Verktyg för hantering av processförändringar

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Datateknik

Examensarbete:
Christopher Svensson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2009

Sammanfattning

Denna rapport beskriver hur man med Microsofts .NET-teknologier, Windows Workflow Foundation, Windows Forms samt Windows Presentation Foundation skapa en applikation för att generera och redigera workflows.
Företag har en rad olika processer som stöd för sin verksamhet, t ex att skapa ett e-postkonto i företagets nätverk. Oftast följer detta någon form av flöde. Vid omorganisationer, effektiviseringar eller liknande måste dessa processer anpassas till nya krav. Det kan vara allt från att skapa en användare i ett datorsystem, till att installera önskade programvaror på en dator. De flesta ändringar görs idag med dyra licensierade verktyg. Målet med detta arbete är att skapa ett nytt verktyg för dessa processer. Företag med en IT-plattform som använder Microsoftteknologin .NET, kan göra dessa ändringar med workflows. Genom nämnda teknologier samt verktyget Visual Studio kan man tackla detta problem.
Beställaren var ATEA Sverige AB, Sveriges största IT-infrastrukturföretag, som säljer IT-lösningar och tjänster till företag. ATEAs klientplattform är byggd på .NET-teknologi och en av ATEAs andra produkter, Accelerator. Detta process-verktyg utnyttjar plattformen och har möjlighet att använda sig av workflows.
Arbetet skedde i Visual Studio 2008, programmeringsspråket var C# och grafiska gränssnittet gjordes med hjälp av Windows Presentation Foundation. Vid skapandet av denna applikation genomfördes förstudier för att förstå hur teknologierna enligt ovan fungerar, samt undersökning av de lösningar som finns på marknaden. Förutom detta gjordes dessutom intervjuer med beställaren, för att kunna ta fram en kravspecifikation för applikationen. Vid utvecklandet av applikationen följde arbetsgången processmodellen Scrum.

Workflowdesigner är en komponent som är en grafisk yta och redigeringsyta för workflows, vilken gjordes om till en User Control av Windows Forms. Sedan kunde denna importeras till Windows Presentation Foundation. Resultatet blev en applikation som har det nya grafiska gränssnittet, Windows Presentation Foundation. Applikationen kan skapa och redigera befintliga workflows. För att få den mest optimala lösningen måste kärnan till workflow-designern byggas om. Detta blir något som kan göras i ett framtida projekt, liksom att implementera en kodvy för att öka användbarheten.

Nyckelord: Processförändringar, .NET, Windows Workflow Foundation, Windows Presentation Foundation.

Abstract

This report describes how you can create an application that uses Microsoft's technologies, Windows Workflow Foundation, Windows Forms and Windows Presentation Foundation, to create and edit workflows. Companies have all sorts of processes, which support their business. Like create an E-mailaccount in the company's network. When companies reorganize or streamline their business the processes needs to follow the company's new demands. The most of the changes are made with expensive tools. Companies with an IT-platform, which use .NET can do this changes with workflows. With this technologies and Visual Studio, this problem can be solved.
The initiative to this work comes from ATEA Sverige AB, which are the biggest company in IT-infrastructure, which sells IT-solutions. One of ATEA's other products, Accelerator, that uses the platform can handle workflows. All necessary equipments came from ATEA. The work was done in Visual Studio 2008, the programminglanguage C# and the graphical interface Windows Presentation Foundation.

Before the creation of the application, some studies were made to understand the technologies and how these work. One thing that also has been made was examination of the solutions in the market. There were interviews too of ATEA and all the information created the demand specification.
The development of the application fallowed the agile process Scrum, which is perfect for software development. Scrum split the work into sprints where a sprint can be a function in the application.
Workflowdesigner is a component, which edit workflows. The component was recreated to a User Control in the technology Windows Forms. This was imported to Windows Presentation Foundation.
The solution was in the end an application with the new interface, Windows Presentation Foundation, which can create and edit existing workflows. To get the best solution, the kernel of the workflowdesigner has to be recreated from the base. This is something that can be made in the future and a codeview can be added for the usability.

Keywords: Process changes, .NET, Windows Workflow Foundation, Windows Presentation Foundation.

Förord

Först av allt, vill jag tacka min handledare vid ATEA Sverige AB, Andreas Fransson som hjälpt och lyssnat på mig. Ett lika stort tack riktar jag till Lise Jensen, min examinator vid LTH.

Innehållsförteckning

Inledning

Denna rapport avser att ge en förklaring till vad examensarbetet handlar om. Den avhandlar även de mål som funnits samt hur dessa har uppnåtts. Rapporten innehåller en introduktion till ATEA Sverige AB, och .NET Framework med dess tillhörande teknologier, utveckling av applikationen och de problem som uppkommit. I slutet av rapporten finns slutsatser och kommentarer samt framtida möjligheter till utveckling av applikationen.

Det som finns som bilaga, är de standard funktioner som ingår i Windows Workflow Foundation.

1 Introduktion

1.1 Bakgrund och Problembeskrivning

Företag har en rad olika processer som stöd för verksamheten. Exempel på en sådan process kan vara att skapa ett e-postkonto. Detta sker oftast utifrån ett visst flöde genom organisationen. Någon är beställare och någon måste kvittera och skriva under för att ett e-postkonto ska sättas upp. Men vad händer när det blir omorganisationer, effektiviseringar eller liknande på företaget? Då måste ibland processer anpassas till verksamhetens nya krav.

Det är inte alltid ett företag använder sig av IT, för att hantera sina processer. Mindre företag hanterar ofta sina processer manuellt. Men för de företag som har köpt en färdig IT-lösning, som stödjer deras nuvarande verksamhet och inte är modifierbar, blir systemet ett problem istället för en hjälp. För att på ett enkelt sätt stödja att företag ändrar sina processer, behövs ett verktyg för att snabbt kunna anpassa befintliga processer till de nya kraven. En möjlighet är att deras IT-lösning är byggd på Microsofts .NET-plattform. Med Windows Presentation Foundation, hädanefter (WPF) och Windows Workflow Foundation, hädanefter (WF), får man robusta och väletablerade verktyg. Tillsammans med Visual studio 2008 har man bra möjligheter för att tackla problemet. Genom att använda dessa teknologier och verktyg, går det att skapa ett verktyg i form av en applikation som kan ändra sådana processer.

De workflows som går att skapa och ändra i den applikation som ska utvecklas, används sedan i företagets IT-infrastruktur. Ett workflow är ett arbetsflöde som innehåller sammanlänkade aktiviteter som bildar ett program. Dessa aktiviteter kan uträttas av både datorn och människan. Har företaget en sådan IT-infrastruktur med ett system som kan läsa workflows, då kan systemet själv generera ett e-postkonto när beställaren har lagt sin order. På så sätt effektiviserar man IT-avdelningens arbete och kan nu arbeta med service av systemet och upprätthålla det, utan att behöva avsätta resurser, till ofta förekommande uppgifter, så som att skapa ett e-postkonto till en nyanställd.

1.2 Problemområde

En utmaning hos många företags IT-avdelningar, är att snabbt kunna bistå med stöd till sin egen verksamhet. Stödet kan vara allt från att skapa nya användare i ett datorsystem, till att installera önskade applikationer på en specifik dator. De flesta uppgifter som IT-avdelningen utför följer en definierad process. I fallet ny användare skulle den kunna se ut så här:

1. En chef beställer ett nytt konto till sin nyanställde, IT-avdelningen underrättas.
2. Kontot skapas i en katalogtjänst (ett slags uppslagsverk i form av en databas).
3. En mailbox för kontot skapas i företagets mailsystem.
4. En dator beställs och placeras ut på den nyanställdes plats.
5. Ett mail skickas till chefen, där det står att allt är klart.

Om verksamheten beslutar sig för att outsourcea sin mailhantering eller byta katalogtjänst, måste processen ändras för att följa de nya kraven.

Idag måste IT-avdelningen göra den ändringen av processen med hjälp av avancerade verktyg. Visual studio är ett sådant, men är dyrt när det gäller licenser. Målet med detta arbete är att verksamheten, med hjälp av ett nytt verktyg, själva ska kunna ändra på processen utan att behöva investera i dyra licenser.

1.3 Företaget ATEA Sverige AB/ Spintop Netsolution AB

ATEA Sverige AB, är det största företaget i Sverige inom IT-infrastruktur. Företaget bildades 2007, då Topnordic och Ementor gick ihop. Sedan dess har företaget blivit större och större genom strategiska uppköp av konkurrenter. Ett av de senaste uppköpen är mjukvaruutvecklingsföretaget Spintop Netsolution AB. Spintop utvecklar och distribuerar IT-lösningar i form av mjukvara till företag. Genom detta uppköp kommer ATEA Sverige att kunna erbjuda helhetslösningar i form av både hårdvara, mjukvara och tjänster till kunder. Spintop är idag en avdelning inom ATEA Sverige, med sitt huvudsäte i Göteborg, med ett 30-tal konsulter. ATEA/Spintop är även partner till Microsoft Corporation och utvecklar all mjukvara med hjälp av Microsoftverktyg och ramverket .NET.

ATEA Sverige hjälper företag att effektivisera sina verksamhetsprocesser genom att automatisera manuella processer. Man kan beskriva ATEA som IT-avdelningen till företagens IT-avdelningar. ATEAs klientplattform är byggd på ramverket .NET och en av ATEAs flera produkter, Accelerator, som kan utnyttja plattformen har möjlighet att använda sig av Workflows. Accelerator

är ett processverktyg, som automatiskt utför alla arbetsuppgifter som behöver göras i en infrastruktur. Exempel på några områden kan vara katalogtjänst, e-post-hantering och debiteringssystem.

1.4 Problemställning

Hur kan man underlätta för IT-avdelningen att ändra befintliga verksamhetsprocesser med hjälp av Windows Workflow Foundation?

Ihop med sin programvara Accelerator, vill ATEA kunna erbjuda sina kunder att själva ändra sina processer utan att de ska behöva köpa in dyra licenser. För kundföretaget kommer applikationen att underlätta deras arbete med att ändra deras processer. Genom att kunna erbjuda en applikation som kan ändra processer, kan ATEA Sverige AB behålla företaget som kund även i framtiden.

1.5 Introduktion till .NET Framework, WF, Windows Forms och WPF

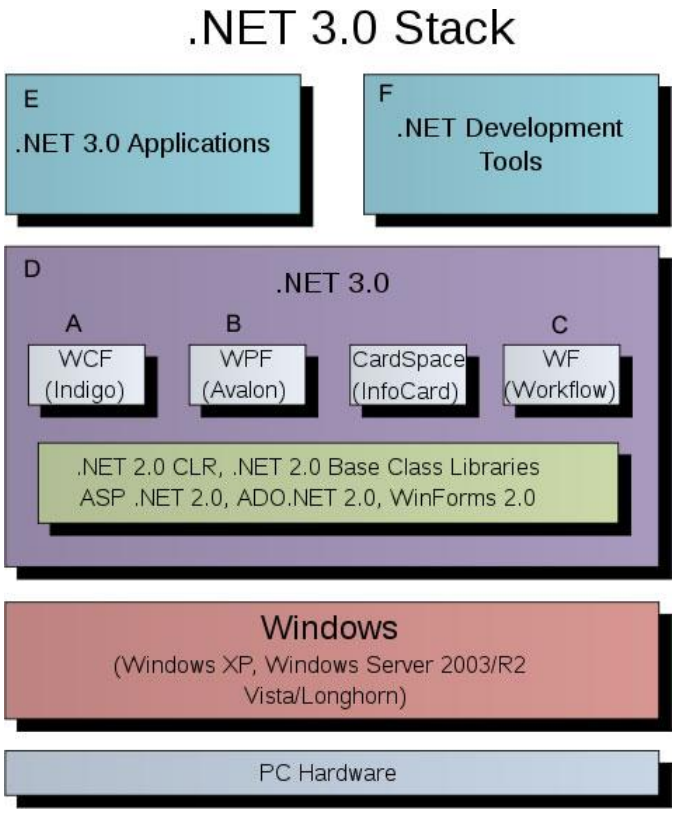
Verktyget Visual Studio och teknologierna WF, WPF och Windows Forms använder sig av .NET. .NET Framework[1] är en systemkomponent som ingår i operativsystemet Microsoft Windows. Den består av ett antal komponenter som hanterar exekvering av program. .NET består dessutom av ett stort klass-bibliotek, som innehåller förkodade lösningar för vanligt förekommande programmeringsuppgifter. .NET Framework är Microsofts svar på Suns Java-plattform.
Fördelen med .NET är att man kan använda de olika programmeringsspråk som stödjer detta. De program som skapas på plattformen, kan med hjälp av inbyggd service kommunicera med varandra.

Många av de programvaror som används av företag, har som gemensamt mål att stödja affärsprocesser. Vissa av dessa är automatiserade och förlitar sig enbart på kommunikation mellan olika applikationer, medan andra måste skötas manuellt.
I båda fallen följer dessa ett visst antal åtgärder, kända som ett arbetsflöde (Workflow), vilka beskriver en viss affärsprocess. Windows Workflow Foundation är en teknologi som ingår i .NET Framework och används för att skapa sådana workflows[2].

Windows Forms ingår, även den, i .NET Framework och är den grafiska yta som syns mot användaren, d v s knappar och fönster. Windows Forms, såg dagens ljus i samband med att .NET 2.0 släpptes. När Microsoft släppte .NET 3.0, kom det nyare gränssnittet för grafisk presentation, WPF[1].

I dagsläget finns både Windows Forms och WPF som grafiska gränssnitt i .NET. Tanken är att WPF ska ersätta Windows Forms, men Microsoft vet inte i dagsläget om WPF kommer att bli mer användbart än Windows Forms.

All kod som skrivs, ligger separerat ifrån det grafiska i alla teknologierna. På så sätt kan man bygga program som använder olika teknologier samtidigt på plattformen. Den skarpa kopplingen är att ett och samma programmerings-språk kan användas för alla teknologier.



Figur 1: Visar .NET 3.0 [1].

Här är en förenklad bild som visa hur teknologierna hänger ihop med övriga program och hårdvara.

A. Windows Communication Foundation (hädanefter WCF) kan exekvera WF.
B. WPF är det nya grafiska gränssnitt som program av typerna WCF och WF kan använda.
C. WF skapar Workflows.
D. .NET är den plattform som gör det möjligt för teknologierna att kommunicera.
E. Den applikation som utvecklas befinner sig här.
F. Visual Studio 2008 befinner sig här.

1.6 Begränsningar

Examensarbetet är uppdelat i tre steg. En första del som är huvuduppgiften och två andra som är uppgifter i mån av tid. De tre delarna bygger på varandra.

1. Skapa en applikation där man kan skapa och ändra befintliga WF-processer.
2. Applikationen ska vara utbyggbar så att fler moduler kan laddas. Detta för att kunna utöka funktionaliteten när nya behov uppkommer. Till exempel en modul för att skapa/ändra WF-process och en annan för behörighetsstyrning i applikationen.
3. Rollbaserade behörigheter ska appliceras på applikationen. Det vill säga, man ska ange den roll som ska utföra operation i applikationen. En roll kan vara t ex administratör. Dessa roller kan användare sedermera vara medlem i.

Det som ligger utanför examensarbetet är, att applikationen ska vara utbyggbar och att rollbaserade behörigheter appliceras på applikationen.

1.7 Förutsättningar

Alla nödvändiga utvecklingsverktyg kommer ATEA Sverige AB att bistå med. Arbetet kommer att ske i Visual Studio 2008, programmeringsspråket är C# och gränssnittet ska göras med hjälp av Windows Presentation Foundation.

2 Workflow Foundation

2.1 Introduktion

Windows Workflow Foundation såg dagens ljus 2006, redan då som en komponent i .NET Framework 3.0. Workflows är som vanliga applikationer med några stora skillnader[2]:

- Kan exekvera under lång tid och även ligga i vänteläge tills det dyker upp något att göra. Den har hela tiden kontakt med databasen.
- En instans av workflow kan bli redigerat dynamiskt under pågående exekvering.
- Ett grafiskt programmeringssätt och har redan har förprogrammerade delar som länkas samman, för att skapa programmet. Således behövs inga kunskaper i programeringsspråk.

2.2 Workflows och hosting

Det finns tre typer av workflows: sekventiella, tillstånds- och regelbaserade. Sekventiella workflows är typiska flödesscheman som går från en nivå till en annan, utan att gå tillbaka.
Tillståndsbaserade workflows har fördelen att de går från ett tillstånd till ett annat och har möjligheten att gå tillbaka om det krävs.
Regelbaserade workflows kan vara av de ovanstående typerna sekventiella eller tillståndsworkflows. Det är reglerna som gör att processen i workflows går framåt.
Dessa tre olika workflows kommer att behandlas lika i den utvecklade applikationen.
Workflows byggs upp av aktiviteter, antingen genom de som finns som standard från början eller egna programmerade aktiviteter. Det finns 39 olika aktiviteter för workflows som standard i Workflow Foundation (Se Bilaga)[3].

Hosting innebär att en applikation plockar in en annan applikation eller komponent och blir då värd. Genom hosting kan teknologier som inte kommunicerar med varandra integreras genom ett fördefinierat sätt. Ett exempel är en virtuell dator i datorn [4].

3 Windows Forms

3.1 Introduktion

Windows Forms är det grafiska gränssnitt som finns inbyggt i operativsystemet Microsoft Windows. De flesta knappar och fönsterhanterare är byggt ur teknologin Windows Forms. I teknologin finns redan färdigbyggda klasser för grafisk presentation. Den stora nackdelen är att Windows Forms är begränsad i den grafiska presentationen gentemot WPF.

3.2 User Control

User Control är en skapad komponent i Visual Studio, som sedan kan användas i andra applikationer, även de är skapade i Visual Studio. Kontrollen bygger på någon av de båda teknologierna Windows Forms eller WPF. För att man ska kunna använda en User Control måste den ha en värdapplikation. Att kunna skapa User Controls i Windows Forms och WPF, gör det mer användbart att använda och integrera de båda teknologierna[10].

4 Windows Presentation Foundation

4.1 Introduktion

WPF är grafiskt mycket starkare än Windows Forms som gränssnitt. Den bygger på programmeringsspråket XAML och blir på så sätt mer modifierbart än Windows Forms. WPF förenar visningen av grafik med mer avancerad grafik. Stödjer bl a 3D, vektorgrafik, animationer och rörlig grafik som film[11].

4.2 Interaktion med Windows Forms User Control

Genom att skapa en User Control i Windows Forms kan man använda re-hosting[4], för att sedan importera den i WPF. Detta sker med referenser och XAML-kod. User Controlen blir som en modul utan gränssnitt. Den kan sedan användas i en värdapplikation vid re-hosting.

Det finns olika programmeringsproblem som löses lättare i de olika teknologierna WPF och Windows Forms, jämfört med andra teknologier. Men har man då bestämt sig för att använda ett visst gränssnitt, är det alltid bra att kunna använda re-hosting för moduler/komponenter som är skapade i andra teknologier[11].

5 Utvecklingen av programmet

5.1 Introduktion

Detta kapitel beskriver den valda metoden och hur applikationen har utvecklats samt vilka krav som ställts. Även de problem och svårigheter tas upp här, som uppkommit under utvecklingens gång av applikationen.

5.2 Metod

Som ett första steg i examensarbetet är att göra förstudier om teknologierna WF, Windows Forms och WPF. Detta för att på något sätt finna samband mellan dem och hur de kan kommunicera med varandra. Efter att detta är gjort, blir det att undersöka liknande lösningar som finns på marknaden och som berör problemområdet. Den stora biten i förstudierna blir intervjun med beställaren. Denna intervju kommer att ligga till grund för kravspecifikationen. Vid utvecklandet av applikationen kommer en processmodell att användas. Den tänkta processmodellen är Scrum, som beställaren själv arbetar utifrån när det gäller produktutveckling. Sedan kommer programmeringen, där varje funktion utvecklas var för sig, för att sedan sammanföras i applikationen.

5.3 Förstudie

Det första steget i förstudien var två djupare intervjuer med ATEA. Dessa intervjuer har bidragit mycket till kravspecifikationen som nämns i nästa kapitel. De har bidragit med hur applikationen ska se ut och vilka funktioner som skulle finnas med. En av de viktigaste delarna av utvecklingen till denna applikation, har varit att samla in så mycket information om hur de tre olika teknologierna fungerar samt deras möjligheter till integrering med varandra. Detta gjordes genom att leta information på Microsofts utvecklingssite [8]. En annan del har varit att ta reda på hur workflows är uppbyggda och att även förstå den nya syntaxen när det gäller C# och XAML. Dessutom undersökt liknade lösningar som finns på marknaden, för att finna det bästa sättet att kunna lösa uppgiften. Den första lösningen som undersöktes finns på Microsofts utvecklingssite [8] och den andra i boken Pro WF – Windows Workflow in .NET 3.5, Bukovics, Bruce[5]. Undersökningen gick till på det sätt, att lösningarna bröts ner i kodavsitt för att kunna förstå lösningarna.

5.4 Kravspecifikation

De krav som ställdes var få till en början, för att sedan bli flera, såväl synliga och dolda. De krav som tas upp i denna rapport bygger både på förstudier av teknologierna, lösningarna som finns på marknaden och på den intervju som gjordes med ATEA samt de som uppkommit under utvecklingens gång.
Vid intervjuerna bestämdes vilka krav som ställdes på applikationen och vad den skulle användas till och vem som ska använda den samt dess funktioner.

ATEA ställde dessa krav på funktioner i applikationen

- Skapa nytt workflow
- Redigera befintliga workflow
- Spara befintliga workflow
- Lägga till egna aktiviteter i form av referenser (aktiviteter som är sparade i Dll-filer)

Detta är synliga krav, då de kan spåras direkt i applikationen. Det fanns även en del dolda krav, som uppkommit under utvecklingen av applikationen. Dessa krav har uppstått då man använder applikationen på ett felaktigt sätt. Några exempel på dolda krav är: Popup-ruta om sparnings-alternativ ska visas, felmeddelande ska visas om aktiviteter inte finns i workflowet och felmeddelande ska visas om man inte valt vilken typ av workflow man vill skapa.
De som ska använda applikationen är oftast de människor som sitter på IT-avdelningen och inte innehar kunskaper i Microsofts teknologier och programmeringsspråk. Därför måste denna applikation vara så enkelt uppbyggd att användaren kan direkt förstå hur applikationen fungerar.

Applikationen ska underlätta för IT-avdelningen att ändra sina verksamhetsprocesser på ett smidigt sätt genom workflower. Detta ska ske på ett enkelt och kostnadseffektivt sätt, som innebär då att IT-avdelning kan göra dessa ändringar utan att köpa in dyra licenser.

5.4.1 Allmänt

1: Applikationen ska köras lokalt på datorn.
2: Gränssnittet skall vara i WPF.
3: Programmeringsspråket skall vara C#.
4: Applikationen ska kunna köras på operativsystemet Microsoft Windows Vista eller senare.

5.4.2 Delen New

1: Kunna välja typ av workflow (sekvens eller tillstånd) samt namnge sitt workflow.
2: Spara kommer att visas på skärmen direkt efter att man skapat ett nytt workflow.
3: Knappen ”Cancel”, sparar inte workflowet.
4: Vid klick på ”Cancel” återgår man till workflowet
5: Varningsmeddelande visas om man angett ett befintligt namn.
6: Felmeddelande visas om man inte valt typ.
7: Felmeddelande skall visas om namn på workflowet saknas.

5.4.3 Delen Open

1: Man ska kunna välja vilken fil man vill öppna.
2: De filer som har ändelse XOML skall vara förvalt.
3: Felmeddelande visas om det blir fel.
4: Felmeddelande visas om det fattas referenser till aktiviteter.
5: Om referenser saknas, skall en ruta fråga efter referenser.

5.4.4 Delen Save

1: Sparar det befintliga workflowet med namn som angavs vid skapandet.
2: Öppnar man ett befintligt workflow under pågående redigering av ett annat, ska dialogruta om att spara visas innan.
3: Skapar man ett nytt workflow under pågående redigering av ett annat ska dialogruta om att spara visas.

5.4.5 Delen Save as

1: En dialogruta ska visas.
2: Filändelsen XOML skall var förvalt.

5.4.6 Delen Reference

1: En ruta med de referenser som redan finns visas.
2: Knappen Add reference visas.
3: En fildialogruta kommer upp.
4: Efter vald fil skall toolboxen uppdatera och den nya aktiviteten visas i toolboxen.

5.4.7 Delen Close

1: Vid avslut av programmet ska dialogruta visas med alternativ om att spara eller inte.
2. Har man sparat kommer inte dialogrutan upp om spara innan avslutning av programmet.

5.4.8 Kapacitets- och korrekthetskrav

1: Endast ett aktivt workflow åt gången.
2: Vid klick på specifik knapp ska programmet utföra uppgift som överensstämmer med klickad knapp.
3: Efter klick på knapp ska uppgiften kopplad till knappen utföras av programmet inom en sekund.

5.4.9 Användbarhetskrav

1: 4 av 5 användare som inte har läst manualen, ska kunna skapa ett nytt workflow och börja arbeta med det inom 5 minuter.
2: Oavsett datorvana ska 7 av 10 användare som läst manualen kunna använda samtliga funktioner i programmet.

5.4.10 Leveranskrav

1: Leveransen sker till beställare ATEA Sverige AB/ Spintop Netsolution AB
2: Leveransen sker i juni på ett datum som bestäms tillsammans med beställaren.
3: Vid leveransen sker acceptanstest mot beställaren.
4: Leveransen innehåller applikationen och en manual som beskriver hur man använder sig av applikationen.
5: Manualen kommer att vara skriven på svenska.
6: Manualen kommer att levereras i pappersformat och som digital fil i PDF-format.

5.5 Jämförbara Lösningar

Det finns redan två olika lösningar till den applikation som skall utvecklas, en som är skapad av Microsoft[12] och den andra finns beskriven i Bruce Bukovics bok: Pro WF – Windows Workflow in .NET 3.5, [5].

Lösning 1: Microsoft har skapat sin lösning genom att använda sig av grafiska gränssnittet Windows Forms. Deras lösning är begränsad i att det går lägga till enbart kodaktiviteter, detta görs genom att skriva in namnet på kodaktiviteten. En aktivitet kan vara exempelvis skapa e-post konto. Funktionerna i övrigt är att man kan skapa, ändra och spara sitt workflow. En bra lösning för dem som vill kunna lägga till enbart aktiviteter och inga andra logiska funktioner i form av if-satser eller liknade[12].

Lösning 2: Där har författaren Bruce Bukovics gjort en komplett lösning med alla standard funktioner som ingår (se appendix 1)[3] i Workflow Foundation.

Även drag och släppteknik är implementerad. Men även denna lösning bygger på det grafiska gränssnittet Windows Forms. Denna lösning är en mycket bra lösning av problemet med hosting med workflow designer utanför Visual Studio. Lösningen är bra, för att den visar att det går att skapa en applikation som är fristående från Visual Studio[5].

Ska applikationen utvecklas ifrån grunden eller ska en befintlig lösning anpassas? Att skapa applikationen från grunden innebär bättre anpassningsmöjligheter och den följer de funktionellakrav som ställts. Dessutom kan man bestämma vilka funktioner som skall implementeras och hur de ska implementeras. Genom att skapa en applikation på detta sätt, innebär mycket investering i tid. I den slutliga koden kan det finnas fel och dessa brukar vara tidskrävande att rätta till.
En redan befintlig lösning blir inte så anpassningsbar och kanske inte följer de funktionella krav som ställts. Men fördelen är att det tar mindre tid att anpassa än att utveckla.

Dessa två lösningar tar upp samma problematik, men de är begränsade och de funktioner som efterfrågas uppfylls inte i någon av dem. Lösningar handlar om re-hosting[4]. De båda lösningarna kan inte läsa in egenskapade aktiviteter i form av referenser. Detta är ett viktigt krav då företag oftast har specifika flöden för verksamheten. Ett annat problem som finns är att WF inte kan integreras i WPF i dagsläget, men integreringen mellan WF och Windows Forms finns däremot. Med hjälp av Bruce Bukovics lösning[5], så kan man återskapa workflow designer till en User Control av typen Windows Forms. Workflow designer är en byggd komponent som besår av en toolbox, en egenskapslista och en designyta för ett workflow. Bruce Bukovic beskriver även hur man skapar en applikation som blir värd till workflow designer i Windows Forms. Genom att bygga om denna med den kod som återfinns i Bruce Bukovics lösning, kan man istället använda sig av WPF. Tack vare att både Windows Forms och WPF är grafiska gränssnitt, kan de integreras med varandra. Att re-hosta en User Control av typen Windows Forms fungerar utmärkt i WPF. Re-hosting innebär att man återanvänder en komponent som är av en redan fördefinierad typ, i detta fall Windows Forms i ett annat gränssnitt, som WPF. Valet att bygga vidare på en färdig applikation, anses vara den bästa och mest tidsbesparande.

5.6 Genomförande

Den valda processmodellen blev en tillämpning av den agila systemutvecklingsprocessen Scrum. Produktägaren är ATEA Sverige AB och processledare samt utvecklare är författaren av detta dokument. I denna tillämpade versionen av Scrum så är varje funktion en sprint i utvecklingen av applikationen.

Nedan följer en kort beskrivning hur Scrum fungerar.

Scrum är en projektstyrningsmodell som är lämplig när det gäller program-varuutveckling och introducerades av Ken Schwaber och Jeff Sutherland[6].

Tillverkningsarbetet i projektet är det stora i Scrum och börjar därför med kravbilden, en s k productbacklog. I denna samlas alla de krav som ställs, där de även får prioritering efter hur viktiga de är för applikationen. Tillverkningen delas sedan in i iterationer, sk sprintar. En sprint lämpar sig i detta fall för en funktion och varar ca 15-30 dagar beroende på komplexitet för funktionen. När man planerar en sprint bryter man ner uppgiften till mindre beståndsdelar (tasks). Resultatet av planeringen blir sprintens aktivitetsplan (sprintbacklog). Under sprinten hålls dagligen ett möte (daily scrums) där varje projektmedlem besvarar dessa tre frågor[7]:

- Vad har jag gjort sedan igår?
- Vad ska jag åstadkomma till i morgon?
- Vad hindrar mig?

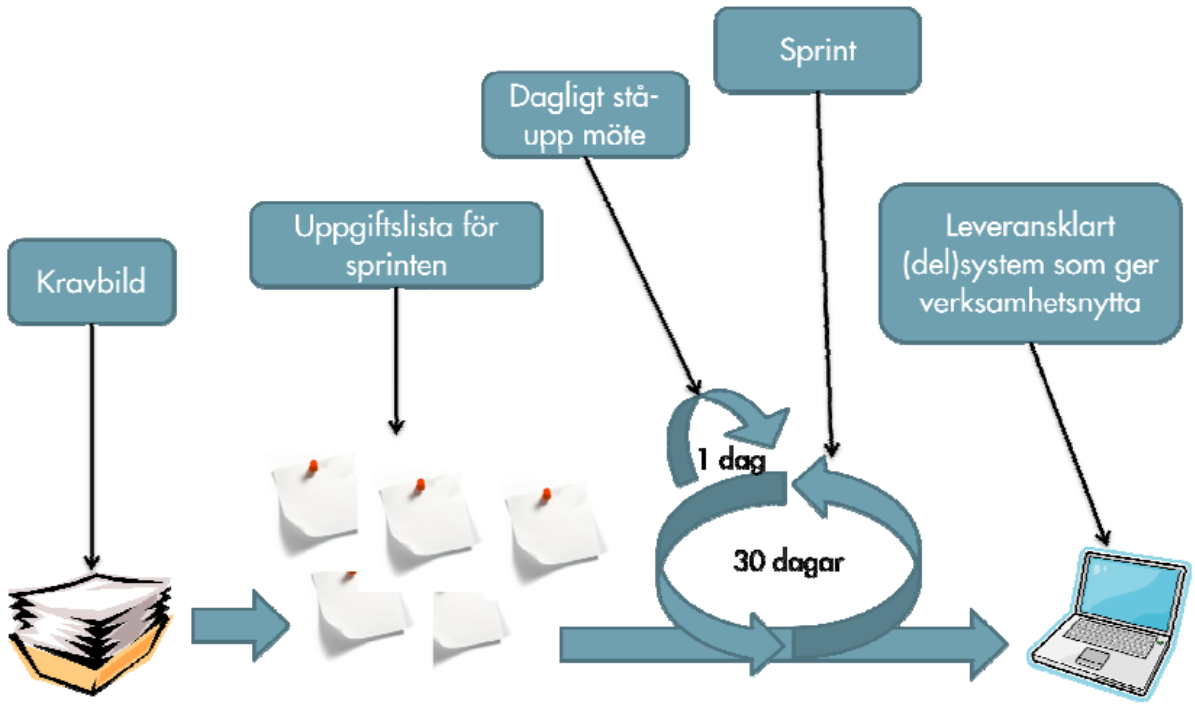
I slutet av varje sprint diskuteras vad som gått bra och vad som gått mindre bra samt hur man ska arbeta vidare med nästa sprint.

Inom Scrum finns olika roller som alla är beroende av projektets resultat. Dessa är produktägare, processledare och gruppmedlem.

Produktägare (Product Owner) – Projektgruppens motpart. Ansvarar för kravbilden samt hanterar och prioriterar önskemål/krav på produkten.

Processledare (ScrumMaster) – Produktägarens motpart. Säkerställer efterlevnad av processen, synkroniserar aktörer samt avlägsnar hinder för utvecklargruppen. Kan liknas vid en projektledare men är i själva verket inte det då projektgruppen är självorganiserande.

Projektgruppen (Team) – Utvecklargruppen är självorganiserad. Det finns inga roller, utan medlemmarna bestämmer allt eftersom, vem som utvecklar vad. Det är därför det inte finns någon projektledare. Gruppen bör bestå av 5-9 personer.



Figur 2: Scrum [7].

5.7 Lösning och problem

Det första steget var att studera teknologierna och ta fram en kravspecifikation. När detta var gjort kunde programmeringsarbetet komma igång. Som ett första led i framtagandet av applikationens funktioner, byggdes de klassfiler som skulle behövas för funktionerna. Vid detta stadium märktes det redan hur svårt och komplicerat teknologierna hänger samman. Då undersöktes andra lösningar som belyst samma problematik och hur de lösningarna är uppbyggda. Dessa lösningar återfinns i kapitlet ”Jämförbara Lösningar”. Vid genomgången av de två lösningarna och deras kod hittades både för- och nackdelar med lösningarna, ett exempel är att endast kodaktiviteter kan läggas till. Det beslutades att återanvända den kod som finns i boken Pro WF – Windows Workflow in .NET 3.5, Bukovics, Bruce [5], därför att det finns en grundlig genomgång av koden och hur den är skriven. Tyvärr inte så grundlig att det går att skriva om kärnan till applikationen som tas upp i boken. Denna blev lite av en guide till hela området med teknologier och kod. Genom att återskapa denna kod till ett fungerande program kunde man få vetskap om hur en lösning kan se ut. Denna lösning använde sig av gränssnittet Windows Forms. Kunde man på något sätt byta ut detta gränssnitt till WPF?

Att skapa WPF fönster och menyer var inte svårt, men att få WF att fungera med WPF ställde till mer bekymmer än räknat. Efter mycket läsande på Microsofts utvecklingssite [8] för programmerare, blev svaret att detta inte fungerar i nuläget. WPF är så nytt att det inte har anpassats för WF. Hur gör

man då? Vid vidare läsning på Microsofts utvecklingssite fanns en artikel om hur re-hosting går till mellan Windows Forms och WPF. Genom att skapa något i ett gränssnitt till en s k User Control, kan denna sedan återanvändas i ett annat gränssnitt.

Det första steget blev att undersöka ett tvådelat exempel som behandlade datum. Det intressanta med detta exempel var att den tog upp User Control och hur denna sedan hämtades in i ett fönster av typen Windows Forms. Genom att bygga ihop de kodavsnitt som gavs, kunde två olika projekt byggas. Det ena projektet behandlade fönstret och det andra User Controlen. Med den nyvunna kunskapen om User Control började försöken med att separera workflow designer ifrån Windows Forms för att sedan göras exakt som de visat i exemplet.

Först gjordes ett nytt projekt av typen Windows Forms Control Library med att skapa User Control för komponenten workflowdesignern. Genom att separera workflowdesignern ifrån Windows Forms i det exempel som finns i boken kunde workflowdesignern byggas till en User Control av typen Windows Forms[5]. Det som är viktigt här var att de klasser som arbetar med själva workflowet måste följa med till projektet för att det sedan skulle fungera att bygga det

I den andra delen av projektet visades den kod som importerade User Controlen i det exempel som behandlade datum. Genom att skapa ett till nytt projekt av typen WPF Application kunde sedan kod återanvändas. Det gick att importera User Controlen med XAML-kod till WPF Application projektet. Genom att återanvända den kod i Bruce Bukovics [5] lösning som sköter hela workflow-designern, d v s det som ritar upp workflow, toolbox och egenskapslistan så behövdes det bara läggas till kod för att metoder skulle hittas och för att få workflow designern att fungera. Så långt fanns ett program som just nu inte kunde göra något. Men det fanns ett grafiskt gränssnitt och en designyta i form av workflowdesignern. Men för att få gränssnittet att kommunicera med work-flowdesignern krävdes det programmering och kunskap om hur workflow laddas in i workflowdesignern.

För att få programmet att fungera fullt ut, krävdes en meny som kunde direkt-kommunicera med workflowdesignern genom WPF. Även här söktes det efter ett exempel på hur man kunde bygga menyer i WPF. Sidan c-sharpcorner [9] visade en mycket användbar guide på hur koden skulle skrivas för att skapa en meny i XAML-kod. Men det räcker inte med en meny som bara syns, den måste ju utföra något när man exempelvis trycker på ”New”. I boken fanns redan kod för de flesta funktionerna som skulle implementeras. Funktionerna delades in i sprintar enligt Scrum och implementerades en och en. Efter att ha

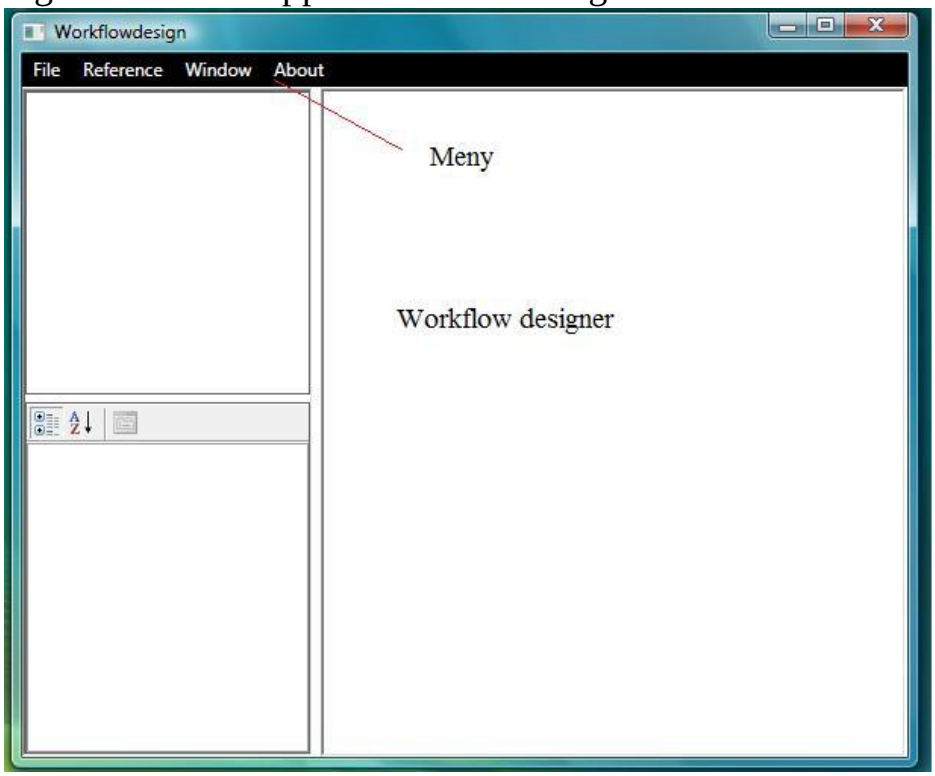
implementerat funktionerna (New, Open, Save, Save as, Close), fanns det nu ett fungerande program som kunde användas till en viss gräns.

Det som ställde till problem var, vad händer om man glömmer att spara? Då gick allt arbete med workflowet förlorat. Här måste som sagt komma in en förfrågan till användaren om att spara, innan användaren väljer att skapa ett nytt workflow eller öppna workflow eller att stänga av programmet. De lösningar som skulle kunna fungera var att låta själva programmet spara så fort en ändring gjorts i workflowet eller ställa en förfrågan vid val av annat menyval som berör workflowet. Det första alternativet skulle vara bra, men då skulle sparfunktionen vara i gång hela tiden och ta mycket kraft för programmet. Därför blev det alternativ två som ansågs vara det bättre alternativet för denna applikation. Genom att studera, ett exempel om hur man skapar dialogrutor kunde detta implementeras överallt i koden där det skulle behövas. Denna säkerhetsåtgärd är att workflowets ändringar inte går förlorade av misstag.

Ett annat problem som nu kom upp till ytan är aktiviteterna i workflowet. Det är inte alltid man använder sig av standardaktiviteter, utan har skapat egna aktiviteter som ska kunna användas. Menyn måste kompletteras med ett menyalternativ för att lägga till egna aktiviteter. Även denna gång var det Bruce Bukovics bok som hade ett kapitel just om detta[5]. Menyn kompletterades med Reference som lägger till egna aktiviteter genom DLL-filer. DLL står för Dynamic Link Library och är filer som innehåller funktioner som program använder. Men i detta fall innehåller DLL-filerna aktiviteter som bara Visual Studio och workflowdesignern kan använda sig av. DLL-filerna kopieras till mappen där kärnan för applikationen ligger och kan på så sätt användas direkt utan att starta om applikationen. Nu kan egna aktiviter läggas till och användas i workflowet.

Om användaren läser in ett workflow som innehåller egna aktiviteter som inte finns i kärnan för applikationen uppstår ett allvarligt fel. För att lösa det bestämdes att en sk try-catchfunktion skulle implementeras. Denna funktion i koden, ska tala om för användaren vilka referenser som saknas och visar då upp en dialogbox, där användaren kan lägga till referenser i efterhand för att försöka öppna workflowet. Detta gjordes men då uppstod följdproblem som inte lyckades lösas på ett bra sätt. Lösningen på problemet blev att användaren får reda på vilka referenser som saknas och ska läggas till innan öppnandet av workflowet. Applikationen startas då om och användaren får chansen att lägga till referenser innan öppnande av befintligt workflow görs igen.

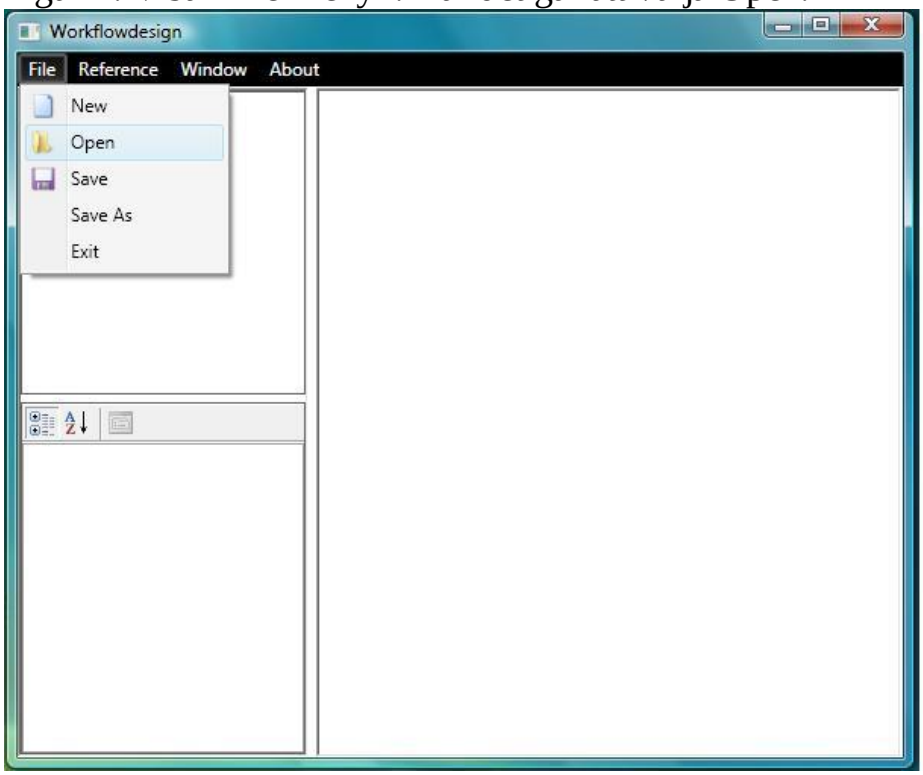
Figur 3: Visar en applikationen utan något workflow laddat.



Figur 3: Visar applikationen utan något workflow laddat.

Genom att välja ett menyalternativ kan man kommunicera med workflow-designern.

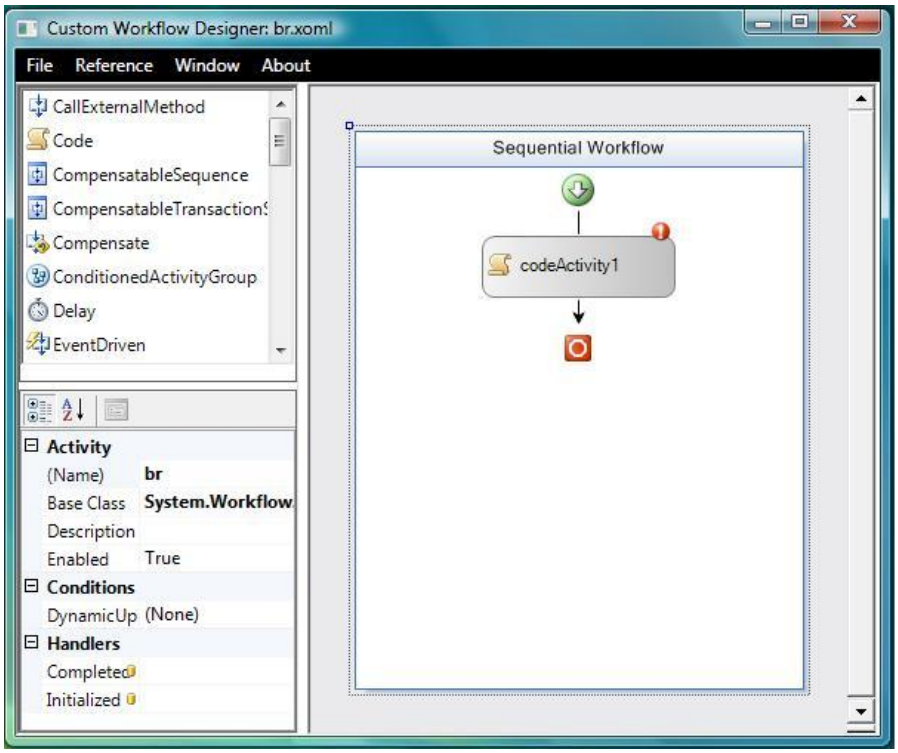
Figur 4: Visar File-menyn. Där det går att välja Open.



Figur 4: Visar File-menyn.

Vi väljer File – Open – den fil vi vill ladda och avslutar med att ladda workflowet.

Figur 5: Visar ett laddat workflow med en aktivitet. Dessutom kan man nu se att hela workflowdesignern är laddad med en toolbox och innehåller standardaktiviteterna, egenskapslista för en aktivitet och designytan för hela workflowet.



Figur 5: Visar ett laddat workflow med en aktivitet i design ytan till höger. Längst upp tillvänster syns toolboxen med standard aktiviteter och under denna egenskapslistan för en aktivitet.

Det som är bra med denna applikation, är att den enbart har funktioner för workflowhantering och ingen annan hantering. Om man jämför applikationen med det mer avancerade programvaran Visual Studio är detta ett lättare program att arbeta i, dessutom är det inte alls lika systemkrävande.

6 Utvärdering och Resultat

Så här i efterhand kan den valda lösningen diskuteras. Hade det varit andra funktioner som skulle utvecklats, hade nog den rätta metoden varit att skriva om kärnan till workflowdesigner. Valet av denna lösningsmetod är att den anses vara den bästa och mest tidsbesparande utifrån de kraven som ställdes av beställaren ATEA.

Förutom problemet med try-catchfunktionen fanns det en del större problem. Ett av dem var att WF kan endast integreras i gränssnittet Windows Forms och inte i det nya gränssnittet WPF . Det tog det tid innan en lösning på problemet kom fram genom boken [5] . Ett annat problem var med User Control och hur denna skulle skapas och implementeras i gränssnittet WPF.

Resultatet blev en applikation som kan skapa, redigera och spara workflows med ett gränssnitt gjort i WPF samt att det går att lägga till egenskapade aktiviteter ifrån DLL-filer. En fördel med Microsofts teknologier är att de använder samma programmeringsspråk och kan på så sätt kommunicera på .NET-plattformen. Användandet av User Control är ett bra sätt att få de teknologier som inte har stöd för att kan kommunicera direkt med varandra på plattformen .NET att kommunicera.

Valet av att fokusera på att skapa applikationen och inte bygga in behörighetsfunktion och utbyggbarheten visade sig vara mycket bra då tiden inte skulle räcka till för att göra detta, men det kan vara något som implementeras i framtiden. En autosavefunktion hade varit mycket välkommet inslag i applikationen, som sparar ändringar i workflowet. Denna funktion skulle nog vara mer lämpad för en större och avancerad applikation, där ändringar kanske är små och görs mycket ofta.

Hade en kodvy implementerats hade det definitiva svaret varit en autosavefunktion, som skulle implementerats för att på så sätt säkerhetsställa att inga små ändringar i koden gått förlorade. Med en kodvy skulle applikationen bli mer användbar då ändringarna av aktiviteterna skulle kunnas göras direkt i workflowet genom att kunna programmera direkt i applikationen. Detta innebär att användaren själv kan gå in i koden för en aktivitet och ändra om det skulle behövas och då även spara den som en DLL-fil för framtida användning. Det som skulle göra applikationen mer användarvänlig är små tipsbubblor med information om just den saken som muspekaren pekar på. Dessutom skulle även ett sökbart hjälpsystem varit en bra åtgärd, som användaren kunnat komma åt genom applikationen. Applikationen stödjer de krav som ställts i kravspecifikationen. Programmet kommer att bli ett mycket bra val istället för det dyrare programmet Visual Studio.

7 Slutsatser och framtida arbete

Detta projekt tar upp hur man kan ändra befintliga verksamhetsprocesser på ett enklare sätt, genom att använda sig av workflows utan att behöva köpa dyr programvara. Förstudien visade att det finns lösningar för att hantera workflows, men i dessa lösningar [4] [5] finns det inte alla funktioner som efterfrågades av ATEA Sverige AB.

Det finns möjligheter att bygga ut programmet ytterligare.

- Skapa ett mer tilltalande och mer användarvänligt gränssnitt. XAML-kod är uppbyggt för att kunna visa rikare grafik.
- Bygga om kärnan från grunden till workflowdesignern.
- Skapa en kodvy för de aktiviteter och funktioner som ingår i workflows. Det innebär att användaren kan koda sina aktiviteter direkt och på så sätt få ännu mer styrning över vad som skall göras.
- Behörighetsstyrning i applikationen. Rollbaserat så att bara vissa med särskild behörighet kan utföra en viss operation på workflowet.
- Sökbart hjälpsystem vore mycket bra att ha då det oftast är specifik hjälp användaren behöver.

8 Terminologi

Ord	Förklaring
.NET Framework	Program utvecklings plattform från Microsoft
C#	Programmeringsspråk
Hosting	Värd, Husera
Re-hosting	Återhusera
Scrum	En metodik för systemutveckling
User Control	En modul som kan skapas i både Windows Forms eller Windows Presentation Foundation
WF	Windows Workflow Foundation
Workflow	Flödesschema
Workflowdesigner	Den yta där workflow redigeras och visas
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XOML	Flödesschemats filformat

9 Referenser

1. http://www.microsoft.com/net/
   (2008) NET Framework
2. http://msdn.microsoft.com/en-us/netframework/aa663328.aspx
   (Maj 2009) Windows Workflow Foundation
3. http://msdn.microsoft.com/en-us/library/ms733615.aspx
   (2007) Workflow Activities
4. http://msdn.microsoft.com/en-us/library/aa480213.aspx
   (Maj 2006) Windows Workflow Foundation: Everything About Re-Hosting the Workflow Designer
5. Bukovics, Bruce, 2008. Pro WF – Windows Workflow in .NET 3.5. Apress. New York, ISBN10 1430209755
6. http://www.controlchaos.com/
   (2009) Scrum
7. http://www.systemvaruhuset.se/media/10727/introduktion%20till%20scrum.pdf (Feb 2009) Introduktion till Scrum
8. http://msdn.microsoft.com/sv-se/default.aspx
   (2009) Microsoft Developer Network
9. http://www.c-sharpcorner.com/UploadFile/mahesh/WPFMenu07282008145508PM/WPFMenu.aspx (Juli 2008) Menus in XAML and WPF
10. http://msdn.microsoft.com/en-us/library/bb762963.aspx
    (2009) Windows Forms
11. http://msdn.microsoft.com/en-us/library/ms754130.aspx
    (2009) Windows Presentation Foundation
12. http://msdn.microsoft.com/en-us/library/ms741708.aspx
    (2009) Basic Designer Hosting Sample

10 Bilaga

Standard aktiviteter som finns i Windows Workflow Foundation[3].

Activity Name	Description
CallExternalMethodActivity	Used with the **HandleExternalEventActivity** activity for input and output communications with a local service.
CancellationHandlerActivity	Used to contain cleanup logic for a composite activity that is canceled before all the composite activity's child activities are finished executing.
CodeActivity	Enables you to add Visual Basic or C# code to your workflow.
CompensatableSequenceActivity	Compensatable version of **SequenceActivity**.
CompensatableTransactionScopeActivity	Compensatable version of **TransactionScopeActivity**.
CompensateActivity	Enables you to call code to undo or to compensate for operations already performed by the workflow when an error occurs.
CompensationHandlerActivity	Wrapper for one or more activities that perform compensation for a completed **TransactionScopeActivity** activity.
ConditionedActivityGroup	Executes child activities based on a condition that applies to the **ConditionedActivityGroup** activity itself and based on conditions that apply separately to each child.
DelayActivity	Enables you to build delays in your workflow based on a time-out interval.
EventDrivenActivity	Wraps one or more activities that are executed when a specified event occurs.

EventHandlersActivity	Provides a framework for associating events with an activity.
EventHandlingScopeActivity	Executes its main child activity concurrently with an **EventHandlersActivity** activity.
FaultHandlerActivity	Used to handle an exception of a type that you specify.
FaultHandlersActivity	Represents a composite activity that has an ordered list of child activities of type **FaultHandlerActivity.**
HandleExternalEventActivity	Used together with the **CallExternalMethodActivity** activity for input and output communications with a local service.
IfElseActivity	Tests a condition on each branch and performs activities on the first branch for which the condition equals **true**.
IfElseBranchActivity	Represents a branch of an **IfElseActivity** activity.
InvokeWebServiceActivity	Enables your workflow to invoke a Web service.
InvokeWorkflowActivity	Enables your workflow to invoke another workflow.
ListenActivity	A composite activity that contains only **EventDrivenActivity** child activities.
ParallelActivity	Lets you schedule two or more child **SequenceActivity** activity branches for processing at the same time.
PolicyActivity	Used to represent a collection of rules. A rule consists of conditions and resulting actions.
ReceiveActivity	Implements a service contract operation using Windows Communication Foundation (WCF).
ReplicatorActivity	Creates multiple instances of a single child activity.
SendActivity	Invokes a client-side synchronous

	operation of a service using Windows Communication Foundation (WCF) .
SequenceActivity	Provides a simple way to link multiple activities together for sequential execution.
SetStateActivity	Specifies a transition to a new state.
StateActivity	Represents a state in a state machine workflow.
StateFinalizationActivity	Used in a **StateActivity** activity as a container for child activities executed when leaving the **StateActivity** activity.
StateInitializationActivity	Used in a **StateActivity** activity as a container for child activities executed when entering the **StateActivity** activity.
SuspendActivity	Suspends the operation of your workflow to enable intervention in the event of some error condition that requires special attention.
SynchronizationScopeActivity	Executes contained activities sequentially in a synchronized domain.
TerminateActivity	Enables you to immediately end the operation of your workflow in the event of an error condition.
ThrowActivity	Enables you to capture business exceptions thrown as part of the process metadata for a workflow.
TransactionScopeActivity	Provides a framework for transactions and exception handling.
WebServiceFaultActivity	Lets you model the occurrence of a Web service fault.
WebServiceInputActivity	Receives data from a Web service.
WebServiceOutputActivity	Responds to a Web service request made to a workflow.
WhileActivity	Enables your workflow to loop until a condition is met.