Institutionen för datavetenskap

Department of Computer and Information Science

Examensarbete

I Microsoft Dynamics AX – databasaccess, kommunikation och tjänster för Commerce Runtime

av

Henrik Karlsson & Mattias Olsson

LIU-IDA/LITH-EX-G--13/035—SE

2013-06-26

Linköpings universitet

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköpings universitet
Institutionen för datavetenskap

Examensarbete

I Microsoft Dynamics AX – databasaccess, kommunikation och tjänster för Commerce Runtime

av

Henrik Karlsson
Mattias Olsson

LIU-IDA/LITH-EX-G--13/035—SE

2013-06-26

Handledare: Jonas Wallgren
Examinator: Klas Arvidsson

I Microsoft Dynamics AX – databasaccess, kommunikation och tjänster för Commerce Runtime

Henrik Karlsson **Mattias Olsson**

SAMMANFATTNING

Examensarbetet undersöker den nya versionen av affärssystemet Microsoft Dynamics AX R2 2012 och dess plattform Commerce Runtime och analyserar dessa ur perspektiven arkitektur och kommunikationsmöjligheter.

Examensarbetet undersöker systemets arkitektur och implementerar tre metoder för att visa upp kommunikationen mellan komponenterna i systemet. Undersökningen avgränsar sig dock till de områden av systemet och plattformen som har med butikslösningen att göra.

Undersökningen går hand i hand med implementationen. Implementeringen sker löpande under arbetet för att säkerställa att undersökningen stämmer.

Rapporten beskriver tre olika kommunikationsmetoder. Synch Service synkroniserar Dynamics AX databas med Commerce Runtime databas med hjälp av Dynamics AX användargränssnitt. SQL-adapter används för att hämta den synkroniserade datan i Commerce Runtime databas. Real-Time Service anropar exponerade metoder i Dynamics AX från en extern klient.

Utifrån utvärderingen så lämpar sig Synch Service bra till när mycket information ska uppdateras rutinmässigt. Real-Time Service lämpar sig när data ska hanteras på ett antal bestämda sätt och då beräkningar av data ska genomföras och lämna tillbaka ett svar som genererats dynamiskt.

ABSTRACT

The thesis examines the new version of the Enterprise Resource Planning system, Microsoft Dynamics AX R2 2012, along with its platform Commerce Runtime and analyze them from the perspectives of architecture and communication possibilities.

The thesis is looking into the system architecture and implements three methods to demonstrate the communication between the system components. The study of the system architecture is limited to the areas of the system and platform that are connected to the Retail solution.

The study goes hand in hand with the implementation. The implementation is being done alongside the study to ensure its correctness.

The report describes three different methods of communication. Synch Service synchronizes the Dynamics AX database with the Commerce Runtime database. SQL-adapter is used to retrieve the synchronized data from the Commerce Runtime database. Real-Time Service calls the exposed methods in Dynamics AX from an external client.

Based on the evaluation; Synch Service is useful when a lot of data is being updated routinely. Real-Time Service is useful when data is to be handled in a number of specific ways and when calculations of data are needed to return a dynamically generated response.

INLEDNING

Microsoft Dynamics AX är ett av Microsofts affärssystem[1]. Systemet erbjuder kunder till Microsoft en rad olika lösningar: redovisning, kundrelationshantering, leverantörshantering, produktionshantering, personal-hantering, projekthantering[2]. Den innehåller även en lösning som kallas butik[3], vilket är den mest centrala lösningen för detta examensarbete. Butikslösningen utökar Dynamics AX kärnlösning med en rad nya begrepp, såsom butik, sortiment och kampanj. Butikslösningen ger kunder möjlighet att kommunicera med externa komponenter, något som i synnerhet kännetecknar handelslösningar. Exempelvis finns en intern POS-lösning (Point Of Sale, d.v.s. kassa) och en så kallad Store Front (e-handelslösning). Både POS-lösningen och Store Front bygger på en utanpåliggande plattform, Commerce Runtime. Commerce Runtime erbjuder möjlighet att interagera både online och offline med Dynamics AX. Commerce Runtime kan exempelvis få tillgång till Dynamics AX aktuella produktsortiment, lagersaldo, eller lägga en försäljningsorder trots att systemen inte har nätverksanslutning.

I den senaste versionen av affärssystemet, Microsoft Dynamics AX 2012 R2[4], släpptes för första gången plattformen Commerce Runtime[5]. Det finns många frågor och funderingar kring dess funktionalitet och hur den kommunicerar med sina omkringliggande lösningar.

Examensarbetets uppgift

Examensarbetet syftar till att undersöka plattformen Commerce Runtime och affärssystemet Dynamics AX. Examensarbetet analyserar dessa ur perspektiven arkitektur och kommunikationsmöjligheter.

Arkitekturen beskrivs som ett resultat av den undersökning som görs under examensarbetets tid och är det första momentet i rapporten. Systemet och dess komponenter

visas upp med hjälp av bilder och beskrivningar så att läsaren får en konkret uppfattning om dess uppbyggnad. Det beskrivs även kort hur navigationen i Dynamics AX användargränssnitt går till.

Kommunikationen mellan systemet och plattformen kommer visas upp med tre olika metoder. Metoderna visar olika möjligheter i systemet, likt en manual, och ger läsaren en möjlighet att kunna återskapa metoderna.

Avgränsningar

Eftersom Dynamics AX är ett så pass stort system så måste det undersökande momentet i rapporten avgränsas. Det är butikslösningen och dess kommunikation som står i fokus i examensarbetet. Undersökningen som görs kommer därför endast att beskriva de delar av Dynamics AX som är relaterade till Commerce Runtime eller har betydelse för någon av de implementerade kommunikationsmetoderna. Examensarbetet undersöker heller ingen säkerhetsrelaterad teori kring komponenterna eller kommunikations-metoderna.

Examensarbetet syftar inte till att visa upp ett bästa-sätt att implementera kommunikationmetoder. Det ska endast ge en översikt av systemet och vara en manual för hur kommunikation kan implementeras.

Motivering av moment

Dynamics AX och dess butikslösning har ett begränsat antal sätt att kommunicera. De två huvudsakliga kommunikationssätten som systemet erbjuder är Real-Time Service och Synch Service, därför kommer dessa att beskrivas och implementeras. När Synch Service är implementerat så kommer det finnas data att arbeta med i Commerce Runtime databas. För att ha någon användning av den data som ligger i databasen så implementeras en SQL-adapter för att kunna hämta ut och modifiera datan. Det finns en inbyggd klass i Commerce Runtime som är till för att hantera sådana operationer, *CommerceEntity*-klassen. Anledningen till att en SQL-adapter valdes före CommerceEntity-klassen diskuteras under avsnittet *Diskussion*.

Moment i examensarbetet

Rapporten beskriver fyra moment:

1. En beskrivning av Dynamics AX och Commerce Runtime arkitektur.
2. En metod för hur en synkronisering mellan Dynamics AX databas och Commerce Runtime databas uppnås med Synch Service.
3. En metod för hur data hämtas ut från Commerce Runtime databas med en SQL-adapter.
4. En metod för att ropa på exponerade metoder i Dynamics AX externt med Real-Time Service.

Varför undersökningen är viktig

Den största anledningen till att examensarbetet är viktigt är för att Medius, examensarbetets uppdragsgivare, och andra Microsoft Partners kommer ha användning av det. De arbetar med att anpassa Dynamics AX-miljöer till deras kunder och har stort intresse av att få kunskap om de nya komponenterna och hur kommunikationen sker mellan dem. Den nya versionen av Dynamics AX innehåller mycket ny funktionalitet, men dokumentationen till den kan vara svårförstådd. Det gör att det blir misstag och tidsspill när utvecklare vill utöka systemet, eftersom den dokumentation som finns tillgänglig inte är tillräckligt omfattande eller tillräckligt djup i beskrivningen.

Vårt examensarbete fungerar som ett exempel på hur kommunikationen mellan Dynamics AX och Commerce Runtime kan skapas. Detta kommer användas i framtida kundcases hos Medius eftersom implementationen och undersökningen till examensarbetet är väldokumenterade. Arbetet som gjorts är en genväg till en lyckad kommunikation där läsare kan återskapa de implementerade metoderna.

BEGREPP & TERMER

Här beskrivs de begrepp och termer som läsaren behöver ha kunskap om för att förstå rapportens innehåll.

Microsoft Dynamics AX 2012 R2

I Dynamics AX finns det ett flertal olika lösningar med mycket funktionalitet. Detta är tillgängligt från Dynamics AX användargränssnitt.

Det finns även ett utvecklarläge där programmerare har möjlighet att utöka systemet med ny funktionalitet, som kunder till programmeraren sedan kommer åt i användargränssnittet. Koden i utvecklarläget i Dynamics AX skrivs i programmeringsspråket X++[1] som är ett Microsoft-utvecklat språk.

Implementationen kräver arbete både i utvecklarläget och i användargränssnittet under examensarbetet.

Commerce Runtime

Commerce Runtime är en plattform som är integrerad med Dynamics AX. Commerce Runtime är motorn i butikslösningen som kopplar ihop en webbutik med en lokal databas som är fristående från Dynamics AX. I stället för att webbutiken ska behöva kommunicera direkt med Dynamics AX (där trafiken kan vara tung då stora datamängder hanteras) så använder den en lokal, synkroniserad databas. Den lokala databasen kan synkroniseras med Dynamics AX så att all data som webbutiken behöver finns tillgänglig.

Under examensarbetet så används Commerce Runtime databas som synkroniseringsmottagare till Dynamics AX databas.

Real-Time Service
Real-Time Service är en webbtjänst som används för att hämta data direkt från Dynamics AX databas[7]. I de fall där data både lagras i Dynamics AX databas och i Commerce Runtime databas så används Real-Time Service för att kunna hämta ut den del av datan som ligger i Dynamics AX databas. För- och nackdelar om att dela upp datalagringen tas upp i avsnittet *Diskussion*.

I en av de implementerade kommunikationsmetoderna så används Real-Time Service för att ropa på en exponerad metod i Dynamics AX.

Synch Service
Synch Service ger en möjlighet att uppdatera en tabell från en databas till en annan enligt ett schemalagt förfaringssätt eller genom direktkörning[8]. Med Synch Service finns möjlighet att uppdatera stora mängder data rutinmässigt, t.ex. prisändringar på produkter.

I en av de implementerade kommunikationsmetoderna så används Synch Service för att synkronisera Dynamics AX databas med Commerce Runtime databas.

AOS
AOS står för Application Object Server och är det mittersta lagret i Dynamic AX *trelagerlösning*[9]. Det översta lagret är klientlagret och det understa är databaslagret. Trelagerlösningen är en intern uppdelning som skiljer på presentation, logik och data. AOS har alla programklasser i systemet och tar hand om all kodkörning.

Examensarbetet kommer i kontakt med alla lager i trelagerlösningen. I AOS skapas metoder, i klientlagret implementeras synkroniseringen och i databaslagret hämtas data från Dynamics AX.

Microsoft Visual Studio
Microsoft Visual Studio är en avancerad programutvecklingsmiljö från Microsoft. I miljön utvecklas bland annat applikationer för Windows.

Denna miljö används vid implementationen av Real-Time Service och då data hämtades från Commerce Runtime databas.

Microsoft SQL Server Management Studio
Microsoft SQL Server Management Studio används för att hämta, konfigurera, hantera, administrera och utveckla komponenter i en SQL-Server.

I examensarbetet används den för att hantera Commerce Runtime databas.

Internet Information Service 8
IIS8 för Windows Server är ett gränssnitt som hanterar Internet Information Service. IIS är en serverprogramvara från Microsoft som används för internetbaserade tjänster.

Gränssnittet IIS8 används då Real-Time Service implementeras. World Wide Web-Service måste startas om med hjälp av IIS8 för att ändringar i koden till Real-Time Service ska uppdateras.

Sharepoint
Sharepoint är en gränssnitt som är skapat av Microsoft och kan användas till att skapa webbutiker. Det är ett gränssnitt som är tänkt att användas av personal, såsom chefer och butiksanställda i en affär.

En webbutik skapad i Sharepoint kan använda sig av Commerce Runtime databas för att lagra och hämta data.

Företag (som begrepp i Dynamics AX)
Ett företag i Dynamics AX utvecklarläge syftar till den miljö med sin unika uppsättning data som finns tillgänglig för varje enskilt företag. Om en tabell är skapad i företaget X miljö så finns den inte tillgänglig för företag Y (om inte tabellen görs tillgänglig till alla företag explicit).

Företaget som används under examensarbetet är *usrt*. Det är viktigt att företaget används konsekvent så att alla data finns samlad på samma ställe.

METOD

Undersökning
Den största delen av examensarbetet handlar om att förstå strukturen och kopplingen mellan Dynamics AX och Commerce Runtime. Därför krävs det undersökningsarbete genom att läsa relevant information. Det görs genom att leta i Microsoft technet-dokumentation och att läsa i manualen till 2012-års upplaga av Microsoft Dynamics AX.

I takt med att systemet och plattformen undersöks så skapas kommunikationsmetoder i de områden som implementation är möjlig. Det görs för att säkerställa att teorin i undersökningen stämmer.

Implementation
Parallellt med undersökningen skapas prototyper på kommunikationsmetoder. För att implementera prototyper så följs dokumentationen från technet-sidor. Vid implementationen av Synch Service utgick dokumentationen från technets *Configure and schedule retail data distribution [AX 2012]*[23]. Där finns en del information om hur Synch Service implementeras och länkar för att få mer information om tjänsten. För att implementera Real-Time Service utgick dokumentationen från *Set up a Real-time Service profile [AX 2012]*[24]. Precis som med dokumentationen av Synch Service så finns det även här länkar som kan följas för att få mer information om den beskrivna tjänsten.

Under implementationen och undersökningen så används olika program och utvecklingsmiljöer. Dessa program och utvecklingsmiljöer väljs på grund av att dokumentationen förespråkar dessa, eller för att det inte finns några alternativ.

För att lagra data i Dynamics AX databas används Dynamics AX utvecklarläge. Från utvecklarläget finns möjligheten att komma åt tabellerna i Dynamics AX.

Till synkroniseringen av Dynamics AX databas och Commerce Runtime databas används Dynamics AX användargränssnitt. Där finns all funktionalitet för att synkronisera databaserna med Synch Service.

För att hämta ut data från Commerce Runtime databas, med SQL-adapter, används Microsoft Visual Studio och SQL Server Management Studio. I uppdragsgivarens datormiljö finns både Microsoft Visual Studio och SQL Server Management Studio installerade. Det är möjligt att använda ett annat program än Microsoft Visual Studio för att skapa en SQL-adapter.

Till Real-Time Service används Dynamics AX användargränssnitt, Microsoft Visual Studio och Internet Information Service 8.

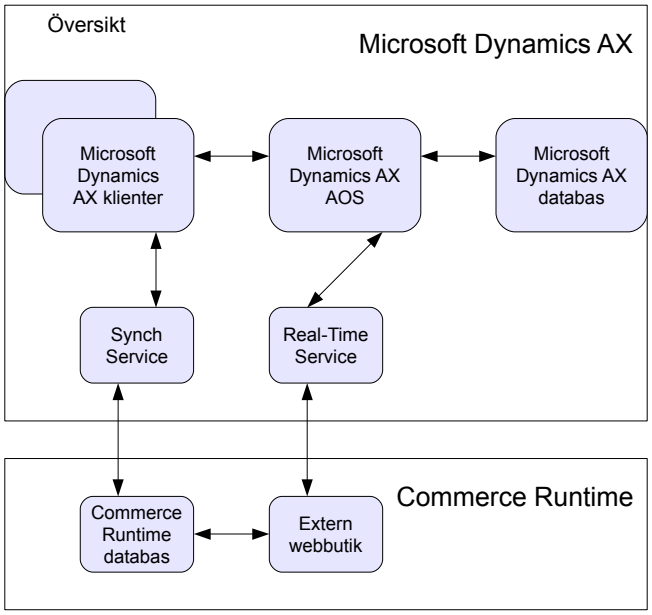
Parprogrammering och pardiskussion

Vid implementationen av kommunikationsmetoder så används metoden parprogrammering. Det finns två orsaker till detta. Dels resulterar det att båda får full förståelse av alla praktiska delar och dels så blir den slutgiltiga implementationen bättre, då parprogrammering är en metod som visat sig vara effektivare än indivuduell programmering[22].

Det är vår uppfattning att två personer utan vana av stora affärssystem kommer mer än dubbelt så långt än någon som arbetar med systemet individuellt med samma kunskapsnivå och inom samma tidsram. Studien om parprogrammering visar resultat på att parprogrammerare slutför en uppgift 40% snabbare än en person som jobbar själv[22]. Artiklen nämner att parprogrammerarna som deltog i studien enhälligt tyckte att paranalys och pardesign var kritiskt för att lyckas, en slutsats som vi med vår erfarenhet delar. Anledningen till detta är att par upptäcker många mer potentiella lösningar och snabbare tar ett beslut om vilken lösning som ska implementeras. Par upptäcker fler lösningar på grund av den aktiva diskussionen som förekommer vid parprogrammering. Detta är något som saknas vid individuell programmering. Uttrycklingen nämns det att par kan klara av uppgifter som är för svåra för en person att klara – vi tror att vårt examensarbete var en sådan uppgift.

ARKITEKTUR

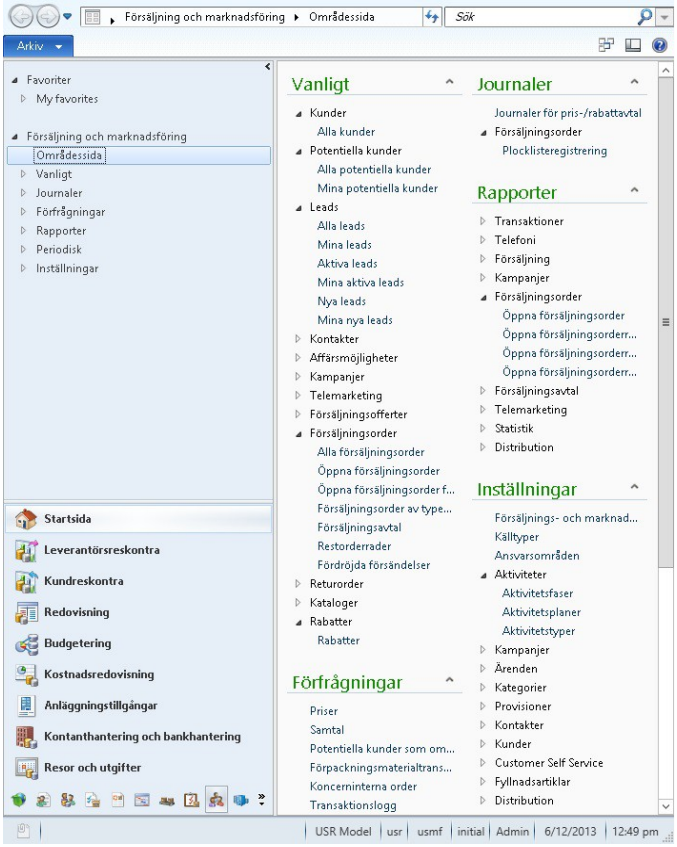
Detta avsnitt ger en översiktlig förklaring av arkitekturen för de olika komponenterna i systemet och plattformen. Figur 1 visar översiktligt kommunikationen mellan komponenterna. Därefter följer en rubrik av varje komponent för en mer detaljerad beskrivning.



Figur 1

I Figur 1 finns Commerce Runtime databas och den externa webbutiken i Commerce Runtime plattform. Commerce Runtime knyter ihop dessa två komponenter och tillåter dessa att kommunicera med Dynamics AX databas via Synch Service eller via Real-Time Service. Commerce Runtime lyfter ut funktionalitet och data från Dynamics AX för att skapa en lokal databas till den externa webbutiken. De tillåter på så sätt en utomstående applikation/butik att kommunicera direkt med Dynamics AX, via den cachade databasen i Commerce Runtime, som om den vore en intern komponent av systemet.

Microsoft Dynamics AX klient



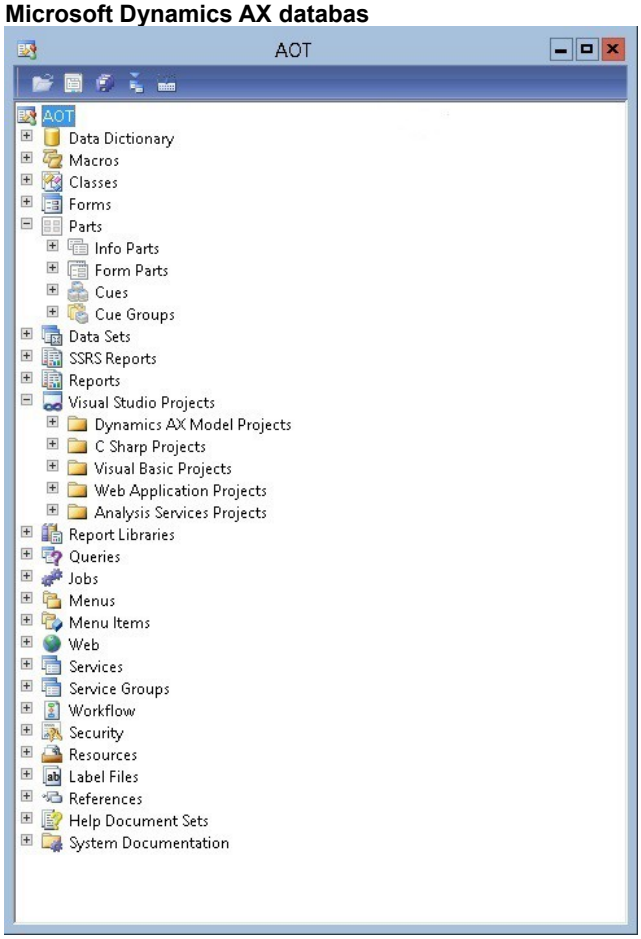
Figur 2

Figur 2 visar en skärmdump från Dynamics AX användargränsnitt, dess klient. Klienten är det första som kunder till Microsoft stöter på när de först öppnar programmet Dynamics AX. Klienten är ett användarläge av Dynamics AX som kunder befinner sig i när de arbetar med programmet. Från klienten kommer kunden åt alla lösningar som Dynamics AX hanterar. I klienten behövs det ingen programmering. I stället klickar kunden sig fram genom flikar och knappar via olika menyer och fält fyllda med data. Klienten är ett verktyg som tillåter kunder att hantera sitt företag från olika aspekter, med stöd för ekonomi, personal, processer och beslut[10].

Microsoft Dynamics AOS

Allt som visas i klienten hämtas från Dynamics AX databas. För att kunna kommunicera med databasen används Dynamics AX *Application Object Server*, AOS, som är en server som lagrar alla programklasser och all kod. AOS är alltså ett logiklager som sköter kommunikationen mellan Dynamics AX klient och Dynamics AX databas.

Microsoft Dynamics AX databas

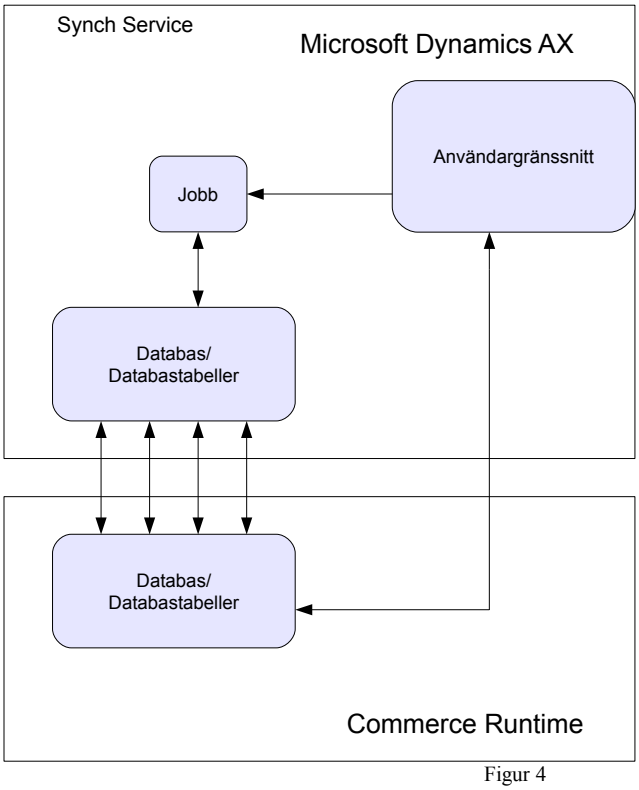


Figur 3

All data som finns i Dynamics AX lagras i dess databas. I Dynamics AX utvecklarläge finns en hierarkisk meny som kallas *Application Object Tree*, AOT, uppbyggt som ett katalogträd, där stora delar av Dynamics AX databas finns tillgängligt. De olika delarna, som t.ex. klasser, är internt organiserade genom hierarkisk namngivning. All funktionelitet som finns i Dynamics AX användargränssnitt är skapad från dess utvecklarläge. Figur 3 visar en bild på AOT.

Synch Service

Figur 4 visar vilka komponenter som möjliggör en synkronisering med Synch Service. Figuren visar upp kopplingen mellan Microsoft Dynamics klient, Synch Service och Commerce Runtime databas i Figur 1.

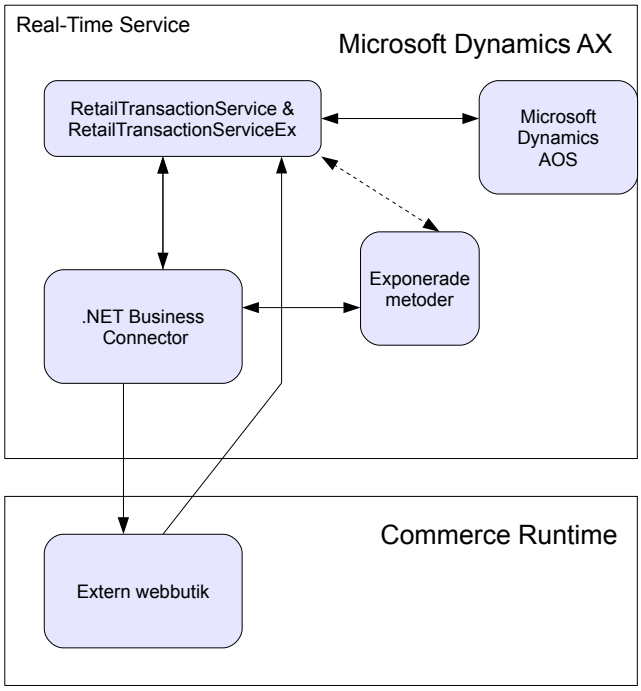


Figur 4

Synch Service är en tjänst som synkroniserar Dynamics AX databastabeller med tabeller i Commerce Runtime databas. Synch Service konfigureras i Dynamics AX användargränsnitt som nås från Dynamics AX klient. Användargränssnittet kopplar ihop vilka tabeller som ska synkroniseras på respektive databas. Detta görs genom att först etablera en kommunikation mellan Dynamics AX och Commerce Runtime databas. Sedan kopplas en databastabell från Dynamics AX till en specifik databastabell i Commerce Runtime genom olika typer av jobb. Jobben genomför en synkronisering åt en bestämd riktning, där antingen Dynamics AX databas speglar Commerce Runtime databas eller tvärtom. En fullständig manual för hur kommunikationsmetoden implementeras beskrivs i avsnittet *Synkronisering mellan Dynamics AX och Commerce Runtime*.

Real-Time Service

Figur 5 visar vilka komponenter som krävs för att kunna använda Real-Time Service. Figuren visar upp kopplingen mellan Microsoft Dynamics AOS, Real-Time Service och den externa webbutiken i figur 1. Den streckade linjen markerar att två komponenter egentligen sitter ihop, men separeras för att förtydliga dess funktion i figuren.



Figur 5

Real-Time Service är en webbtjänst som tillåter synkron kommunikation. Microsoft Dynamics AOS tar hand om alla klasser och all kodkörning i Dynamics AX. I två av systemets klasser, *RetailTransactionService* och *RetailTransactionServiceEx,* exponeras metoder externt [11]. Klasserna är en del av en Windows Service och är konfigurerade till att lyssna på en specifik TCP/IP-port efter efterfrågningar utifrån. När klasserna fått en efterfrågan så lämnas dessa över till AX .NET Business Connector som kör en metod som ligger i någon av klasserna. AX .NET Business Connector är en komponent till Dynamics AX som tillåter en applikation att interagrera med Dynamics AX[12]. I figur 5 är det den externa webbutiken som är applikationen.

Commerce Runtime databas

Commerce Runtime databas är en databas som har möjlighet att synkroniseras med Dynamics AX databas med hjälp av Commerce Runtime. En stor anledning till att en extern databas används är för att trafiken till Dynamics AX kan bli tung om många webbutiker ofta ropar på data från systemet. Istället används en lokala databas, som lagrar den data som är nödvändig för den enskilda webbutiken. Det finns möjligheter att schemalägga synkronisering mellan den lokala Commerce Runtime-databasen och Dynamics AX databas vid de tillfällen då trafiken är lättare för att förhindra väntetider.

Extern webbutik

En webbutik kan skapas med hjälp av Sharepoint. Sharepoint-koden är skriven i Microsoft Visual Studio och generar en hemsida, t.ex. en webbutik. I Visual Studio kopplas sedan webbutiken till Commerce Runtime-

databasen och/eller till webbtjänsten Real-Time Service. Webbutiken kan sedan låta användare interagera direkt mot Dynamics AX databas eller mot en lokal, synkroniserad Commerce Runtime-databas. Från databaserna så kan webbutiken få alla produkter och all butiksrelaterad information den behöver för att visa upp en hemsida för sina kunder.

Navigering i klienten

Det blir snabbt många delsystemsnamn och många termer att hålla reda på i Dynamics AX-programmering. Därför görs en djupare beskrivning av systemen med start från en Dynamics AX-användares perspektiv, som inte är samma sak som en Dynamics AX-utvecklares perspektiv. Detta ger en större förståelse för hur systemen används.

När Dynamics AX startas visas användargränssnittet, se figur 2. Här sköts navigering genom ett adressfält som i Internet Explorer. Två skillnader är att Dynamics AX adressfält har pilar efter varje "mapp" som går kan klicka på. Den ger även användaren alternativ till vart den kan navigera härnäst. Navigeringen är uppbyggd på ett sådant sätt att företag väljs först. Företag är en miljö i systemet och har en mängd tillbehör och en unik uppsättning data. Om en användare klickar på navigeringspilen så väljer den vilken modul klienten ska navigera till. Dynamics AX har så kallade huvudmoduler inbyggda i användargränssnittet:

- *Redovisning*
- *Bank*
- *Kundrelationshantering*
- *Kundreskontra*
- *Leverantörsreskontra*
- *Lager*
- *Huvudplanering*
- *Produktion*
- *Produktbyggare*
- *Personal*
- *Projekt*
- *Grundläggande*
- *Administration*

Förutom de listade modulerna så finns det tilläggsmoduler. I tilläggsmodulerna hittas den mest relevanta modulen för vår utökning, *Butik*. Denna modul, till skillnad från de flesta andra, fokuserar inte på intern företagsshantering utan berör i stället slutkunder (en kund som surfar in på en hemsida och vill köpa en produkt/tjänst) eller hantering av detaljhandel.

Om användaren klickar på navigeringspilen vid *Butik* så finns det en mängd alternativ för de olika delsystemen som en butik består av. Här görs de flesta av inställningarna som berör synkroniseringen av databaser.

METOD 1: SYNKRONISERING MED SYNCH SERVICE

Nedan beskrivs de steg som behövs för att synkronisera den data som finns i Dynamics AX databas med Commerce Runtime databas. I figur 1 undersöker denna metod kommunikationen mellan Dynamics AX AOS, Synch Service och Commerce Runtime databas.

Tabeller med data i Dynamics AX databas

I Dynamics AX finns det så kallade jobb. Jobb innehåller ett block med kod som kan köras helt fristående från resten av systemet. Det finns två sätt att lägga in data i Dynamics AX databas. Det ena sättet är att lägga till tabellkolumner manuellt i databasen och sedan fylla dessa kolumner med värden. Det andra sättet är att skriva in kod i ett jobb som lägger in data i databasen. För att skapa en tabell så används Dynamics AX utvecklarläge. Figur 3 visar att i AOT (Application Object Tree), katalogträdet, finns möjlighet att navigera till *Datastrukturer*, *Tabeller*. Genom att högerklicka och välja *ny* så skapas en ny tabell där kolumner och värden kan läggas in.

I undersökningen används godtycklig data för att undersöka kommunikationen. Det är inte viktigt hur den data som arbetas med ser ut.

Tabeller i Commerce Runtime databas

För att ha något att synkronisera Dynamics AX databas med så kräver det att en tabell i Commerce Runtime databas skapas. För att göra det så används programmet *Microsoft SQL Server Management Studio*. Programmet hanterar ett flertal databaser, bland annat Commerce Runtime databas. Commerce Runtime databas heter *AxRetailSP* och har redan många kolumner och rader fyllda med data. Genom att högerklicka på databasen så skapas en ny tabell. För att kunna koppla ihop de skapade kolumnerna i Dynamics AX så skapas matchande (lagrar samma typ av data) kolumner även i Commerce Runtime databas. Kolumnerna binds ihop vid en senare inställning.

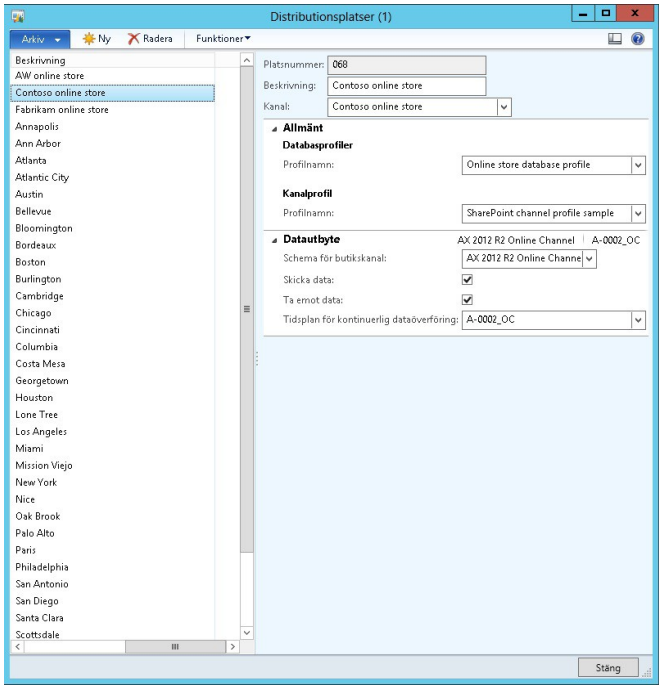
Synkronisering mellan Dynamics AX och Commerce Runtime

För att synkronisera med Synch Service så krävs det ingen programmering. All kod för att synkronisera databaserna finns inbakad i systemet. Istället för att skriva kod används knappar, flikar, menyer och fält för att skriva in värden i Dynamics AX användargränssnitt.

Nedan följer en beskrivning om hur synkronisering kan skapas, där Figur 4 översiktligt visar vilka komponenter som blir beskrivna.

I användargränssnittet i menyn *Butik, Inställningar, Butik schemaläggare*, *Kanalintegrering* finns inställningar för kopplingen till Commerce Runtime databas. Dessa är redan konfigurerade så att kommunikationen fungerar. Av den anledningen undersöks inte dessa inställningar djupare utan används direkt. Databasprofilen som används heter *Online store database profile* och är kopplad till *AxRetailSP*, som

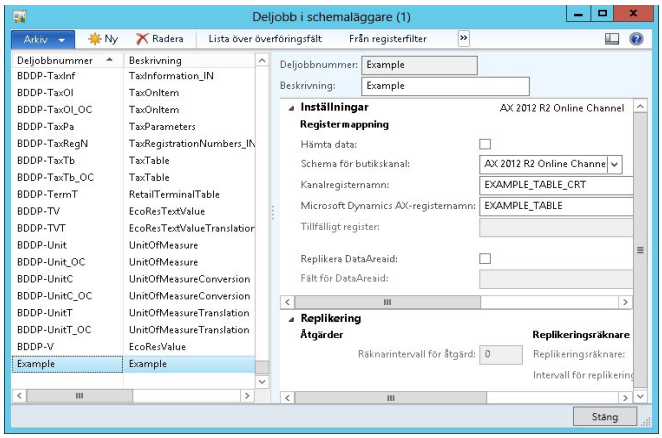
är namnet på Commerce Runtime databas.



Figur 6

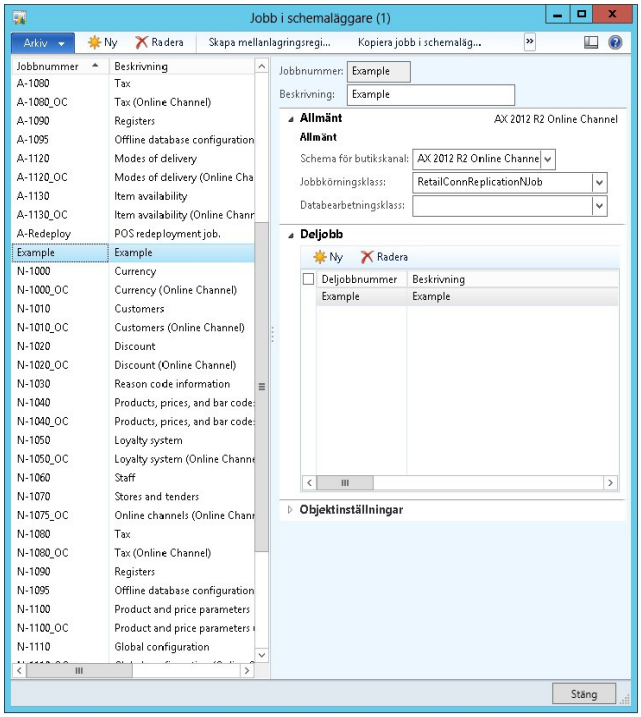
I menyn *Butik, Inställningar, Butik schemaläggare*, *Distributionsplatser,* Figur 6, finns inställningar för att ställa in vilken webbutik som hör till vilken databasprofil. Den redan inställda webbutiken *Contoso online store* som Microsoft använder som exempelbutik i systemet används. Till den kopplas profilen *Online store database profile*, en profil som är kopplad till Commerce Runtime databas. Alla webbutiker får ett platsnummer som identifierar butiken, *Contoso online store* har platsnummer *068*. Till webbutiken ställs även datautbyte in, här väljs *Schema för butikskanal* till *AX 2012 R2 Online Channel* då Commerce Runtime databas får data distribuerad till sig via den kanalen. *Skicka data* och/eller *Ta emot data* kryssas *i/av* beroende på vilken riktning synkroniseringen ska ske.

Under fliken *Funktioner* så finns det en knapp, *Läs schema*. Knappen gör att tabellerna från Commerce Runtime databas läses in, detta måste göras innan deljobb och synkroniseringsjobb skapas för att de ska kunna hitta tabellen som skapats i Commerce Runtime databas.



Figur 7

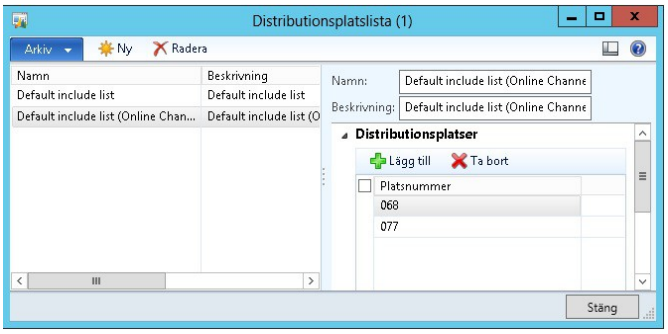
I menyn *Butik, Inställningar, Butik schemaläggare*, *Deljobb i schemaläggare,* Figur 7, skapas deljobb. Med hjälp av ett deljobb så kopplas tabeller från Dynamics AX ihop med tabeller i Commerce Runtime. I inställningarna till det skapade deljobbet väljs tabellnamnet i Commerce Runtime som *Kanalregisternamn* och tabellnamnet i Dynamics AX som *Microsoft Dynamics AX-registernamn*. I fliken över inställningarna, *Lista över överföringsfält*, kopplas kolumnerna i respektive tabell ihop. *Schema för butikskanal* väljs till *AX 2012 R2 Online Channel*.



Figur 8

I menyn *Butik, Inställningar, Butik schemaläggare*, *Jobb i schemaläggare*, Figur 8, skapas synkroniseringsjobb. Ett synkroniseringsjobb kan innehålla ett eller fler deljobb och används för att kunna uppdatera flera tabeller med data samtidigt (d.v.s. kunna köra flera deljobb samtidigt). Det tidigare skapade deljobbet läggs till i det nyskapade

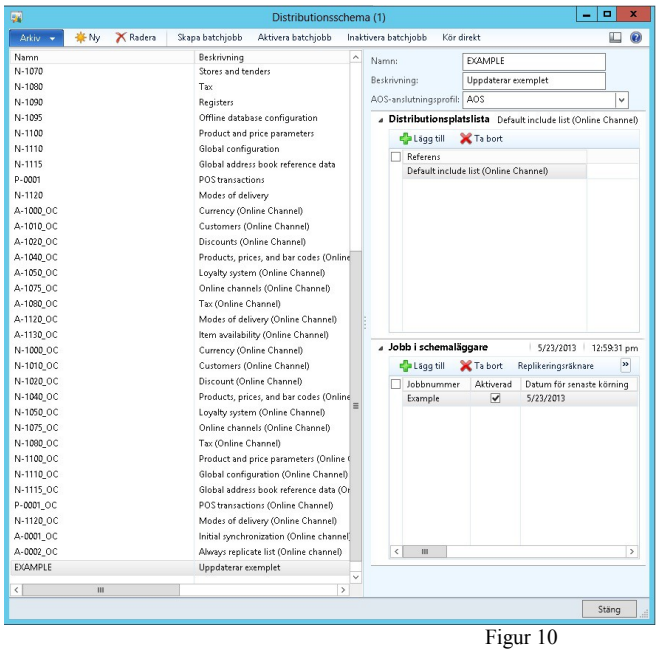
synkroniseringsjobbet så att dess information om vilka tabeller som ska uppdateras skickas vidare. I Figur 8 syns detta under rubriken *Deljobb*, där ett deljobb med beskrivningen *Example* blivit tillagt. I de allmänna inställningarna väljs *Schema för butikskanal* till *AX 2012 R2 Online Channel* och *Jobbkörningsklass* till *RetailConnReplicationNJob*. Det finns tre relevanta jobbkörningsklasser som används och det som skiljer dem åt är vilken typ av jobb de kör. Ett A-jobb uppdaterar den data som finns i databasen, ett N-jobb ersätter all data i databasen och ett P-jobb replikerar data till databasen. För att inte få gammal synkroniseringsdata i tabellerna i Commerce Runtime-databasen väljs jobbkörningsklassen till *RetailConnReplicationNJob*[13].



Figur 9

I menyn *Butik, Inställningar, Butik schemaläggare*, *Distributionsplatslista*, Figur 9, skapas listor av webbutiker. Listorna används i en senare inställning för att välja vilka webbutiker som ska få data distribuerade till dem. *Contoso online store*, med platsnummer 068, ligger i listan *Default include list (Online Channel)* som Figur 9 visar.

I menyn *Butik, Vanligt, Butikskanaler*, *Onlinebutiker* finns det inställningar för den valda webbutiken *Contoso online store*. Här väljs bland annat något som kallas *Företag*. I Dynamics AX finns det flera olika så kallade företag, där varje företag har en egen uppsättning med tabeller och data. För att tabellen som skapats i Dynamics AX ska kunna synkroniseras så krävs det att den är skapad i samma företag som webbutiken hör till. För att redigera ett företag, dubbelklicka på den rad som ska redigeras. I examensarbetet används företaget *usrt*, som står för *Contoso Retail USA*. Om företaget till webbutiken inte matchar företaget där tabellerna är skapade i så kommer inte synkroniseringen att fungera. Detta är för att webbutiken letar efter tabellerna hos ett företag där tabellerna inte finns. Det finns alternativ i Dynamics AX utvecklarläge för att få en tabell att lagras globalt, d.v.s. till alla företag i Dynamics AX, men det innebär att all data blir synlig och tillgänglig för alla.



Figur 10

I menyn *Butik, Periodisk, Datadistribution, Distributionsschema*, Figur 10, så skapas scheman till synkroniseringsjobb. I flikarna till distributionsschemat går det att skapa så kallade *Batchjobb* som kör valda synkroniseringsjobb. Det går även att köra jobb direkt. För att köra det skapade synkroniseringsjobbet så krävs det att ett nytt *körjobb* skapas. I körjobbet läggs det till en referens till den distributionsplatslista som webbutiken *Contoso online store* ligger i, d.v.s. *Default include list (Online Channel)*. Sedan läggs synkroniseringsjobbet till i *Jobb i schemaläggare,* Figur 10 visar ett synkroniseringsjobb med beskrivningen *Example* under denna rubrik. Jobbet körs genom att klicka på knappen *Kör direkt*. När det gjorts är synkroniseringen fullbordad, tabellerna som blivit valda i Dynamics AX och Commerce Runtime kommer nu att vara synkroniserade.

METOD 2: HÄMTA DATA MED SQL-ADAPTER

Den andra kommunikationsmetoden handlar om att hämta ut den synkroniserade datan som ligger i Commerce Runtime databas. I figur 1 är denna metod en undersökning av kommunikationen mellan Commerce Runtime databas och en extern webbutik. I denna metod så implementeras dock inte en webbutik – utan en konsol är tillräcklig för att se om kommunikationen fungerar. Det krävs ingen mer detajlerad bild över implementationen än vad Figur 1 visar eftersom implementationen är väldigt liten.

I Microsoft Visual Studio så skapas ett *Console Application Project*[14]. Ett Console Application Project är en applikation utan ett grafiskt gränssnitt, i stället finns en konsol som används för att läsa och skriva ut data. En sådan applikation passar utmärkt då implementationen görs för att undersöka en kommunikation mellan Commerce Runtime databas och en extern klient. För att hämta ut data så används en så kallad *SQLDataAdapter* för att kommunicera

med Commerce Runtime[15]. En SQLDataAdapter är en brygga som används mellan en SQL Server och ett DataSet. Ett dataset är en container som behåller kolumn- och radstrukturen i en tabell. Att använda en SQLDataAdapter är ett enkelt sätt att kommunicera med externa databaser då den enbart behöver en anslutningssträng och en *query*.

```
string connectionString = "Data Source=(local);
  Initial Catalog=AxRetailSP;
  Integrated Security=True";

string query = "SELECT * FROM dbo.TRACKINGNUMBER";

SqlDataAdapter adapter =
  new SqlDataAdapter(query, connectionString);
```

En *query* ställs direkt mot den berörda databasen. Det data som efterfrågas hamnar sedan i SQLDataAdapter-objektet.

```
DataSet ds = new
  DataSet();adapter.Fill(ds,"TRACKINGNUMBER");
```

För att få ut data från objektet så används funktionen *Fill*. Fill tar data från objektet och lägger det i det dataset som skickas med som parameter.

```
foreach (DataRow row in
  ds.Tables["TRACKINGNUMBER"].Rows)
  {
    foreach (object item in row.ItemArray)
    {
      Console.WriteLine(item);
    }
  }
```

Då datasetet itereras blir all data som tidigare lades in i Dynamics AX databas tillgängligt.

METOD 3: REAL-TIME SERVICE

Den tredje kommunikationsmetoden implementerar ett sätt att kalla på metoder från Dynamics AX via en extern klient med hjälp av Real-Time Service. Figur 5 visar översiktligt vilka komponenter i systemet som blir beskrivna.

Tabeller med data i Dynamics AX databas

Det här steget görs redan då Dynamics AX synkroniseras med Commerce Runtime, se avsnitt *Tabeller med data i Dynamics AX databas* under Metod 1.

Klassmetod i Dynamics AX

Det finns två klasser i Dynamics AX som används för att exponera metoder som en webbtjänst, *RetailTransactionServiceEx* och *RetailTransactionService*. När en metod skapas i en av dessa två klasser så finns det möjlighet att kalla på metoderna externt från en klient. Den enda skillnaden mellan klasserna som upptäckts i detta arbete är att de ropas på med olika funktioner från klienten.

I undersökningssyfte så skapas en metod som endast returnerar ett meddelande i en av klasserna. Metoden döps till *package*. Meddelandet som skrivs ut är "LUIGI", enbart för att se om kommunikationen mellan Dynamics AX och klienten fungerar. Det finns några krav för att metoden som skapas ska kunna användas[16]:

- Metoden måste vara en *public static*-metod.
- Returvärdet till metoden måste vara en container med en längd på minst två.
- De första elementen i containern måste vara en boolsk variabel följt av en sträng
- Det sista elementet måste vara en variabel av följande Microsoft AX primitiva typ: *boolean, int, int64, date, str, guid* eller *Real*.

När metoden skapas och följer kraven så blir den automatiskt exponerad som en webbtjänst.

Konfiguration av server och klient

Till Real-Time Service finns det en server vars uppgift är att hantera webbtjänsten som exponeras från Dynamics AX. För att kunna konfigurera servern så används programmet *Internet Information Service 8*, IIS8. I IIS8 heter den webbtjänst som Real-Time Service exponerar *CommerceDataExchangeRealtimeService*. Till tjänsten finns en tillhörande konfigurationsfil, *web.config*. Detta är serverns konfigurationsfil. Konfigurationsfilen är skriven i *XML* och bestämmer hur servern ska bete sig. För att kommunikationen mellan en server och en klient ska fungera så kräver det att säkerhetsnivån konfigureras enhetligt. Även med en fungerande kommunikation så behöver World Wide Web-Service startas om med hjälp av IIS8 för att ändringar i koden till Real-Time Service ska uppdateras.

Det första steget som krävs är att konfigurera *web.config* till servern. *Security mode* valdes till *Transport* och *clientCredentialType* till *None*. Transport har inbyggd säkerhet, därför krävdes det ingen *protectionLevel*. Även om en *protectionLevel* lades till påverkar det inte konfigurationen något, då Transport ignorerar alla andra valda säkerhetsinställningar[17] (därav den överstrukna kodraden i exemplet nedan). Examensarbetet undersöker dock inte frågor gällande säkerhet i de implementerade metoderna utan endast kommunikationen.

```
<security mode="Transport">
  <message clientCredentialType="None"
protectionLevel="EncryptAndSign" />
```

När servern är konfigurerad kan en klient som används för att ropa på webbtjänsten skapas. För att göra det används programmet Microsoft Visual Studio och den tidigare skapade konsolapplikationen. När en konsolapplikation

skapas genereras en fil automatiskt som heter *App.config*, vilket är klienten konfigurationsfil. För att klienten ska kunna agera server åt tjänsten som implementeras så läggs en webbreferens till adressen *http://ax2012r2a:9080/-CommerceDataExchangeRealtimeService/Service.svc* i konsolapplikationen. Om tjänsten blir navigerad till med en webbläsare visar sig en guide för hur klienten ska skapas för att kunna använda tjänsten. Guiden beskriver att filen *svcutil.exe* ska köras. Svcutil är ett verktyg som använder den skapade webbtjänstens serverkonfiguration för att skapa en motsvarande klientkonfigurationsfil[18]. Svcutil körs från en kommandorad med denna syntax:

svcutil.exe http://ax2012r2a.contoso.com:9080/CommerceDataExchangeRealtimeService/Service.svc?wsdl

Kommandot genererar en konfigurationsfil och en *TransactionService.cs*-fil. Konfigurationsfilen ska ersätta den befintliga *App.config* som skapades med konsol-applikationen. *TransactionService.cs* innehåller koden till klientens klass, den klistras in i den skapade konsolapplikationen för att kunna anropa webbtjänsten[19].

Klientens anrop

Den skapade konsolapplikationen används för att skriva kod för att komma åt de exponerade metoderna i webbtjänsten. För att hitta metoderna så skapas ett *TransactionServiceClient*-objekt, som är ett klassobjekt till klienten. För att kunna skapa ett sådant objekt krävs det att en dll-fil läggs till. I implementationen används denna dll-fil: *C:\Program Files (x86)\Microsoft Dynamics AX\60\Commerce Data Exchange\Synch Service\bin\Microsoft.Dynamics.Retail.TransactionServices.ClientProxy.dll*.

Funktionen *InvokeExtensionMethod* används för att komma åt den skapade metoden i klassen *RetailTransactionServiceEx*. För att komma åt metoder i klassen *RetailTransactionService* så används funktionen *InvokeMethod*. Båda funktionerna tar tre argument: en variabel av *info*-typ, en sträng med klassmetodens namn och sist de parametrar som klassmetoden vill ha.

```
TransactionServiceClient client = new
  TransactionServiceClient();
object[] parameters = new object[1];
```

Ett klientobjekt och en parameterlista skapas.

```
schemas.microsoft.com.Dynamics._2012._05.
  CommerceRuntime.TransactionService.RequestInfo
  info = new schemas.microsoft.com.Dynamics.
  _2012._05.CommerceRuntime.TransactionService.
  RequestInfo();
parameters[0] = "2014";
```

Ett informationobjekt skapas och parameterlistan fylls i.

```
schemas.microsoft.com.Dynamics._2012._05.
  CommerceRuntime.TransactionService.ServiceResponse
  serviceResponse =
  client.InvokeMethod(info, "package", parameters);
```

Funktionen som tidigare skapats, *package,* anropas och svaret sparas i ett *ServiceResponse*-objekt.

```
Console.WriteLine(serviceResponse.Data[0] as string);
```

Den data som klienten får tillbaks skrivs ut i konsolen.

DISKUSSION

CommerceEntity & Services

CommerceEntity är basklassen för alla entiteter i Commerce Runtime[20]. CommerceEntity-klassen är en egenskapsbehållare som lagrar egenskaper som nyckel-värde, som en hash-tabell. De lagrade egenskaperna motsvarar fält i Commerce Runtime databas. Detta gör att programmerare inte behöver komma ihåg det exakta namnen på kolumnerna när klassen används, vilket kan vara passande då databasen är stor.

I Commerce Runtime finns det *services* som utför olika arbeten[5]. Ett exempel på en service är *ProductAvailabilityService*, som räknar ut hur många tillgängliga produkter det finns av en vara. Det går att implementera nya services om de som erbjuds inte räcker till, och det går även att redigera befintliga. Services sköter kommunikation med databasen och är en del av ett så kallat *workflow*, som i sin tur används när Commerce Runtime *Application Programming Interface*, API, används. Commerce Runtime API erbjuder sin funktionalitet till en webbutik.

SQL-adapter eller CommerceEntity

Den största anledningen till att SQL-adapter används är för att den passar bättre till ändamålet – den är enkel att ansluta till en databas och den kräver väldigt lite tid att implementera. Till skillnad från CommerceEntity-klassen så är användningsområdena för en SQL-adapter väl-dokumenterade. Commerce Runtime databas som används är liten. Det gör att CommerceEntity-klassens lagrade kolumnnamn inte skulle underlätta implementationen. Något som också påverkar valet av verktyg var mängden tid som fanns tillgängligt. För att kunna förstå hur CommerceEntity används och implementeras skulle för många timmar krävas. Detta påverkade även valet av verktyg.

I ett stort projekt där det finns intresse att nyttja alla egenskaper hos Commerce Runtime så skulle CommerceEntity vara ett mycket bättre alternativ, då mer inbyggd funktionalitet erbjuds. De services som är tillgängliga täcker många av de behov som finns då en

webbutik implementeras: som att lagra kundinformation, räkna ut valuta enligt nuvarande växlingskurs eller att ansluta till ett betalningssystem.

Tjänsternas för- och nackdelar

Synch Service

Synch Service i grunden är en möjlighet att uppdatera en tabell från en databas till en annan. Detta görs antingen i ett schemalagt förfaringssätt med *batchjobb* eller körs direkt med så kallade *körjobb*. De inbyggda begränsningarna gör att det endast går att göra dataöverföringar, inga omberäkningar eller andra anrop kan skapas.

Synch Service handlar i första hand om datauppdateringar, men också om den prestandafördel som följer av den asynkrona, schemalagda kommunikationen. Synch Service lägger upp sin kommunikation med mål att minska antalet nätverkspaket som behövs. Detta ger en prestandamässig fördel över Real-Time Service, som använder synkron kommunikation.

För en slutkunds webbutik så är det troligtvis pris- och produktuppdateringar som är aktuella i första hand. Att kunna schemalägga, d.v.s. skapa *batchjobb*, till de större datatransaktionerna och veta att överföringen kommer att vara effektiv och inte blockerande kommer också vara av intresse.

En mer subtil fördel med Synch Service är att arbete och uppdateringar kan automatiseras då överföringen av data kan schemaläggas.

Real-time Service

Real-Time Service har fördelar som att få ett omedelbart svar och autentisering. En annan möjlighet med Real-Time Service som inte är uppenbar är att det går att välja olika svarssätt beroende på data genom kontrollsstrukturer i koden. För en webbutik så är Real-Time Service nödvändig för t.ex. personalinloggning och kundförfrågningar om aktuella kampanjer med information som ändras snabbt.

För en intern webbutik så är Real-Time Service ett enkelt sätt att ändra data i Dynamics AX, med den data som butiken får. Det går också att programmera ett dynamiskt system till skillnad från Synch Service som bara kan göra dataöverföringar.

Real-Time Service kan användas för att förprogrammera funktionalitet i Dynamics AX användargränssnitt som sedan kan säljas/ges ut som ett färdigt paket.

När bör vilken tjänst användas

Utifrån den undersökning som gjorts så lämpar sig Synch Service bättre till när mycket information ska uppdateras rutinmässigt. Synch Service lämpar sig även där det inte finns ett behov av att ändra koden som hanterar data, t.ex. vid informationsuppdateringar av en produkt och tillägg av nya produkter i existerande ramverk.

Real-Time Service lämpar sig när data ska hanteras på ett antal bestämda sätt och då beräkningar av data ska genomföras och lämna tillbaka ett dynamiskt svar som inte är en kopiering av en tabell. Ett exempel är vid jämförelser mot olika tabeller eller då det genomförs analyser av tabellinnehåll och paketeras som en tjänst.

SLUTSATSER

Synch Service

Implementation

Undersöks systemet noggrant så är en synkronisering lätt att skapa. Den kräver att vi förstår hur delar i systemet kommunicerar och vilka steg som måste tas. Nu när vi förstår detta så tar hela synkroniseringen några minuter att sätta upp.

Är utökningen tillförlitlig?

När vi skapar jobb för dataöverföring så fungerar dessa rent kommunikationsmässigt. Jobb är enkla att skapa och i jobben är det enkelt att välja vilka fält som ska uppdateras. Det är även enkelt att välja vilken riktning synkroniseringen ska ske åt, d.v.s. från Dynamics AX till Commerce Runtime eller tvärtom. Det svåraste med Synch Service är att hitta och förstå hela kedjan från ena databasen till den andra.

Största problemen med synkronisering

Största problemet som nya användare av det här systemet är svårigheterna att följa den befintliga dokumentationen. Många exempel innehåller punktlistor där vissa punkter är komplexa och beskrivs på en nivå som gör att det är svårt att följa. Den största delen av examensarbetet lades ner på att försöka hitta en förklaring på hur synkroniseringen skulle ske. I efterhand känns många av de svårtydda momenten i dokumentationen triviala – men det gjorde de inte vid första ögonkastet. Dokumentationen ger oss en bra utgångspunkt och utan den hade examensarbetet aldrig gått att färdigställa. Det är vår förhoppning att denna rapport låter läsaren återskapa de implementerade metoderna utan svårigheter och att vår rapport kommer komplettera den befintliga dokumentationen väl.

Real-Time Service

Implementation

Real-Time Service är den implementation som är enklast att uföra, tack vare den tydliga dokumentationen. Efter att ha skapat en metod i en av de exponerade klasserna så finns den sedan automatiskt tillgänglig för webbanrop. Klientkoden för anrop är inte heller svår att implementera i efterhand. Uppskattningsvis tar det tio minuter att få hela kedjan uppe, från att exponera metoder i Dynamics AX till att anropa metoderna från en klient.

Är utökningen tillförlitlig?

Utökningen fungerar bra efter att vi skapar den. Den är väldigt flexibel då vi kör kod både på Dynamic AX och på vår anropande klient. Detta ger oss mycket kontroll på vad

vi vill ska hända och ger stora möjligheter att vara kreativa och utöka systemet med nya funktioner. Om läsaren har erfarenhet av att jobba med webbtjänster så är övergången till Real-Time Service enkel att göra.

Största problemen med Real-Time Service

Största problemet med Real-Time Service är XML-konfigueringen för klient och server. Det är svårt att veta var alla inställningar görs, det är t.o.m. svårt att veta var konfigurationsfilerna är till en början. Nu vet vi att XML-filernas konfiguration måste matcha varandra på serven och klienten. En annan stor utmaning är att få klienten att anropa webbtjänsten. Problemet där är att det finns en uppsjö av klientobjektreferenser att välja bland och det är inte tydligt vilken som är korrekt att använda.

REFERENSER

[1] The Microsoft Team (2012). *Inside Microsoft Dynamics AX 2012*. Microsoft Press.

[2] Microsoft (2013). *Microsoft Dynamics AX*. http://www.microsoft.com/sv-se/dynamics/ax.aspx.

[3] Microsoft (2013). *Retail [AX 2012]*. http://technet.microsoft.com/en-us/library/hh597299.aspx.

[4] The Microsoft Dynamics AX product team blog (2012). *A sneak peek tour: 4 Business Benefits of Microsoft Dynamics AX 2012 R2*. http://blogs.msdn.com/b/dax/archive/2012/10/23/a-sneak-peak-tour-4-business-benefits-of-microsoft-dynamics-ax-2012-r2.aspx.

[5] Microsoft (2013). *Commerce Runtime [AX 2012]*. http://technet.microsoft.com/en-us/library/jj916620.aspx.

[7] Microsoft (2013). *Install and configure Commerce Data Exchange: Real-time Service [AX 2012]*. http://technet.microsoft.com/EN-US/library/jj679919.aspx.

[8] Microsoft (2013). Commerce Data Exchange: Synch Service overview [AX 2012]: http://technet.microsoft.com/en-us/library/jj973039.aspx.

[9] Microsoft (2013). *AOS Overview [AX 2012]*. http://msdn.microsoft.com/en-us/library/aa660629.aspx.

[10] Microsoft (2013). *Varför Dynamics*. http://www.microsoft.com/sv-se/dynamics/varfor-dynamics.aspx.

[11] Shane Erstad. *AX for Retail 2012: Customizing the Transaction Service.* http://blogs.msdn.com/b/axsupport/archive/2012/04/18/ax-for-retail-2012-customizing-the-transaction-service.aspx.

[12] Microsoft (2013). *NET Business Connector Overview [AX 2012]* http://msdn.microsoft.com/en-us/library/aa659581.aspx.

[13] Microsoft (2013) *Configure jobs and subjobs in Retail Scheduler [AX 2012]* http://technet.microsoft.com/EN-US/library/jj680084.aspx

[14] Microsoft (2013) *Console Application Template* http://msdn.microsoft.com/en-us/library/k1s1x6ck(v=vs.80).aspx

[15] Microsoft (2013). *SqlDataAdapter Class.* http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=vs.71).aspx

[16] Microsoft (2013). *How to: Extend the Commerce Data Exchange: Real-time Service [AX 2012]*. http://technet.microsoft.com/en-us/library/dn126098.aspx.

[17] Microsoft (2013). *Understanding Protection Level.* http://msdn.microsoft.com/en-us/library/aa347692.aspx.

[18] Microsoft (2013). *ServiceModel Metadata Utility Tool (Svcutil.exe).* http://msdn.microsoft.com/en-us/library/aa347733.aspx.

[19] Okänd författare. *Transaction Service.* http://cloud.gosecureauth.com/SATransaction/Transaction.svc

[20] Microsoft (2013). *How to: Customize the Data in a Commerce Entity [AX 2012]*. http://msdn.microsoft.com/en-us/library/jj916616.aspx.

[22] Laurie Williams, Robert R. Kessler, Ward Cunningham & Ron Jeffries (2000) *Strengthening the Case for Pair Programming.* IEEE Software.

[23] Microsoft (2013). *Configure and schedule retail data distribution [AX 2012]*.http://technet.microsoft.com/en-us/library/jj679920.aspx.

[24] Microsoft (2013). *Set up a Real-time Service profile [AX 2012]*. http://technet.microsoft.com/EN-US/library/hh580631.aspx.