stämmer överens med vårt ämne. För att få trovärdig information så ligger basen i vetenskapliga artiklar. Vissa artiklar kan anses vara föråldrade men de har använts i syften då vi anser att de fortfarande har relevans. Somlig information är hämtad från Microsoft och MSDN. Där har vi ställt oss väldigt kritiska till informationen och endast använt det som produktpresentation samt till liknande syften då den av oss uppfattas som väldigt partisk i bedömningen om man ska göra en övergång eller ej.

8.3 Vidare forskningsförslag

Under arbetets gång har vi funnit en del ämnen och inriktningar som vi funnit intressanta men som vi dels p.g.a. frågeställningen och dels tidsmässigt inte kunnat täcka. Vi presenterar dem dock under detta kapitel som förslag på framtida uppsatser.

- Ett intressant synsätt vore att se på hur faktorerna skiljer sig på olika nivåer inom organisationen. Vi har funnit tendenser till att personer på ledningsnivå inte fokuserar på samma faktorer som t.ex. utvecklarna. Vi har till viss del nämnt det under arbetet men en djupare forskning kan vara intressant då det kan hjälpa organisationer att förstå vad en övergång kan ge dem.
- En annan intressant studie skulle kunna vara att göra en undersökning på hur pass väl förbereda organisationer är innan de gör en övergång samt om de är införstådda i vad en övergång innebär. Motivet kan vara att försöka motverka många av de problem som uppstår för organisationer vid en övergång.
- Det är även tänkbart att utföra en liknande studie på en annan plattform och se om man finner liknande resultat, d.v.s. är de faktorer vi funnit någorlunda generella för en övergång.

Presentation av respondenter samt deras organisationer

Respondenter:

Respondent A
Vår respondent sitter på en beslutsfattande position inom organisation X. Respondenten har arbetat inom företaget sedan dess start 1989 och har varit med och genomfört alla de uppgraderingar som tidigare skett inom organisationen mellan olika versioner av Visual Basic.

Respondent B
Sitter som utvecklare på organisation X, har bred kunskap inom programmering och har varit med och drivit utvecklingen av organisation X produkt sedan 2000. Vi väljer att intervju respondenten då dennes information kring en migrering troligtvis skiljer sig mot respondenten på en beslutsfattande position. Vi kan då analysera från två olika plan på samma organisation. Respondenten har även en del kundkontakt i samband med utvecklingen av produkten.

Respondent C (Martin Sällberg)
Martin Sällberg har även han jobbat på Organisation X innan de hade planer på att göra en migrering. Han har tidigare erfarenheter av att göra en migrering till .NET från Visual Basic 6.0. Detta utfördes under tiden han arbetade för Semcon. Vi antager att han besitter en hel del fakta och information till vad det var som spelade roll när hans organisation skulle göra en migrering.

Organisationer:

Organisation X

Är ett företag som bedriver ”in-house” development, vilket innebär att de utvecklar ett eget system som de sedan marknadsför ut mot potentiella kunder. Deras nuvarande utveckling sker i Visual Basic 6.0 och de står inför frågan om de ska göra en uppgradering till ett nyare språk eller inte.

Mandator/Semcon AB

Avdelningen som Martin jobbade på tillhörde först Mandator men blev uppköpt av Semcon AB 2004. Semcon utvecklar högteknologiska system inom ett flertal olika branscher. Avdelningen som var av intresse för oss tillverkade testsystem för Volvo. Denna utveckling har skett i allt från vb 5 till nuvarande C# .NET. Anledningen till att organisationen är intressant för oss är att de har utfört en migrering inom det område där vi utvecklar vår modell.

Ordlista

Abstraktion – Är ett begrepp som identifieras i objektorienterade språk där ett objekt i systemet är en abstraktion av ett verkligt objekt.

ACM (Portal) - **A**ssociation for **C**omputing **M**achinery, Stor portal med artiklar inom computing, informationstenknik. www.acm.org.

ADO.NET – **A**ctiveX **D**atabase **O**bject. Är ett sätt att upprätta kontakt mellan program och servers. Vanliga servrar som det ado.net stödjer är MS SQL, OLE DB, OBDC. Med dess funktioner kan man även manipulera data sparad i databasen.

API – **A**pplication **P**rogramming **I**nterface är en regelbok som tillhandahåller färdiga klasser, som man kan anropa i ett givet programmeringsspråk.

ASP.NET – Active **S**erver **P**ages. Används för att skapa dynamiska webbsidor och baseras på .NET. Skriptspråk som är vanligt vid utvecklandet av webbsidor.

CLR - **C**ommon **L**anguage **R**untime. CLR är där själva "motorn" hos .NET ligger. Minneshantering, hantering av datatyper m.m. Här översätts alla datatyper till ett standard språk som exempelvis möjliggör arv mellan olika språk.

CPU – **C**entral **P**rocess **U**nit. Processorn i dator, här sker beräkningarna i olika hastigheter beroende på prestanda av processorn. AMD, Intel är exempel på olika märken som tillverkar processorer.

DBMS - **D**ata**b**ase **M**anagement **S**ystems. System som behandlar data och lagar data. Vanliga former av DBMS är ORACLE, MS SQL, Microsoft Access.

Enkapsulation - Begrepp som är vanligt inom objektorienterade språk, innebär att man döljer ett objekts specifika funktioner för omvärlden. Objektet kan enbart påverkas av funktioner som tillhör just dess klass.

ELIN – Electronic **L**ibrary **I**nformation **N**avigator integrerar information från flera förlag, databaser och öppna e-print arkiv.

IDE - Integrated **D**evelopment **E**nvironment, Visual Studio är en är en IDE för t.ex. Visual Basic, där man kan utveckla själva programmen samt kompilera koden.

MVC - **M**odel **V**iew **C**ontroller, är ett sätt att bygga upp sitt system i olika nivåer. Ett interface mot användare eller annat system, en controllerklass samt en abstraktion av själva problemområdet som representeras av klasser, objekt, metoder och dess kopplingar.

.NET 2.0 – Är en senare version av .NET framwork, har bland annat stöd för bättre chaching vid programmutveckling.

Overload – Innebär att du kan skapa flera metoder med samma namn inom samma klass. Vilken metod som väljs, styrs av data man skickar till den.

Prototyping – Innebär att man tar fram prototyper av programmet väldigt tidigt i utvecklingen

som man sedan later användaren testa och utvärdera.

RAD - Rapid **A**pplication **D**evelopment (METOD). Innebär att man avänder sig av en metod för systemutveckling som medför en snabb utveckling av system.

RAD - Rapid **A**pplication **D**evelopment (Verktyg). Visual Studio är ett RAD verktyg som möjliggör snabb utveckling genom text att erbjuda utvecklaren färdiga komponenter, som kan dras in i koden.

Threads – Multitrådning, innebär att man kan skriva program som utför många processer samtidigt.

UML - Unified **M**odelling **L**anguage. Är ett sätt att modellera, man kan exempelvis modellera en verksamhets problemområde för det område man avser att utveckla ett system. Kan användas vid programmering i objektorienterade språk.

Vektorhantering – Array samt minneshantering i programmeringsspråket.

Visual Studio – Microsofts senaste utvecklingsverktyg, med stöd för ett flertal programmerinsspråk.

VPL - Visual **P**rogramming **L**anguage, Språk som använder sig av ett grafiskt interface istället för en textbaserad interaktion.

XML – e**X**tensible **M**arkup **L**anguage. Är en standard för vars syfte är att skicka data mellan system som ren text, som även ska kunna förstås av människor.

Programmeringsspråk

Integrerade i Visual Studio 2005

- C#
- J#
- Visual Basic 2005
- Visual C++

Övriga Programmeringsspråk

- Java
- UCSD Pascal
- OSU APL
- C
- C++
- Fortran
- Visual Basic 6.0 (äldre version av Visual Basic 2005)

Referenslista

Litteratur

Alvesson, M. & Sköldberg, K. (1994): *Tolkning och reflektion – Vetenskapsfilosofi och kvalitativ metod*. Studentlitteratur, Lund

Avision, D. & Fitzgerald, G. (2006): *Information Systems Development: Methodologies, Techniques and Tools*. 4th ed. McGraw-Hill, London.

Brill, N., & Levine, J (2005): Working with people: *The helping Process*, 8/e. ISBN 0-205-40184-8, Allyn & Bacon, Boston

Bryman, A. (2002): *Samhällsvetenskapliga metoder.* 1.3. uppl., Liber ekonomi, Malmö. 502 s.

Hatch, Mary Jo (2002), *Organisationsteori Moderna, symboliska och postmoderna perspektiv,* Studentlitteratur, Lund. ISBN 91-44-02128-3

Lundahl, Ulf & Per-Hugo Skärvad (1999*): Utredningsmetodik för samhällsvetare och ekonomer.* Studentlitteratur, Lund 1999.

Microsoft Corporation (2005): *Upgrading Visula Basic 6.0 Application to Visual Basic .Net And Visual Basic 2005.* 694 s.

Falk T. & Olve N-G. (1996): *IT som strategisk resurs*, Liber Ekonomi
ISBN 91-47-04119-

Evjen, B. Lhotka, R. Hollis, B. Sheldon, B. Sharkey, K. McCarthy, T. & Ramachandran, R. (2006): *Professional VB 2005.* Wrox, Indianapolis

Artiklar & Journaler

Ahmed, R. (1999): *VISUAL LANGUAGES: A NEW WAY OF PROGRAMMING: Malaysian Journal of Computer Science, Vol. 12 No. 1, June 1999, pp. 76-81*

E. Baroth and C. Hartsough. (1994): *Visual Programming in the Real World.* In M. M. Burnett, A. Goldberg, and T. Lewis, eds., Visual Object-Oriented Programming: Concepts and Environments, Englewood Cliffs, Prentice-Hall, 1994.

R. Baleviˇcius, A. Džiugys, R. Kaˇcianauskas, A. Maknickas, K. Vislaviˇcius (2006*): Investigation of performance of programming approaches and languages used for numerical simulation of granular material by the discrete element method.* Elsevier B.V.

Bergey, J., O'Brien, L., Smith, D., (2001): *DoD Software Migration Planning* Technical. Note CMU/SEI-2001-TN-012 Carnegie Mellon University, August 2001

Bisbal, J., Lawless, B., Wu, B. & Grimson, J. (1999): *Legacy Information Systems : Issues and Directions*: Trinity College Dublin, IEEE Software – September/Oktober 1999

Booch, G. (1991) *Object-Oriented Design with Applications, Benjamin/Cummings, Redwood City*, California

Gefen & Straub (2000): *The Relative Importance of Percieved Ease if Use in IS Adoption: A Study of E-Commerce Adoption*. Journal of AIS 1 (8).

Gowthaman, K. Mustafa, K & Khan R.A (2005): *Source Code Migration to DOT NET Framework; A Re-engineering Application Perspective.* Information Technology Journal 4(4): 420-427, 2005, ISSN 1812-5638

Ghezzi, C. & Jazayeri, M. (1987): *Programming Language Concepts 2/E*. John Wiley & Sons, New York

T. R. G. Green, M. Petre, and R. K. E. Bellamy. (1991): *Comprehensibility of Visual and Textual Programs: A Test of Superlativism Against the Match-Mismatch Conjecture*. In Fourth Workshop on Empirical Studies of Programmers, New Brunswick, December 1991, pp. 121-146.

Good, D. (2002): *Legacy transformation:* Club de Investigación Tecnológica, Aug 2002

Guindon, R., Kranser, H. & Curtis, B. (1987): *Breakdowns and processes during the early activities of software design by professionals.* In G. M. OLSON, S. SHEPPARD & E. SOLOWAY, Eds. *Empiricial Studies of Programmers: Second Workshop,* pp. 65-82, Norwood, NJ: Ablex.

Hough, D. (1993): *Rapid Delivery: An evolutionary approach for application development*, IBM Systems Journal, Vol. 32, No. 3

Hayes, F. (2002): *Visual Basic hits .NET crossroads*: Computerworld Aug 12, 2002

Hilson, G. (2001): *Microsoft says ease of use behind major VB changes*: Computing Canada Feb 9, 2001 vol 27.

Hovland, T. & Johansen, T. (2001): *Evaluation of Microsoft.NET versus Java 2 Enterprise Edition:* Proxycom AS, Oktober 2001.

Navarro-Prieto, R., & Cañas, J.J. (2001): *Are visual programming languages better? The role of imagery in program comprehension.* Int. J. Human-Computer Studies (2001) 54, 799-829

Ossher, H. & Tarr, P. (1999): *Using Subject-Oriented Programming to Overcome Common Problems in Object-Oriented Software Development/Evolution:* Yorktown Heights, NY 10598

H. Ossher, M. Kaplan, A. Katz, W. Harrison, and V. Kruskal (1996): *Specifying subject-oriented composition*. TAPOS, 2(3):179-202. Special Issue on Subjectivity in 00 Systems.

R. K. Pandey, M. M. Burnett, (1993): *Is it Easier to Write Matrix Manipulation Programs Visually or Textually? An Empirical Study*, in IEEE Symposium on Visual Languages, Bergen, Norway, Aug. 1993, pp. 344-351.

Scholtz, J. & Wiedenbeck, S. (1991): *Learning a New Programming Language: A Model of the Planning Process:* System Sciences, 1991. IEEE Comput. Soc. Press

Scholtz, J. & Wiedenbeck, S. (1993): *Using unfamiliar programming languages: the effects on expertise*: Interacting with Computer vol. 5 no 1 (1993) 13-30

Shu, N. C. (1988): *Visual Programming Language Designed for Automatic Programming*, IBM Los Angeles Scientific Center

Song, G., (1994): *Visual Object-Oriented Programming: Concepts and Environments*. Department of Computer Science, York University

Subramanian, G.H. & Zarnich, G. E., (1996): *An examination of some system development effort and productivity detrminants in ICASE tool projects*. Journal of Management Information Systems, Vol 12, No4

Zerzelidis, A. & Wellings, A.J. (KOMMER): *Requirements for a Real-Time .NET Framework*. Department of Computer Science, University of York, U.K.

Zubeck, J. C., (1997): *Implementing Reuse with RAD tools' Native Objects*. IEEE 1997 Washington

Blom, J. & Ljung, J. (2004): *Dynamisk Webbprogrammering Varför väljer systemutvecklande organisationer ASP.NET?*: Linköpings Universitet; Institutionen för teknik och naturvetenskap

Internetkällor

MSDN(2007 a): *Visual Basic Language Concepts: Visual Basic*
http://msdn2.microsoft.com/en-us/library/2x7h1hfk(VS.80).aspx, hämtat den 2007-04-26

MSDN(2007 b): *Visual Studio: Introducing Visual Studio*
http://msdn2.microsoft.com/en-us/library/fx6bk1f4(vs.80).aspx, hämtat den 2007-04-26

MSDN(2007 c): *Visual Basic Developer Center*
http://msdn2.microsoft.com/en-us/vbasic/default.aspx, hämtat den 2007-04-26

MSDN(2007 d): *Visual Studio Tools for the Microsoft Office System*
http://msdn2.microsoft.com/en-us/library/23cw517s(VS.80).aspx, hämtat den 2007-05-03

Intervjuobjekt

Organisation X
Respondent A
Respondent B

Lunds Energi
Martin Sällberg

Bilaga 1 – Mall för intervju med Organisation X

Syfte med intervjun är att få reda på underliggande önskemål av en tänkt uppgradering till Visual Basic 2005 samt dess kringliggande miljö. Intervjun går ut på att försöka extrahera och utröna faktorer som ligger bakom önskemålet med en uppgradering, för att få fram olika faktorer presenterar vi olika ämnen för respondenterna som vi vill ha svar inom. Dessa faktorer ska sedan jämföras med den litteratur vi hittar som möjligtvis behandlar liknade och eller samma faktorer. Faktorerna ska sedan sammanställas till en möjlig beslutsstödsmodell. Vi använder oss av just Organisation X för att lättare kunna skapa användarfall längre fram i studien. Dessutom passar de väldigt bra in på den studie vi gör då Organisation X funderar över en uppgradering. En av orsakerna till intervjun är även att se om företag anser att det är nödvändigt med en beslutsstödsmodell vi utvärderingen till beslutet om man ska uppgradera till Visual Basic och dess kringliggande miljö eller inte.

Intervjun kommer vara av slaget "kvalitativ". Med det menas att frågorna som ställs kommer vara öppna och svaren kommer till stor del bestå av att du berättar om saker och ting. Det finns således inga rätt eller fel utan det är bara att prata på! Vi har inga direkta frågor utan har snarare ämnen som ska beröras och formar våra frågor allteftersom intervjun fortlöper.

Intervjun kommer att spelas in för att sedan transkriberas och kodas. Intervjun kommer ta drygt 30 min. Självklart skickar vi den transkriberade intervjun till dig så du kan läsa och godkänna den. Är det sedan något som du vill ta bort eller redigera i efterhand är det bara att göra det. Vill man kan intervjun vara helt anonym och även uteslutas helt från arbetet i sin helhet.

Vi väljer att sätta ämnen vi söker information om ur ett Visual Basic och Visual Studio 2005 perspektiv. Vi har framförallt valt att söka information på 3 olika nivåer vilket listade nedan.

- **Visual Basic 2005 och dess egenskaper**
  Här eftersöker vi information som är mer specifika för Visual Basic 2005, på ett tekniskt plan.

- **Teknisk nivå**
  Här söker vi information hur och vad tekniken har för inverkan på en övergång eller ej. Spelar möjligtvis VBs egenskaper in på ett beslut eller ej?

- **Organisatorisk nivå**
  Vilka organisatoriska faktorer tros spela roll vid övervägandet av att migrera.

Finner vi faktorer som spelar roll vill vi även försöka få fram om dessa faktorer påverkar varandra och i så fall hur. Detta skulle möjligtvis ge ett bra underlag för strukturen på vår framtida modell.

Vi tackar så väldigt mycket för att du tar dig tid att besvara våra frågor!

Med vänliga Hälsningar

Johan Frisell & Björn Gustafsson

Bilaga 2 – Mall för intervju med Martin Sällberg (Respondent C)

Hej Martin!

Vi (Johan Frisell och Björn Gustafsson) skriver vår kandidat uppsats för Lunds Universitet. Arbetet utförs hos Organisation X och kommer slutligen förutom den akademiska uppsatsen utröna i en projektrapport specifikt anpassad för Organisation X. Arbetets syfte är att bygga en modell över vilka faktorer som kan påverka valet av en migration till Visual Basic och Visual Studio 2005 (.NET). Modellen belyser olika negativa och positiva faktorer som både är specifika för VB2005 men även sådana som är generella för migration till nya språk. Vi har valt att dela upp faktorerna i två huvudgrupper, den grupp av faktorer som beskriver Visual Basic och Visual Studio 2005 som vi kallar VBVSF (Visual Basic och Visual Studio Faktor) samt de Organisatoriska aspekterna som vi belyser faktorer som ligger väldigt talande på det organisatoriska planet, vilka vi väljer att kalla OF (organisatoriska faktorer). Faktorerna som vi har identifierat än så länge finner du nedan:

VBVSF

- Objekt-orientering
- RAD
- VPL
- Syntax
- .NET

OF

- Inlärningstid av ett nytt språk / verktyg
- Migrationen samt dess verktyg i sig- tid, planering
- Personal
- Konkurrens fördelar, m.m.)

Dessa ganska specifika egenskaper hos VB2005 är sedan indelat i något större mer övergripande faktorer:

- Kompatibilitet
- Underhållbarhet
- Komplexitet
- Säkerhet(Reabilitet)
- Utvecklingstid
- Kostnader

Som i sin tur kanske kan delas in ännu mer övergripande i

- Effektivitet
- Organisation
- Möjligheter

Vi kommer att förklara sambandet mer ingående innan intervjun startar, men du kanske kan få en inblick i vad det är vi undersöker.

Arbetet är dels uppbyggt kring litteratur men även kring intervjuer. Tanken är att intervjuerna ska göras med både de som ska göra en uppgradering och sådana personer som kan ha hjälpt till vid en uppgradering.

Syftet med den intervju vi önskar göra med dig är att dels undersöka om det är några faktorer som vi har missat eller inte tänkt på, helt enkelt vad fokuserade du på när du hjälpte företag att migrera för miljö x till miljö y, vad skulle de tänka på o.s.v. Ett ytterligare syfte med intervjun kan vara att försöka förstå hur dessa punkter hänger samman.

Intervjun kommer vara av slaget "kvalitativ". Med det menas att frågorna som ställs kommer vara öppna och svaren kommer till stor del bestå av att du berättar om saker och ting. Det finns således inga rätt eller fel utan det är bara att prata på! Vi har inga direkta frågor utan har snarare ämnen som ska beröras och formar våra frågor allteftersom intervjun fortlöper.

Intervjun kommer att spelas in för att sedan transkriberas och kodas. Intervjun kommer ta drygt 30 min. Självklart skickar vi den transkriberade intervjun till dig så du kan läsa och godkänna den. Är det sedan något som du vill ta bort eller redigera i efterhand är det bara att göra det. Vill man kan intervjun vara helt anonym och även uteslutas helt från arbetet i sin helhet.

Vi tackar så hemskt mycket för att du tar dig tid att besvara våra frågor!

Med vänliga Hälsningar

Johan Frisell & Björn Gustafsson

Bilaga 3 – Intervju med respondent A

Genomförd: 2007-04-05

J = Johan Frisell
B = Björn Gustafsson
A = respondent A

Kodningsmall och förkortningar:

* K.M = Kostnad Migrering
* M.V = Migrering Verktyg
* K.I = Kostnad Inlärning
* I = Inlärning
* PF.P = Politisk Faktor Personal
* PF.K = Politisk Faktor Konkurrens
* VPL = Visuellt Programmeringsspråk
* OO = Objekt Orientering
* RAD = Rapid Aplication Dev.
* .NET = .NET
* S = Syntax
* T.M = Tekniska Möjligheter
* TdM = Tidigare Miljö

TdM	J: då börjar vi att o fråga vad ni har för utvecklingsmiljö idag? A: vi använder visual basic 6.0, mm B: mm J: mm A: som är det huvudsakliga redskapet, sen har vi också redskap i sql server J: mm A: där det finns en del programkod man kan göra också, framförallt det man kallar stored procedures i sql, sen har vi specialprogram för matematisk optimering som ligger helt separat J: m A: som e ja som e väldigt specifik B: m, okej J: något speciellt, någon speciell anledning till att ni valde den miljön från början alltså visual basic 6 då? A: ja därför att e innan dess så fanns det visual basic 4 J: m A: och innan dess fanns visual basic 3 B: och ni har kört dem också? A: jag vi körde dem från början B: jaha ja det var ju faktiskt intressant. A: från början, i början på 90-talet så, de var då windows kom B: mm A: och det var då den första versionen av programvaran gjordes, då baserade vi den på windows 3.11, microsoft visual basic 3.0 men ingen databas. J: ookej B: hur lagrades datan då?. I textfiler? A: i textfilerna! J: mm okej A:B: J: —”pratar i munnen på varandra” A: va sa du? B: det har hänt en hel del sedan dess! Ha

	A: ja visst..Sen migrerade vi då till 4.0 , 6.0 visual basic B: hur gick den migreringen till? A: aa den va rätt så okomplicerad B: den var det… A: det var inga större problem, de va mycket enkelt J, så de var bakåtkompatibelt? A: ja de får man nog säga, inte helt o hållet men nästan B: nä men så gott som A: däremot var den stora förändringen när vi skulle göra ett helt nytt system, i samband med att vi gick över till 6.0 gjorde vi ett helt nytt system helt enkelt,
TdM	med databas, så det var ett helt annat system vi gjorde då helt enkelt. J: öhumm, ny struktur då? A: allt va nytt
K.M	B: hur lång tid tar det att göra en sådan grej, alltså att börja om från början? A: A de var en stor grej, det var många manår B: de va det?` A: ja de va det J: om vi går över till det här 2005 då, om du försöker tänka dig tillbaka till den första tanken ni hade med en övergång då, vet du vad det var?’ A: övergång till? J: till visual basic 2005, eller .NET
TdM	A: .Net, ja…….Ja för oss är det natruligt i och med att vi legat i 6.0 B: mm A: så kan man ju tro att nästa version av visual basic ska innehålla lite förbättringar o olika möjligheter som finns åå då tittar man ju naturligtvis på det, de har vi tittat ee ett antal år nu, men varje gång har vi blivit avskrämda,
M.K	eller vad heter det. Skrämda av framförallt arbetsinsatsen. B:,mm B: men hur var det då, alltså det måste väl ha varit något liknande när ni bytte
S	mellan 3.0 till 6.0 eller förlåt jag menar 5.0 till 6.0 att det bara flöt på. A: där fanns inga problem, det bar flöt på J: Men ni har försökt er på en uppgradering till 2005 då, B. nej J: eller till .NET dvs A: Nej J: eller tittat på det så att säga A: vi har tittat på det B: mm A: åå hört med folk i branschen om olika saker å B: mm A: det har inte varit uppmuntrande J:. nej B: nej A: men detta sista var ett par år sedan sist J: okej A: oo då la vi det på hatthyllan tills vidare J.mmm,,,,Så ni har tittat på det men aldrig satt igång att arbeta med att göra det så att säga A: nej för tecknen var inte tillräckligt lovande J: jaha okej

	B: Hur länge har ni arbetat med miljön 6.0? A: eee det är sedan år 2000 ungefär B: okej, så det är typ nästan 7 år A: ja det är rätt lång tid B: mmm J: mmm
T.M ?	A: så vi känner väl att 6.0 säkerligen kan vara med ett par år till va men vid något tillfälle så börjar den bli för gammal helt enkelt, av olika anledningar o då e det klart att då måste man ändå göra något, men vi är inte där ännu att vi måste göra något, men däremot så kanske vi vill göra något ändå B: ja J: innan det blir ett måste så att säga A: mm J: om du spontant försöker säga vad du förväntar dig av 2005, något specifikt så där bara
.NET, RAD	A: ja det är olika saker, dels förväntar jag mig att man har lite mer effektiv programmeringsmiljö J: mm
T.M	A: allmänt sätt ur programvaruframtagning, vi utgår ifrån att den måste vara bättre, annars har man misslyckats kapitalt när man gjort det B: haha ja verkligen
T.M, .NET	A: men jag förväntar mig mer än så, öö jag förväntar mig att det finns ehh stöd för många nya möjligheter att kombinera en viss funktion som man vill ha utfört med användarinteraktion de e väl det jag är mest intresserad av egentligen, ungefär som om man tänker sig gamla dataspel som egentliggen gör samma sak, med samma regler men som om man jämför med en gammal dos miljö där man fick , ja, trycka på tangenter o tangentkombinationer, medan nu har man spel som är väldigt mycket mer avancerade, mycket grafisk miljö o.s.v., man upplever sakerna på ett mer intuitivt sätt, så att min tanke är då att de sakerna vår programvara gör kan man presentera på en väldigt många olika sätt, tråkigt eller trevligt, medryckande o intuitivt, vi har kommit
T.M	en bra bit på vägen med nuvarande programvara, men det finns ingen anledning till att tro att det inte kan bli mycket bättre B: allt kan ju förhoppningsvis bli bättre. ¨A: ja så kan man se det rent filosofiskt naturligtvis, ”allt kan bli bättre”, o i det här fallet borde det finnas möjligheter, om man ser på datorspel idag utvecklas med world of warcraft, o allt vad det är för något, de e det jag kallar för lite ”lull lull” B: haha J: hahah A: men det är ju väldigt roligt ändå, de som håller på med det tycker det är stimulerande B: a absolut J: a det är ju säkert en viktig aspekt J om vi tänker på den gången ni tittade på VB. NET då så att säga, då ni tittade på problem o så där, förändrades din syn på .NET då eller? A: mot innan? J: ja för flera år sedan A: njea alltså det……mot nu? J: a precis, mot nu

	A: nä för jag vet inte mer nu än vad jag visste på den tiden
	B: men när ni fick, när ni frågade folk inom branschen, vad det något specifikt de nämnde förutom att det skulle ta lång tid?
	A: svårt och ta lång tid!
M.K	B: svårt och ta lång tid!
	A: ja, o vi tittade på en del material också, dels microsofts egna tankar om det hela, men sen tittade vi på en massa andra som hade synpunkter utifrån egna erfarenheter, det pekade i en annan riktning faktiskt, att eeh, det var komplicerat helt enkelt, och att det gäller att göra rätt också.
	B: absolut
	A: så min tanke har väl vara lite att…Så här kan det inte va, de e för dåligt, och att det finns en väldigt massa programmassa här i världen som är visual basic , säkert 6.0 mycket fortfarande, så att göra en sådan dålig övergång till .
M.V	NET det är ju ingen bra lösning de måste hitta på något där helt enkelt, och
TdM	det måste ju finnas en stor marknad för verktyg som konverterar, så jag bara väntar på att några kloka personer hittar på den bästa konversionsmöjligheten.
	J: ja nä absolut
	J: vad tror du de största svårigheterna skulle ligga i en övergång
	A: a de har ju berättat vad det var för något men, det hade något med arrayer att göra, dittan o dattan, som jag inte kommer ihåg helt nu.
	J: men du tror att det ligger på en tekniska nivå?
	B: programmeringsbasis?
	A: Ja
I	J: så du tror inte det har med inlärning att göra?
	A: nej det tror jag inte, eller du menar att jag skulle tro att det tar lång tid för vår personal att lära sig det hela?
	J: aaa
	B: ja
I	A: nej det tror jag inte, de kommer lära sig det på några dagar
K.M	J: ja okej, du tror den största faktorn är den tekniska biten, överföringen av programkod.
	A: sett som svårighet ja
	B: mm
	J: mm, och kostnad också?
	A: mm
	J: mm
	A: så e det ju
	J: finns det något som du, nämner det här redskapen så att säga, finns det något annat du kan tänka dig som kan hjälpa till vid en övergång, förutom just de här programkonstruktions biten så att säga, om du går på den andra delen, något annat som du tror skulle kunna hjälpa dig att identifiera problem och sånt innan, något åt det hållet?
TdM	A: ja sen är det väl vad man vill ha ut av det hela, vilka möjligheter man vill ha, de e möjligt att man kan göra några förändringar vid övergången för att förbereda någon ny förändring eller något annat, så det finns ju anledning till
T.M	att se på möjligheterna, vad man kan få nytt så att säga. De tycker jag är mest spännande egentligen.
	J: mm
	J: Om man kollar på en sådan här modell då som skulle kunna väga in de olika faktorerna vid en övergång, tror du att den skulle kunna hjälpa dig vid valet av

	en övergång eller ej? Tror du att det skulle vara något att titta på? A: a precis, det skulle mycket väl kunna vara så, man har olika aspekter på cost benifit på det hela sen kan man sätta olika vikter på det, ja säg, de kan ju vara olika på olika företag också. J: a precis, en sådan modell då som man skulle kunna anpassa till vilket företag som helst då!. J: eehh, tror du att VB VS 2005 kommer påverka din organisation annat än produkten om man tänker på er organisation det vill säga, B: sett till personal! A: Ja det tror jag faktiskt J: har du någon tanke om hur?
PF.P	A: ja speciellt om det är många nya grafiska möjligheter så kommer det ju säkert innebära mer jobb på den grafiska biten J:mm B:mm
.NET, T.M	A: det kan jag tänka mig, och sen kan jag även tycka att det hade varit väldigt intressant om den påverkar hur våra kunder arbetar med den. B: över Internet och så vidare eller?
T.M	A: Ja t.ex. eller om man kan ha lite mer, hur ska jag säga, i den dagliga användningen t.ex. lite mer interaktion mellan programvaran och människan så att det är lite mer spännande o dom upplever det som lite mer stimulerande helt enkelt, det får inte bli för tråkigt!! J: nej B:nej
	A: det gillar inte användarna, och om då den omgivande miljön för programmet har höjt sig så måste ju våran programvara inte vara sämre än den iaf, helst bättre. B: ja de e ju klart att man vill sträva efter det J. Tror du den kommer påverka något för programmerarna rent utvecklingsmässigt, inte det här att de lägger ner tid på design, utan att det kommer förändra något för dem, t.ex. att det kommer behövas fler programmerare med tanke på att det finns fler möjligheter, eller tror du det kommer bli så att det blir färre programmerare ?
PF.P	A: ja alltså alla förändringar kräver ju insatser, och ska man eventuellt utnyttja alla nya möjligheter i .NET så krävs det säkert att man får lägga jobb på det, det är nog så. ¨J: okej…Har du tänkt något på metoder, systemutvecklingsmetoder, man pratar ju mycket om objektorientering, man pratar om rapid development, extrem programming, massa såhär tillväga gångsätt, har du tänkt på hur VB / VS skulle kunna förändra någon sådan del av arbetet?
OO	A: jag fattar det som .NET eller VB 2005 understödjer mer klassprogrammering än 6.0, nu har vi visserligen klassprogrammering i 6.0 med, genom diverse tricks o man skulle kanske kunna göra det snyggare, men jag vet inte om det lönar sig att göra det nu när koden finns där, det är möjligt att det inte lönar sig att jobba mer på den punkten, men hade man kunnat göra det från början hade man ju tänkt sig att det såg lite annorlunda ut! Men det är
S	mer en teknisk fråga, jag tror inte vi mår illa av hur det ser ut nu. J: nää, men det är ju ändå en ny metod, negativt eller inte! A: Nya möjligheter som jag inte är helt säker på att vi kommer utnyttja, just vad gällande klassprogrammeringen.

K.M	J: okej J: Skulle du kunna uppsatta en tid idag vid en övergång till VB 2005? A: ja men det vet jag ju inte, det är ju det som är den stora frågan!! J: har du någon smärtgräns över hur lång tid det får ta? A: njea, det finns säkert, men jag har inte tänkt igenom det så noga, det är ju också en avvägning, om det finns massa fördelar med det, som jag inte heller känner till riktigt idag så kanske man är redo att betala lite i tid också, men extremläget är att det inte finns några fördelar för oss eller ur, användarens synvinkel då är det klart att smärtgränsen blir väldigt mycket lägre. B: ja något måste ni få ut då det är en stor investering J: så det beror på vilka faktorer som identifieras, och så tar man tiden i efterhand A: mmm visst är det så J: har du något mer att tillägga? B: nej vi har gått igenom alla punkterna J: har du något mer som du känner att du vill säga, som du inte fått sagt?
T.M	A: ja det intressanta är ju att se på alla nya möjligheter det kan de, de är ju det som är mest spännande faktiskt, de e ju där det avgörande ligger tror jag. J: tekniska möjligheter B: och grafiska?! A: ja B: ja men är det inget mer att säga så tackar vi för alla svar J: bra svar, tackar

Bilaga 4 – Intervju med respondent B

Genomförd: 2007-04-05

J = Johan Frisell
M = respondent B
B = Björn Gustafsson

Kodningsmall och förkortningar:

* K.M = Kostnad Migrering
* M.V = Migrering Verktyg
* K.I = Kostnad Inlärning
* I = Inlärning
* PF.P = Politisk Faktor Personal
* PF.K = Politisk Faktor Konkurrens
* VPL = Visuellt Programmeringsspråk
* OO = Objekt Orientering
* RAD = Rapid Aplication Dev.
* .NET = .NET
* S = Syntax
* T.M = Tekniska Möjligheter
* TdM = Tidigare Miljö

	J. Då sätter vi igång med hur det ser ut idag, och börjar med att fråga : vad använder ni för utvecklingsmiljö i dagens läge?
	M. Vi kör Visual Basic och sql server, det är egentligen det ända vi utvecklar i.
	J. Vilken version av VB är det ni använder?
	M. 6.0 kör vi.
	B. och i Sqlen?
	M Ehm.. SQL server 200 kör vi hör på kontoret. Men vi arbetar även med sql server 2005 och sql 7 ute hos kunder. Eller... sql7 har vi egentligen frångått.
	B. Är sql 7 äldre?
	M. ja det är den som kom innan 2000.
	J. Kommer du ihåg hur ni beslutade er för de här verktygen.
I, TdM, S	M. Ehm, nej, eller ja lite gran, det fanns en version av programmet och den var gjort i vb redan, vb4.Så det var önskemål när jag började jobba att det skulle i vb också, eftersom det kunde folk redan. Sedan fanns det en tanke om att det är lätt att ta sig in i för en oinvigd, att börja jobba med.
	B. Hur kommer det sig att ville uppgradera?
PF.K	M. Jag vet inte, det var före min tid. Den gamla versionen jobbade med filer t.ex. inte med databaser. Så det fanns en tanke om att göra något nyare modernare och att det skulle lagras i en databas. Men jag tror inte det fanns någon som egentligen visste varför man skulle använda en databas. Det var bara för att det var det som gällde. Men det här var innan jag började arbeta här. Så när jag kom fanns redan beslutet om VB 6
	B. Men dom grundade inte beslutet om vb 6 och sql mer än det var häftigt?
	M. Nee jag antar att det var mycket problem med textfiler, det funkade väl, men jag vet inte. Tror det var stora risker, avbrott i nätverket och liknande. Att det strulade på det viset.
	J. Det kanske kan ha att göra med att det var svårt att hitta i filerna osv?
	M. mm t.ex. Men eeh. Det var ju nån systemvetare som sa att man ska inte använda filer utan det ska vara en databas.
	J. Om vi går över på eh, typ VB/VS 2005.
	J. kommer ni ihåg var er första tanke var med en övergång?
	M. Asså, med .Net då? Eller VB...?
	J. Ja med ett byte av hela utvecklingsmiljön. Om du vet vad den är?

.NET, T.M	M. Ja tanken asså, om vi tar från ledningen eller respondent A:s sida så är det om det finns grejer i det som kan underlätta för kunderna. Om det är lättare att lägga in Internet förbindelser och andra grejer, så han kan sitta hemma t.ex. och på ett enkelt sätt koppla upp mot databasen, mer sånna grejer. För min del är det väl lite mer... Det
PF.P OO	är en nyare programmeringsmiljö. VB6 har ju en del brister. Vi programmerar objekt-orienterat men man får ju trixa till det lite grann. VB6 är ju ett hybrid språk så man får göra rätt många nödlösningar som redan finns.
	B. Men hur fungerar de här nödlösningarna? För objekt-orientering? Hur kommer man runt det?
	M. Asså det går... man får ju. som ett exempel. I vb6 kan man ju inte kopiera objekt t.ex. Det går inte Om man skickar ner grejer i funktioner och sånt, så kan man skicka ner ByRef eller ByVal. Ref är referenser och val är värden. Det fungerar ju på alla datatyper. Men så fort det är objekt så blir det alltid ByVal. Så då får vi sitta o skriva egna kopieringsmetoder så att vi på ett enkelt sätt kan skriva .copy. Men därimellan har vi skrivit kod för att...
	B. kunna göra det.
T.M, OO	M. mm precis. Men sen är det ju sånt som arv osv. som finns i riktigt oo. som inte finns i vb6. Så på det sättet tror jag det är enklare att programmera. Sen är det ju sånt med tidningar osv. Förr hade vi varje vecka programmerings tidningar med Visual Basic osv. Det finns ju inte tips nu utan allt är med vb.net och inte vb6 nuförtiden.
	B. Ja precis.
	M. så det har blivit... [Björn hostar ljudet oupptagligt..].
	B. Jaa
	J. mm, så det vill kunna skicka ner "ByObj" d.v.s.?
.NET, T.M	M. Eeehm jaa, precis. Något åt det hållet. Men det är ju bara en liten grej. Men det är ju en bit. Sen samma sak. jag har hört att det finns hela dom här bitarna med E-mail klienter osv inbyggt. Det finns ju inte i gamla vb utan vi får arbeta en hel del kring sånna saker. Vi har ju önskemål att kunna göra larmsystem i vårt program som ska kunna skicka larm till oss från kunderna.
	B. mhm
T.M	M. Det får ju med den miljö vi har idag får vi lösa det med att skicka med dom verktyg som finns i sql servern. Medan man med en senare version av vb kan skriva en egen liten klient ganska enkelt som kan ta hand om sånna grejer.
	J. ok
	M. så det är lite sånna tankar jag har.
	J. Så det är ganska mycket tekniska och metodmässiga bitar du söker i...
PF.P	M. Ja precis. O sen jag bara folk som har varit här och arbetat med VB o senare miljö, som säger att det är som dag o natt. Att miljön är så mycket bättre att arbeta i. Men jag har inte själv arbetat med det egentligen, jag bara gissar.
	J. Men eh. har dom sagt något om vad det är som är bättre?
OO, .NET	M. Ja det är dels dom har grejerna som jag har sagt nu. Det är ju helt objektorienterat. O sen de här bitarna med Internet och kommunikation. O sen bara det att miljön är snabbare och enklare att arbeta med.
	J. Lite mer inspirerande?
	M. Jag tror inte ens det... utan bara det att ehm,... eh jag vet inte, svårt att säga.
	J. Hög återanvändbarhet, objekt-orientering osv?
	M. Ja det är det. Jag skulle tro det iaf.
	B. Sen är det alltid lite roligare med en ny miljö?
TdM, PF.P	M. Ja visst är det så. Jag har arbetat med vb6 i 6-7 år nu så det börjar bli lite tråkigt. Sen ser jag alla brister också, det finns ganska mycket buggar i vb6.

	B. Fortfarande?
	M. jaja visst. O dom gör ju inget åt det, de bara ligger där... Dom buggar som ligger där dom kommer aldrig försvinna.... O buggar finns det! Det finns säkert i nyare också men, där rör dom till det efterhand.
	B. Ja.
	J. Har ni försökt er på en uppgradering någon gång?
	M. nej det har vi inte, vi har ju kollat på andra företag som har gjort det. Men jag vet inte, det är ju lite svårt att. Man kan ju inte bara titta på ett annat företag fråga hur dom har gjort och hur det gick för dom. Det har ju mycket med att göra hur vad dom har för produkter och hur dom har programmerat i grunden och så.
	M. Så det har vi inte gjort.
	J. Nej.. Om vi tänker spontant, så förväntar du dig mycket objektorienterat och tekniska specifikationer. Är det något annat som du förväntar dig, någon mer organisatorisk förändring av införandet?
PF.P	M. eeehmm. neee, eller asså jag tror det kan bli lättare att få in nytt folk här. Det finns ett visst motstånd för nya när man säger att dom ska sitta i visual basic 6. Så det tror jag kan vara en fördel också. Det påverkar ju organisationen lite grann. Jag vet inte... Vi får ju alla lära om oss lite grann, men samtidigt, men jag tror alla här måste lära om oss lite grann, men samtidigt tror jag inte att gå över till VB.Net eller 2005 inte är
TdM, I, S	sådär jätte svårt om man har suttit i VB6. Men det är ju så generellt, kan man programmera så handlar det bara om lite syntax grejer man ska lära sig när man går över till ett nytt. Annars vet jag inte om det finns några andra organisatoriska aspekter med att gå över till VB2005.
	B. mm ok.
	J. Har du kollat något på VB2005 under tiden här nu under åren. Om du har försökt läsa sig, inte hört med andra utan kollat själv?
	M. Nja inte direkt... inte så mycket. Mest det här om hur det går till alt konvertera vad
TdM	det finns för problem. Det har inte funnits något större önskemål från vårt företag att byta till .Net eller 2005. Men de brister jag har hittat är, eller inte brister men problem som kan uppstå är att den här datatypen variant som finns i vb6 inte finns i2005. Så där får man hitta på andra lösningar.
	B. Aa just det.
	B. mm
	M. Och variant har vi använt för mest för att vi behöver någon typ av... asså det är lite som att arbeta med objekt i variant också. Eftersom det inte är helt objektorienterat så får man ha lite knep.
	J. Kan du tänka dig några negativa aspekter med vb 2005?
M.V	M. Ehhm. Bortsett från att det är nog ganska svårt att göra om hela vår produkt till ny kod, så är det... det är väl egentligen det ända. Annars kan jag inte se några sånna direkta .
	J. Inga brister?
	M. Nej jag vet för lite om det. Men det är... men jag tror inte. Eller, allt som finns i vb6 finns ju även i senare version. Misstänker jag också. Av dom man använder.
	J. Vart tro du de största kostnaderna finns med i en övergång.
	M. Ja det är juu.... själva utvecklingskostnaden. Om vi ska bygga om vårt program får
K.M	vi ju inga pengar för det av kunderna. Det är ju så att om vi ska hitta på en ny funktion till vårt program så frågar vi ju kunderna om det är något de vill ha, så säger de ja, sen får de betala lite för utvecklingstiden o.s.v. Att konvertera programmet är det ju ingen som betalar för så där ligger en stor utgift... i utvecklandet.
	B. Men kan man inte se det som intäkter istället? Eller man kollar kanske inte på det

	så kanske?
PF.K	M. njaaa, nej det tror jag inte. Om vb2005 hade varit ett försäljningsargument så hade det varit det. Men jag är inte säker på att det är det... svårt att säga. Om man kollar på
PF.K	andra företag och deras företag och deras produkter så stoltserar de ofta med att de har programmerat i .Net miljö. Även om det inte har någon betydelse för användaren direkt. Så är det fortfarande ord som de stoltserar med. Så i den bemärkelsen kanske
RAD	man kan säga att det är en ekonomisk fördel, men inte annars... Eller det skulle vara om vi kan spara utvecklingstid osv.
	B. Om det går snabbare att programmera?
RAD	M. Ja precis. Om man slipper göra långsamma lösningar. Sitta o skriva samma kod om och om igen.
	B. Ja det är ju inte bra, eller tröttsamt om inte annat.
	M. Ja precis.
	J. Finns det något som du tror skulle hjälpa dig vid en övergång?
M.V	M. Eeehm... det är .. det finns säkert konsultfirmor som kan göra det. Hjälpa till. men assåå.. eeehm. Jag tror inte det. Det skulle vara om man konverterar med hjälp av olika
I	verktyg så kanske man skulle kunna behöva hjälp med hur dom fungerar. Men annars handlar det mycket om hur vår kod fungerar.
M.V	J. Men du tror det kan finnas verktyg som kan hjälpa er?
	M. Ja det tror jag. Det är den uppfattningen jag har fått iaf.
	B. Migreringsverktyg osv.
	M. Ja eller vilka ord man nu använder. Konvertingarverktyg kanske...
	B. Men då är det framförallt konsultfirmor som ni är intresserade av att hyra in eller?
I	M. Eehm babababupp. Ja. Det är vad jag skulle tippa på iaf. Eventuellt gå kurser då isf.
	B. Annars skulle det vara du och person y som satt här och gjorde det?
K.M	M. Mmm, ja, ja precis. Eller om vi har andra programmerare som sitter här och gör det också.
	J. Tror du det skulle innebära att det kommer fler anställda till organisation X om ni...
PF.P	M. Njaa asså jag tror det beror på hur många vi är beredda att anställa, men jag tror att det blir lättare att få hit folk om vi byter till 2005.
	J. Men jag tänkte mer... Tror ni att ni kommer bli tvungna att anställa fler?
	M. Jaså du menar så... Jaaa, eller. Det är inte helt omöjligt.
	B. ok.
	J. om vi kollar på en sån här beslutstödsmodell med olika faktorer. Både positiva och negativa. Är det något som du tror skulle kunna hjälpa dig vid en övergång.
	M. ja det tror jag. Asså jag vet inte så mycket om sådana här modeller men man måste ha något att bygga fakta på. Bygga beslutet på. Vad det finns för nackdelar och fördelar och vad de olika bitarna kan leda till.
	B. Typ som ett utvärderingsverktyg?
	M. Ja något sånt ja.
	B. jaao. blupp blupp
	J. Då ska vi se... Vi har varit inne lite på det innan, men om vi bortser från den tekniska biten tror du en övergång skulle förändra ditt arbetssätt i övrigt? Tror du att du kommer få arbeta mer med något? Typ mer med den grafiska biten eller något i den stilen? Tror du det kommer innebära någon sån förändring?
	M. Nej det tror jag inte.
	J. Det tror du inte?
	M. Nepp.
	J. Det kommer vara samma typ av...

	M. Mm precis. J. Eh om du skulle uppskatta hur lång tid det skulle ta, en sån här uppdatering. Hur lång tid tror du det skulle ta? Eller gissa snarare. M. En kvalificerad gissning? J. Hehe ja precis. M. Eeehm. Ja det är faktiskt jättesvårt att säga.... eeeehm, heltidsarbete. eeehm. då
K.M	hela produkten... jag skulle tippa på att det tar... eeehm. Ouph, jag vet inte..... ett halvår kanske. J. Men det är månadsbasis iaf? Inte år? M. Nej det tror jag inte. Eller det hoppas jag inte iaf, eller jag tror.... Men sen i kalendertid kanske det blir längre tid då.. J. Mm M. Men jag tror ett par månader i ren arbetstid iaf.
TdM	B. mm. Säkerhetsaspekter. Det är inget du har kollat på? Om det kanske är säkrare att programmera i ett modernare språk. Jag tänker kanske på buggar osv. M. Eeehm. eller... nej det har vi inte funderat så mycket på.. B. Det är ingen kritisk faktor för er? M. Njaaa, nee om buggar finns så är det inget som syns för slutkunden utan det är mer för oss att det blir lite krångligare att komma runt osv. B. ok. J. Tror du det finns någon smärtgräns för hur lång tid det får ta? Någon maxtid eller liknande? M. Ehm, inte i tid. I pengar tror jag. Om man tänker på en anställd som sitter med det
K.M	på heltid så ska han ha lön. Jag tror det är där det ligger. Men jag är inte insatt i det ekonomiska så jag vet inte riktigt. B. Ok. M. Men det är ju så att allt som inte bringar in pengar så på sekunden det vill man ju
K.M	inte jättegärna.... B. Klart det vill man ju inte... M. ne precis. B. Hoppa på liksom. M. mm. J. Ja.. Är det något som du känner att vi har missat som du gärna vill ta upp? M. Eeeh. dudu. Eller andra tekniska, jag ser det ju mest ur utvecklingssynpunkt. B&J. Mm M. Och vad jag tror är att det kan finnas andra färdiga komponenter eller bitar som man kan integrera i vårt system. eller i själva miljön. Ibland köper vi komponenter
T.M, .NET	eller vissa bitar som är gjorde i Visual Basic eller annat som man kan ta in. B. Ja asså deras funktioner osv. M. Ja precis man behöver inte skriva sådant som redan finns, jag tror att den biten blir lättare också i 2005. J. Känner du igen begreppet RAD? M. Eh nej, RAD? J. Ja RAD är Rapid Application Development, det är mycket sånt som du precis pratar om, färdiga lösningar osv. M. Ja ok. J. De kan vara färdiga lösningar, snabb framtagning av prototyper osv.
RAD	B. Är det något som är relevant? M. Ja det tycker jag, ja visst! Så att kunna programmera snabbt och kunna få ut produkter snabbt är viktigt för oss.

	J. En sån funktion kanske skulle kunna hjälpa er? Så man kan ta fram prototyper snabbt osv? M. Mm. J. Man kanske skulle kunna ta fram prototyper snabbt till era kunder osv? Integrera användaren i utvecklingsprocessen för att testa. M. Ehhm... Asså grejen är denna att vi utvecklar mycket snabbare än så. Grejen är den att vi är alltid intresserade av vad kunden tycker så vi pratar mycket dem dem innan vi börjar bygga, gör specar osv. Och sen skärmbilder, det är ju lite prototyper det med. J. Mm. M. Men att först bygga en prototyp och sen arbeta i den, sen diskutera, det är lite för långsamt för oss. B. Ok. M. Så det känns inte så relevant. Och gör vi en prototyp så gör vi den som kunden vill ha den så det redan är en färdig funktion, sen ändrar vi i den. J. mm. Det är mycket den typen av prototyper man pratar om i RAD. M. Ja ok. Det är egentligen lite så vi arbetar. Men vi kallar det inte för prototyp. Utan vi säger att den är färdig så installerar vi den, så får kunden komma med invändningar så ändrar vi den efter det. J. Ok intressant. Jag har egentligen inget mer att säga. B. Nej det är väl inte... Jag vet inte om det är någon betydelse för dig som programmerare vilken miljö du sitter i om det är mer stimulerande? M. Eeehm. Du menar att utveckla i? B. Ja eller om det helt enkelt blir roligare att gå till jobbet.
PF.P	M. Ja naturligtvis. Det är lite därför jag är intresserad av att gå över till Vb2005, som jag sa det är lite tradigt att köra i samma miljö fortfarande. Jag har kört i den i 7 år nu så det börjar bli lite tradigare. Man behöver alltid lite nytt, utvecklas. B. Ja nej, kanon! J. Vi tackar så hemskt mycket för dina svar! Det var jättebra svar!

Bilaga 5 – Intervju med respondent C

Intervju med respondent C (Martin Sällberg)

Genomförd: 2007-05-02

M = respondent C
J = Johan Frisell
B = Björn Gustafsson

Kodningsmall och förkortningar:

* K.M = Kostnad Migrering
* M.V = Migrering Verktyg
* K.I = Kostnad Inlärning
* I = Inlärning
* PF.P = Politisk Faktor Personal
* PF.K = Politisk Faktor Konkurrens
* VPL = Visuellt Programmeringsspråk
* OO = Objekt Orientering
* RAD = Rapid Aplication Dev.
* .NET = .NET
* S = Syntax
* T.M = Tekniska Möjligheter
* TdM = Tidigare Miljö

	J: vi börjar med att säga med vad du har arbetat tidigare med så att säga M: Ja det kan vi ta J: Din erfarenhet av vb M: jag började med vb 3, som det hette på den tiden 1996 på Organisation X och vi gick snabbt över till vb 4 och höll på med det så länge jag jobbade där, fram till 98 tror jag, sen efter det jobbade jag på Mandator där avdelningen bytte namn Semcon våren 2004 och där började vi med vb 5 men gick snabbt över till vb 6, jag tror det var 2004 vi började titta på .NET och möjligheterna med att gå upp där, påbörjade även en uppgradering där 2005 och vi var inte riktigt klara med den när jag slutade men delar av den stora programvaran var uppgraderad och levererad.
PF.P	J: Nu kommer en ganska svår fråga kanske, men vet du varför ni valde att uppgradera?
Kompatibilitet	M: E, varför vi ville uppgradera var ju ganska självklart för oss som var anställda och jobbade där, vi såg ju att alla andra uppgraderade och att vi inte kan ligga kvar i ett gammalt verktyg som det kanske inte finns support på om några år, de kommer nya operativsystem, nya Windows plattformar och då är det ju inte säkert att dom gamla versionerna kan köra i all evighet och då kan man inte stå där o uppgradera allt pang boom på en gång, det är en väldigt lång process. Ledningen var ju lite emot det hela, de såg ju att detta kommer ju att kosta massor med pengar, så det var nog de anställda som fick köra en
M.K	övertalningsprocess där rätt länge innan vi kom igång och det gick till så att jag och en till fick efter lite tjafs då tillslut gå en kurs iaf i VB.NET, där vi efter den
Inlärning	skulle rapportera vad detta innebär och så vidare och de gjorde vi…..vi började skriva en del små exempelprogram, egentligen bara programsnuttar som gjorde liknande saker som vårat gamla program i vb.net för att få en uppfattning om vad det innebar och vilken tid det tog och så vidare, vi provade att köra ett uppgraderingsverktyg som Microsoft påstod att ” kör ni detta så skall koden
M.V	uppgraderas automatiskt från vb 6 till vb.net” men det fungerade inte över huvudtaget tyckte vi, inte på de programmen vi hade, inte ens på de enklaste.

Tidigare Miljö PF PF.P PF.P Kompatibilitet M.V	Nu vet jag inte riktigt vad det berodde på, kanske för att vi körde en del active X komponenter och inte en helt objektorienterad struktur och så vidare men även mycket mycket enkla program klarade den inte av. J: okej J: Om vi försöker kolla på vad det var specifikt, var det några egenskaper hos .net som ni saknade i tidigare miljö? M: När vi började diskussionen hade vi väldigt lite kunskap i .net, nästan ingenting mer än att det är nästa generation vb som kommer här nu, och varför ska vi inte ta den? Även som utvecklare kände vi att vi inte ville sitta kvar i det gamla för då har vi inga möjligheter att få jobb om 10 år, då har vi inga kunskaper om verkligheten och vad det är som gäller då. Det var ju också en drivkraft även om den inte nämndes högt så att säga. B: Det kan man ju faktiskt förstå J: ja M: och så är det ju också att hade vi inte fått påbörja den uppgraderingen så är det nog många som hade slutat. Nu slutade jag ändå men inte av den anledningen så att säga. J: Så det var dels en faktor för personalen, att det var modernt, lite nyare grejer?! M: Ja och man såg ju också att ville man använda någon inköpt ActiveX komponent så som vi gjorde i vissa fall, typ grafkomponenter och sånt, dom vi hade då hade ju fortfarande stöd för den gamla miljön men man såg ju att det kom fler och fler nya verktyg som bara var gjorda för .NET som vi då inte kunde använda och det var ju en begränsning. J: när ni gick en övergång, det ända ni hade som de här migreringsverktygen de va ingen hjälp alls? M: Nej J: så det ni hade som hjälp var alltså kurser?! M: det kan man säga ja, den avgörande faktorn att vi faktiskt fick uppgradera sen var att istället för att välja vb.net att välja C#, och likheterna mellan vb.net och C# är ju, jag skulle vilja säga att de är till 99 % lika om man bortser från syntaxen och vi hade ju många som programmerade i c++ och som bara kunde c++ och om vi då valde C# är ju den syntaxen samma som c++ delvis iaf. Det var vårt slutargument som fick ledningen att säga ”okej vi kör på detta, de verkar bra” att då kommer dom att få lättare att läsa, vb kod blev det inte men den koden vi gjorde för användargränssnittet. B; Men tror du att det hade begränsat dem mycket som programmerare om ni istället valt att byta till vb.net? M: Då hade nog det nog sett ut som innan att vissa höll på med c++, men nu fick vi en spridning så alla kunde hålla på med alla olika delar. B: så ni som var vb utvecklare fick övergå till c# M: ja B: Och hur var det att övergå mellan de två språken? M: Som jag sa så började vi ju på den har .net kursen och började skriva små snuttar program, och sen när vi gick över till c# så hade vi som gjort detta redan skrivit lite c++ så man kunde syntaxen där lite. Steget vb 6 till vb.net är mycket mycket större än det mellan vb.net till c#. J: när ni väl hade påbörjat den här övergången, hittade ni något ytterligare som var positivt om man kollar på de här tekniska egenskaperna så att säga? M: Ja det märkte man ju många, många grejer helt enkelt, allt från små detaljer

	i hur man utformar användargränssnittet, hantering av fel på ett smidigare sätt,
OO, VPL	den största fördelen var att man tvingades skriva snyggare kod,
Säkerhet	objektorienterad kod. I vb 6 var det betydligt enklare att slarva med objektorienteringen och komma fram ändå.
	B: men ni programmerade ändå objektorienterat i vb 6?
	M; ja delvis, halvt om halvt.¨
	J: Så objektorientering var en positiv faktor eller att ni blev tvingade till objektorientering?!
OO	M: Ja man kan säga det, bättre struktur på koden, sen kan man också säga att om man tar ett vb 6 program och tittar på hur det är skrivit och hur det är patchat i många år och säger ”nej nu måste vi skriva om detta från början” så hade det även om du skrivit om det i vb 6 en gång till så hade det blivit mer strukturerat, så det är lite svårt att säga vad som är vad där.
	B: Men kan man säga att objektorienterad programmering är mer strukturerad, lättare att skriva snyggt än tidigare versioner typ vb 6.0?
	M: ja sen finns det hjälpmedel i .NET. som vi köpte in från Borland, kommer inte ihåg vad det hette, där man såg klassdiagrammet samtidigt som man programmerade, där man med en knapptryckning kunde få en översikt över
Kompatibilitet	sina klasser, man kunde gå in och skriva direkt i diagrammet och så automatgenererades det kod, ett väldigt smidigt verktyg.
	J: Ett Uml diagram?
	M: Ja
	J: Om vi försöker titta på fler fördelar som ni kände. Om vi kallar de här Visual Basic 6 då, som visuella programmeringsspråk, man sitter i en programmeringsmiljö där man kan lägga till knappar och ett gränssnitt som är
VPL	mer visuellt då, blev det några sådana förbättringar då?
VPL	M: Ja det tycker jag, i början tog det ju klart lite längre tid, men kan säga den
VPL	första månaden, det man skrev där kan kanske ha skrivit under en vecka i vb 6 men det man skrev den 12 månaden hade tagit 1.5 månader att skriva i vb 6 så på sikt skriver man nog kod snabbare. Och där finns en del finesser just vid design av gränssnitt att man kan ärva även användargränssnitt t.ex. Små
OO	finesser som om att man drar i ett fönster kan man i vb.net få kontrollerna att följa med snyggt de blir större och växer om man drar i fönstret, det var man tvungen att skriva kod för själv i vb 6
	J; så det är en effektivare arbetsmiljö?
	M: Ja där finns en hel del nya egenskaper som när man kommer underfull med dem är de väldigt bra
VPL	J: Om man ser till .NET som sådant så innehåller det en massa små delar så som XML, ADO. Tittade ni något på den, eller hittade ni några fördelar där?
	M; Ja där var ju integrerad xml vilket var ganska bra med tanke på att vi skrev en del filer för olika leveranser en sorts .ini fil som man nu kunde lägga i xml
.NET,	istället.
Kompatibilitet	B; okej okej, något annat som du kommer att tänka på?
	M; Ja något annat är det språkstöd. .NET har ett väldigt bra stöd för språk, och detta på ett enkelt sätt.
Kompatibilitet	B: Hur menar du nu?
	M: Man lagrar kontrollernas språk i olika xml filer
	B: okej?
	J: okej?
	M: det är jättesmidigt att inne i studion växla mellan olika språk

	J: Så är .NET mer kompatibelt med olika grejer än tidigare versioner?
.NET	M: de har absolut tänkt på det att de inte bara är USA som är en marknad utan att man kanske vill ha fler språk än engelska. Det är en jättefördel.
.NET	J. mm, Hur kände du med ado.net mot databaser?
Kompatibilitet	M: där fanns mycket mer databas stöd men nu gjorde vi en egen komponent eller assembly som det heter i .NET ändå för access mot databasen
	J: okej
.NET	B: så ni kunde inte använda er av dem som fanns inbyggda i .net?
Kompatibilitet	M: Vi kunde kanske gjort det men i den gamla vi hade, skriven i vb 6 hade vi lagt på så mycket extra funktionalitet som var inne och pillade i Triggrar och allt möjligt som man gjorde i insert och update så man var ändå tvungen att göra något där. Så därför blev det inte den inbyggda databaskopplingen tyvärr.
	J: När du försökte övertala dina chefer, var det själva migreringen då, att det skulle ta tid att skriva om koden som var deras argument emot det så att säga.
	M: ja det var det översatt i pengar
	B: så det var det som vägde emot fördelarna.
M.K	M; ja
	B: Om man kollar lite på tiden att lära sig den nya syntaxen, var det något ni valde att väga in i beräkningarna, att man var tvungen att lära sig programmera objektorienterat!?
I	M: njea det var det väl inte, vi hade väldigt svårt att uppskatta även fast vi började skriva en del så var det oerhört svårt att uppskatta tider och sådant, men det..En kurs var ju att rekommendera för dem som inte kunde, och det är klart det kostade ju en 20000:- per skalle.
	J: men det tror du är det bästa sättet att lära sig?
I	M: Ja jag tror det är det bästa sättet att lära sig, att få det intensivt ett par dagar så man ser vad det är sen sätta sig ner i lugn och ro, man ser ju oerhört mycket på en sådan här kurs, nu gjorde vi så att alla gick inte den här kursen utan vi höll internutbildningar, de som hade hållit på med det ett tag, jag bland annat, höll internutbildningen.
	B: Vi har tagit del av viss litteratur som säger att det blir lättare att lära sig något om man hela tiden har någon att fråga, någon form av mentor än att just sätta sig på en utbildning och få mycket information på en gång. Hur ser du på det?
I	M: jag tror det är bra att börja med något intensivt, men sen är det naturligtvis jättebra att ha någon att fråga för även fast en utbildning är på flera dagar så är det väldigt översiktligt och kanske inte på de områdena man behöver det på ändå.
	J: du sa tidigare att du tvingades skriva bättre kod, när du menade bättre kände du att produkten blev mer stabil, säkrare, mindre fel?
	M: e, njea det vet jag inte det är ju så med all kod men skriver, det gäller ju att se till så man har vattentäta skott överallt och en bra felhantering. Visst kompilatorn sväljer ju en del men det går ju aldrig att bygga en vattentät kompilator.
	J: nä okej, men om du tänker på slutprodukten var den lättare att underhålla på något sätt än tidigare, lättare att lägga till nya grejer?
	M: ja det var det ju, när man gör något sådant här så utgår man ju från kod som flera har varit inne patchat i under flera år och sådan kod blir ju per automatik rörig. Oavsett om man ska byta programmeringsspråk eller inte bör man nog skriva om all kod som lever minst var 10 år de e kanske sällan nog, kanske var

Underhåll-Barhet	5 år. Tittar man själv tillbaka på ett program man skrivit för ett halvår sedan ser man alltid förbättringar man kan göra och det har inte med att man har programmerat dåligt från början att göra utan det kan vara att nya förutsättningar har tillkommit och det kanske man inte ser förrän programmet kommit i drift. B: okej, Om man tänker sig att .NET kan fungera som någon form konkurrensaspekt för företaget ut mot kund, man kan t.ex. påstå att vi är långt framme rent tekniskt för vi sitter i .NET. eller man ser inte på det så kanske? M: Njea, det är ju bra att kunna säga att man jobbar med de senaste verktygen för att kunna hävda att man kommer finnas kvar även i framtiden och satsar på nyutveckling, det är det alltid bra att kunna hävda, sen är det ju inköpta komponenter, de kommer ju nya flashiga grejer hela tiden, man kanske vill hotta upp sina program med snygg grafik och så vidare och då är det ju kanske
P.K	svårt att göra det om man ligger kvar in någon gammal miljö. J: Nu har vi ju försökt få fram lite positiva grejer kanske men när ni väl gjorde den här övergången, vilka var de största problemen ni stötte på?
Kompatibilitet	M: de va svårt att automatkompilera, de gjorde vi förut, vi byggde all vår kod och kompilerade den på en server under natten, de vet jag att vi hade mycket problem med, en del beroende på de här assemblyna alltså .NETs motsvarighet till .dll och .ocx att de var ju tvungna att vara registrerade eller vad det nu hette, eller byggda på ett visst sätt och inlagda på datorn, så det var en del strul med
Migrering	det.
Td.M	J: mm rent syntaxmässigt var det några problem där? M: nej det tycker jag inte det var J: de gick lätt att föra över kod till nästa så att säga. B: Så det var framförallt databaskopplingarna som krackade M: ja precis J: men du tror de här problemen med assembly har med .NET att göra och inte med själva konverteringen att göra?¨ M: nej det var ju, de e ju ren okunskap också att man inte vet hur saker och ting fungerar när man börjar med något
Migrering	J: mm, intressant B: Fanns det några direkta fördelar för er som utvecklare att sitta i en ny
I	programmeringsmiljö? M; ja de fanns de men det är ju en mycket om en fråga om tycke och smak, men i det nya finns ju en del nya möjligheter helt klart, sen var det ju också en fördel att man satt i samma miljö oavsett vilket språk man än jobbade i. B: ja
T.M, PF.P	J: sen att den var lite effektivare vad vi förstått M: ja J: Om man kollar på den modellen vi har tagit fram, nu är den inte utritad, men då har vi hittat en rad egenskaper hos .NET och de egenskaperna i sin tur kan
VPL	man översätta i de här, kompatibilitet, underhållbarhet, simplicitet, komplexitet, säkerhet som vi va inne på, utvecklingstiden sen kostnader för migrering och inlärning då, samt den konkurrensfaktor för personal samt ut mot kund, är det något du känner som saknas? B: kanske en svår fråga att kasta på dig sådär! M: njea J; något som du känner att vi helt har glömt här! B: något som vi inte har behandlat med under denna intervju!

Td.M	M: där är inget som jag kan komma på så här på rak arm J: tycker du de stämmer överrens med det ni fan samt de ni argumenterade för M: njea de e ju väldigt svårt man kan ju dra in väldigt mycket i dessa begreppen. B: ja de e klart J: ja B: De kan ju vara, något du funderar på, och undrar varför vi inte tagit upp ”detta” under intervjun. J: De kan ju vara ett problem också som vi glömt bort M: Ja jo jag kan komma och tänka på ett problem här nu, det var problem att köra program som låg på en server från en klient, det fungerade inte B: okej M: Utan koden måste ligga på den maskin som exekverar den. J: Okej så du kan inte?.. M: exefilen kan inte ligga på en maskin och köras på en annan J: jaha okej, gick det att göra på tidigare versioner? M: Ja B: När brukar det vara så? J: Sitter och tänker på när det kan vara så i vanliga fall! M: Men jag vet att vi kom runt det på något sätt, kommer inte ihåg hur riktigt, men jag vet att där hade vi problem. J: Men du menar inte att gå in med en viritual server och köra det så utan du menar att. J & M ”pratar i munnen”… M: Om där ligger en exefil på en utdelad mapp helt enkelt J: dubbelklicka på den. M: ja B: de gick inte? M:nej B: fick man reda på varför det inte gick? M: Ja det kom fram: J: så det har med .NETs kompatibilitet att göra? M: ja och det var ju någon säkerhetsgrej i grunden. Microsoft tyckte väl att det var bättre detta. B: kanske inte ändå? M: haha nej precis J: men det gick inte att stänga av på något enkelt sätt då? M: Nej J: hur kände du för säkerheten, blev koden mer säker på något sätt, alltså kunde man använda funktionen för att gömma koden och sådan grejer eller? M: njea inte som man märkte som programmerare i alla fall J: nej, de e intressant för Microsoft presenterar det som någonting säkrare och vi förstår inte riktigt vad det är som är säkrare. Men det kanske är sådan saker då som att man inte kan köra exe filer på en klient, M: ja J: har du något mer att tillägga? B: nej inte vad jag kan komma och tänka på, vi har ju gått igenom våra punkter. J: Vi har ju fått med lite av de här faktorerna då och egenskaperna. Björn listar lite faktorer för att summera J: en fråga jag kan komma på som är lite mer generell sådär, som inte har så

S	mycket med VB att göra. Syntaxen är den något som du känner att, är det viktigt hur den är, alltså hur den är utformad när du utvecklade?
S	M: Njea mer att den är konsekvent, tycker jag J: mmm,.Men om man då tänker läsbarhet, är det viktigt att den är lätt att återgå till och läsa i, lätt att skriva och så vidare M: ja, ja det är ju viktigt, läsbarhet har ju med konsekvens att göra, att man
S	skriver på samma sätt hela tiden så jag tror det är viktigt att ha programmeringsregler som alla följer, att si och såhär gör man att man har kommentarer och vilka kommentarer man har. B: I den miljön ni satt i tidigare, fanns det möjlighet att många satt och programmerade i samma filer samtidigt, typ någon form av team miljö? M: ja vi hade något sådant verktyg i vår äldre miljö. Någon form av Source safe, men det fanns i tidigare miljöer också. B: okej J: ja jag är jättenöjd B: jepp jag med. J: tackar så väldigt för dina svar.