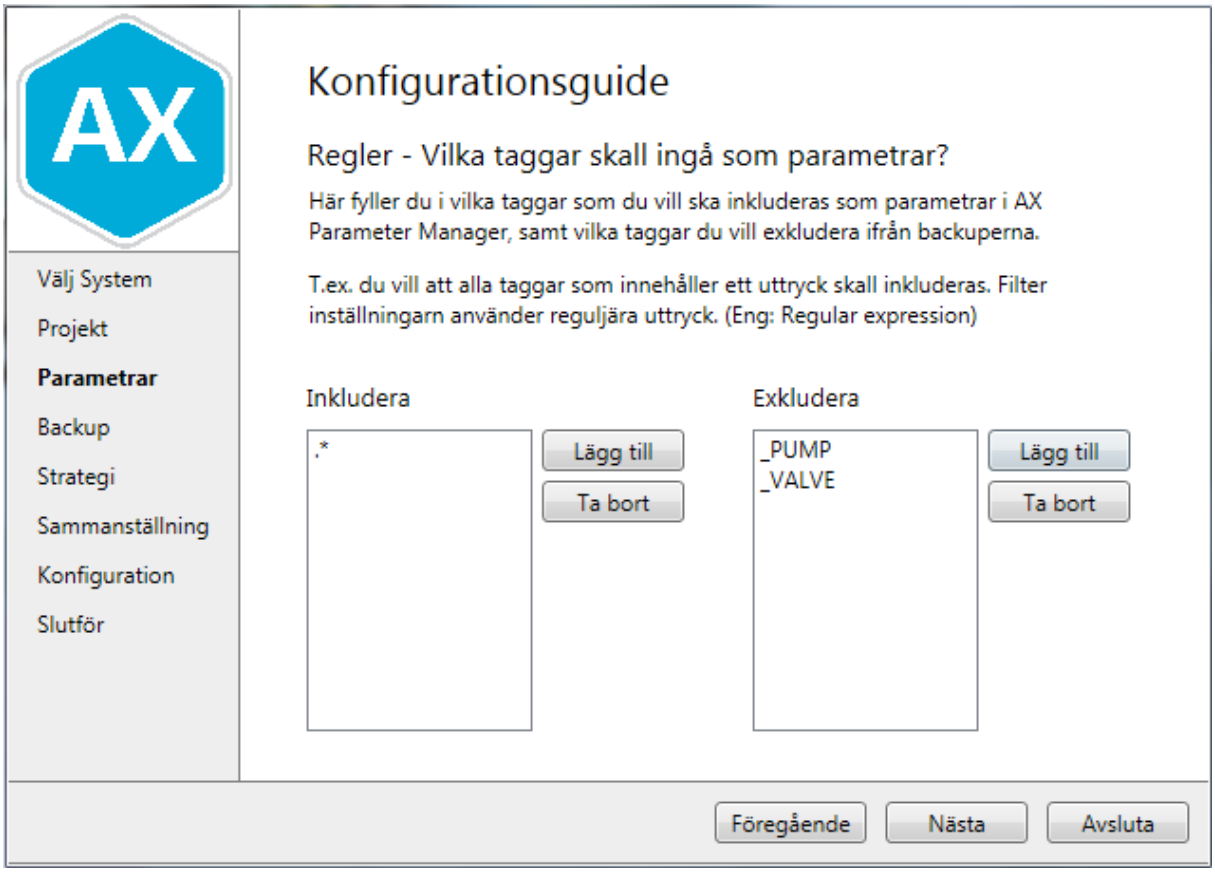
CHALMERS



Konfigurationsguide för backup-program avseende driftparametrar

Configuration guide for a backup-program regarding operation parameters

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Niclas Karlsson
Pontus Savolainen

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2016

Konfigurationsguide för backup-program avseende driftparametrar
Configuration guide for a backup-program regarding operation parameters
Niclas C. Karlsson
Pontus M. Savolainen

Department of Signals and Systems
Chalmers University of Technology
SE – 412 96 Gothenburg
Sweden
Phone + 46 (0)31 – 771 1000

Cover:
The cover picture shows a screenshot of a page in the finished product.
Taken by: the authors

CHALMERS

Förord

Fyra år sedan träffades det två vilsna själar på Götaplatsen i Göteborg, de skulle börja på Mekatronikingenjörsprogrammet på Chalmers. Det var vi, författarna, som träffades, Pontus och Niclas. Sedan dag ett har vi studerat tillsammans och vi har utfört majoriteten av alla projektarbeten tillsammans. Så när tiden kom för att vi skulle göra exjobb var det inget tvekande om vi skulle göra ett projektarbete tillsammans för en sista gång.

När vi satte oss ned under hösten tvåtusenfemton och började planera exjobb så vart det ingen självklarhet var vi skulle göra exjobb den kommande våren. Men, som snart utbildade ingenjörer, öppnade vi ett Excel-ark och börja lista alla företag som vi skulle kunna tänka oss att göra exjobb på, ifall de erbjöd exjobb, inom vilket område exjobbet skulle vara i och om vi hade någon i vårt kontaktnät som jobbade på företaget. Därefter började vi eliminera och rangordna företag efter olika kriterier. Det företag som hamnade högst upp i vår lista var AcobiaFLUX. Vi hamnade tillslut på AcobiaFLUX.

Först och främst vill vi tacka Robin Bandgren och Hanna Mandir på AcobiaFLUX. Robin för att han la in ett gott ord om oss och Hanna för att hon trodde på oss och tog in oss för att göra exjobbet. Sen vill vi ge ett stort tack till Daniel Gunnarsson på AcobiaFLUX och Morgan Osbeck på Chalmers för att ni har stått ut med oss som handledare och väglett oss när vår egen kunskap inte räckt till.

Sist så vill vi tacka alla på AcobiaFLUX som har gett oss många roliga och lärorika veckor. Det har varit ett nöje att få känna sig som en del av er grupp och fått lära känna er. Vi har trivts väldigt bra och känt oss mycket välkomna, vi starkt rekommendera andra att göra examensjobb på företaget.

Ha en trevlig läsning och tack för oss,
Pontus Savolainen och Niclas Karlsson

Sammanfattning

I takt med att automatisering ökar i dagens samhälle ökar behovet av säkra automationssystem, ifall ett system kraschar kan värdefulla inställningar och justeringar gå förlorade, vilket kan ta dagar till veckor att återgå till. Som svar på det har AcobiaFLUX AB i Göteborg utvecklat ett säkerhetskopieringsprogram, AX Parameter Manager (AXPM), som utför backuper på kritiska inställningar i automationssystem. Problemet med AXPM är att det i programmet är för svårt att konfigurera inställningarna av val för vad som skall backas upp och hur backupen ska ske. Arbetet skedde under en 10 veckors period där uppgiften var att utveckla en intuitiv och användarvänlig metod för att konfigurera AXPM. Resultatet av arbetet är att konfigurationsmetoden blev i form av en installationsguide där användaren får på ett intuitivt sätt utan större förkunskap kunna ansluta sig till rätt databas, ställa in vilka inställningar som ska säkerhetskopieras och hur säkerhetskopieringen ska se ut. Slutresultatet har en stor potential att vara en del av AcobiaFLUX produktutbud.

Summary

As automation is increasing in today's society, the need for secure and stable automations system is increasing as well. If a system crashes valuable settings and adjustments can be lost, which may take days or even weeks to recover. In response AcobiaFLUX AB in Gothenburg have developed a backup program, AX Parameter Manager (AXPM), which makes backups on crucial settings in automation system. The problem with AXPM is that the program is hard to configure which settings to include in the backup process. The assignment took place during a 10 weeks period were the task was to develop and design an intuitive and user-friendly method to configure AXPM. The result was a configuration method formed in to a configuration guide were the user in an intuitive way without needing any major prior knowledge could connect to the right database and choose the right settings. The end result have great potential to be in AcobiaFLUX AB product range

Innehåll

Beteckningar

AXPM	AX Parameter Manager
C#	Programspråket C#
IPC	Interprocess Communications
JSON	JavaScript Object Notation
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
TPL	Task Parallel Library
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Uttryck

Cron-uttryck	Ett utryck för tid/tidsintervall
Deserialisera	Konvertera/Packa upp en JSON eller XML-fil till ett objekt
Reguljärt uttryck	Ett uttryck som används för att matcha mönster i en text
Serialisera	Konvertera/Packa ner ett JSON eller XML objekt till en fil

1. Inledning
Arbetets syfte är att förbättra och förenkla ett program för säkerhetskopiering. I detta kapitel beskrivs några av grunderna kring arbetet och varför det utförts.

1.1 Bakgrund
AcobiaFLUX levererar lösningar inom industriell IT och automation till kunder i segmenten industri, infrastruktur och fastighet. De arbetar med hela processen ifrån problemanalys, design, genomförande, installation till support.

I många automationssystem, främst i produktionsprocesser, finns det många parametrar. Dessa parametrar justeras ofta efter driftsättning för att effektivisera processerna, ändringarna lagras endast lokalt i styrsystemen. Utan att ha en strategi eller ett program som säkerhetskopierar parametrar kan alltså ett haveri av ett styrsystem vara förödande. Därför har AcobiaFLUX skapat säkerhetskopierings- och övervakningsprogrammet AX Parameter Manager (AXPM) för att ha kontroll över parametrarna hos systemen.

Problemet AcobiaFLUX i dagsläget har, även för företagets egna medarbetare, är att det i AXPM är för komplicerat att konfigurera vilka inställningar man vill ha vid säkerhetskopieringen. Därför vill AcobiaFLUX gärna skapa ett grafiskt gränssnitt för konfigureringsverktyget. Dessutom vill de göra allmänna förbättringar av programmet för att göra det lättare och tydligare för användaren.

1.2 Syfte
Projektets huvudsyfte är att öka användarvänligheten hos AXPM, att göra det enklare och effektivare för en användare att konfigurera inställningar för säkerhetskopieringen. Dessutom göra programmet tydligare.

1.3 Avgränsningar
För att hålla projektet inom tidsramen så kommer ingen större förundersökning hos AcobiaFLUXs kunder genomföras. Dessutom tror uppdragsgivaren att ändringen ifrån prototyp till fullfjädrat verktyg kommer vara en nog stor uppdatering. Sedan kommer projektet endast behandla säkerhetskopiering för CitectSCADA.

1.4 Mål
Arbetet har som mål att skapa ett grafiskt gränssnitt där att vid utvärdering av resultatet skall följande tre första frågor besvaras med ”ja”. Därefter ska den fjärde frågan eventuellt besvaras med presentation av någon eller några nya funktioner.

– Är verktyget lättförståeligt och förstår man vilket syfte det har?
– Är designen av gränssnittet utfört så det känns naturligt att manövrerar i det?
– Är det lätt att lägga till eller ta bort parametrar att säkerhetskopiera?
– Vilka extra funktionaliteter har programmet fått under projektets gång?

2. Teknisk bakgrund

Detta kapitel beskrivs de tekniska termer och andra förutsättningar som krävs för att läsaren ska ha tillräckliga kunskaper för att kunna ta in rapportens innehåll. För att läsa mer om den information som presenteras i detta kapitel finns de referenser som använts i slutet av respektive stycke.

2.1 Automationssystem

Automationssystem är system som är skapade för att utföra en process automatiserat för att minska risken att människor blir utsatta för olyckor samt öka effektiviteten i processen, systemen styrs av styrsystem. Styrsystemen är de system som styr processerna och det görs ofta av en programmable logic controller (PLC). PLC är en stabil dator som används inom automation och använder som namnet antyder logik för att styra automationen. Tre stora användningsområden för automationssystem, och de områdena AcobiaFLUX arbetar med, är industri, infrastruktur och fastigheter [1, 2].

2.2 Mjukvara

Nedan följer beskrivningar på den mjukvara som använts och har haft betydelse för projektet.

Supervisory Control and Data Acquisition (SCADA) är ett gränssnitt för att övervaka och styra över automationssystem. Gränssnittet ger en operatör möjligheten att på en central plats påverka och övervaka processer i ett automationssystem. CitectSCADA (Citect) är ett SCADA-system distribuerat av elektroteknikkoncernen Schneider Electric och är ett av de mest använda SCADA-systemen på marknaden. Citect är ett SCADA-system som i stor utsträckning används av AcobiaFLUX [3–5].

Visual Studio är en utvecklingsmiljö framtagen av Microsoft och är mycket integrerat i Microsofts Windows. Det används framförallt för att framställa applikationer till Windows men hanterar även internetbaserade och smartphonebaserade applikationer. Utvecklingsmiljön är kraftfull och lätt att arbeta med, vilket gör det enklare när man arbetar med stora projekt [6].

Structured Query Language (SQL) är ett standardiserat programspråk för databaser, framtaget av företaget IBM. SQL är inte en databas själv utan ett sätt att få fram data ur databasen genom språkliknande kommandon där ett exempel kan ses i figur 1. Där Microsoft SQL Server är en databashanterare baserat på SQL utvecklat av Microsoft [7, 8].

```
SELECT FROM Friends
WHERE FriendName='Robin Bandgren' AND FriendName='Pontus Savolainen';
```

Figur 1: Exempel på ett SQL-kommando

2.3 Utvecklingsverktyg

Nedan följer beskrivningar de utvecklingsverktyg och programspråk som använts i projektet.

.NET (*dot net*) är ett ramverk som bland annat hanterar användargränssnitt, nätverksanslutningar, trådar och minne. Det innehåller också ett stort klassbibliotek för att göra användningen enkel [9–11].

C# (*C-Sharp*) är ett objektorienterat programspråk som har sin grund i programspråket C++ men C# har också många likheter med JAVA. Språket är framtaget av Microsoft och körs på .NET-ramverket, därför används språket mestadels vid skapandet av program till Windows. Likt JAVA så har C# hög typsäkerhet, vilket innebär att det är lätt att upptäcka fel i kod [8, 11].

Windows Presentation Foundation (WPF) är ett system inom .NET som använder XML baserade språket XAML för att enkelt skapa användargränssnitt till applikationer. XML är ett märkspråk vilket är ett sidbeskrivningsspråk som med hjälp av text beskriver hur ett gränssnitt ska presentera text och design [12, 13].

JavaScript Object Notation (JSON) är ett textbaserat format för att skicka data som är lättläst för både människa och dator. Formatet är lätt att serialisera (konvertera) klasser i programkoden till textfiler och kan dessutom deserialisera (packa upp) filer och läsa in dem som klasser. Med formatet kan man på ett lätt sätt med ett program skapa t.ex. inställningsfiler som är både lättlästa för människa och dator. Ett populärt kodbibliotek för JSON i C# är Newtonsoft.Json [14, 15].

Flertrådning (eng. multithreading) är när ett program delar upp ett arbete i flera parallella processer, trådar, för att genomföra flera beräkningar samtidigt. Att arbeta i flera trådar kan göra så att ett program kan köra flera operationer samtidigt så som att både genomföra en tung beräkning samt utföra det grafiska gränssnittet. I Visual studios och C# används flera olika metoder för att få flera trådar, varav en är ”tasks” via kodbiblioteket task parallel library (TPL) [8, 12, 16].

Interprocess Communications (IPC) är flera tekniker som används för att skicka och ta emot information mellan flera trådar i ett eller flera program. Vissa tekniker stödjer också kommunikation mellan olika datorer via nätverk [17].

Pipelines eller ”rör” (eng. pipes) är en typ av IPC som skapar en eller flera klienter och servrar som kommunicerar med varandra. Ett rör kan implementeras som en kommunikationskanal mellan två program [18]. Det finns två varianter av rör:

- Namngivna rör, som stödjer både lokal- och nätverkskommunikation samt flera klienter och serverar. Namngivna rör har också tvåvägskommunikation.
- Anonyma rör, som bara stödjer lokal kommunikation, bara en server och klarar bara av envägs kommunikation (Server till klient).

Reguljära uttryck (eng. Regular Expression) är ett sätt att matcha ett uttryck eller mönster i en text. Det som görs med reguljära uttryck är att man jämför ett uttryck eller mönster, som är skapade efter speciella regler, med en text och indikerar ifall någon stämmer överens [19].

Sekunder	Minuter	Timmar	Datum	Månad	Veckodag	(År)
XX	XX	XX	XX	XX eller YYY	XX eller YYY	XXXX

Tabell 1: Format på ett Cron-uttryck

Cron är ett system inom Unix baserade operativsystem för att utföra uppgifter under ett visst tidsintervall och ett Cron-uttryck är ett sätt att ge tidsintervall en kompakt och hanterlig form. Det accepterar vissa specialkaraktärer för jokertecken, olika intervall och flervals alternativ. Det är i formatet som finns i tabell 1 (X = siffra, Y = bokstav) där årtal är frivilligt. Några exempel på Cron-uttryck kan ses i tabell 2 [20, 21].

Sek	Min	Timmar	Datum	Mån	Dag	Förklaring
0	30	12	*	*	SUN	Varje söndag klockan 12:30.
0	0/20	*	1-15	*	*	Varje 20:e minut mellan den 1:e och 15:e varje månad
0	0	12,14,16	*	6-8	*	Klockan 12:00, 14:00 och 16:00 varje dag mellan juni och augusti

Tabell 2: Exempel på Cron-uttryck

3. Metod
Under arbetets gång har flera olika arbetsgångsmetoder tillämpats. I detta kapitel redovisas de tillvägagångsätt som har använts.

3.1 Förarbete
Inför uppstarten av arbetet skrivs en planeringsrapport och ett Gantt-schema med tidsplanering, det i samarbete med handledare ifrån både Chalmers och uppdragsgivaren. Planeringen sker för att alla parter i arbetet ska ha en klar översikt om vad som ska göras och inom vilken tidsram.

3.2 Uppstart och analys
Vid uppstarten av arbetet görs det förstudier genom att lära sig essentiella programvaror och programspråk, för att samla den kunskap som krävs för att angripa problemet och förstå hur det nuvarande systemet är uppbyggt. Därefter prövas och analyseras det befintliga systemet för att få en förståelse över varför problemen finns. Utifrån analysen av problemet kan lösningsförslag och en plan över utformandet av arbetet skapas.

3.3 Fördjupad problembeskrivning
För att skapa en tydlig målbild arbetet ska gå mot, görs en fördjupad problembeskrivning baserad på den beskrivning som getts vid uppstart och med den kunskap som fåtts under analysen.

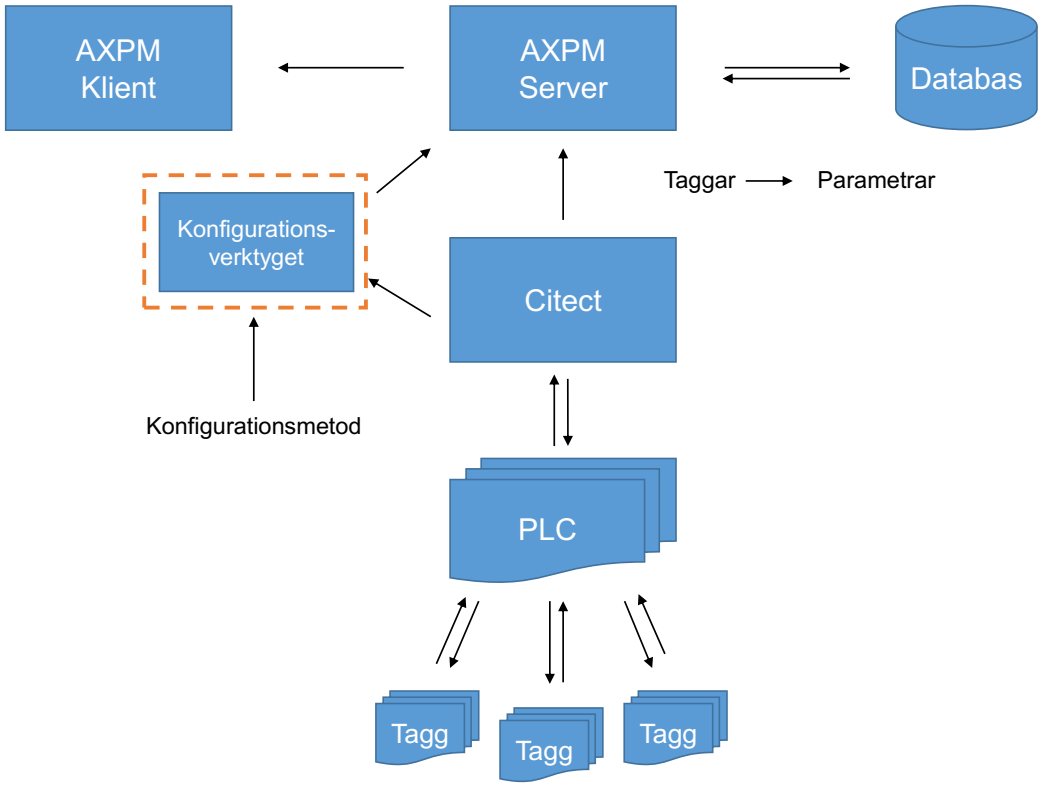
3.4 Utförande
Vid utförande utgås det ifrån den plan som sattes upp vid uppstarten. Sedan arbetas det metodiskt genom varje delmoment. Vid nya uppkomna problem, analysera problemet och sök efter lämpliga lösningförslag. Tillämpa sedan bästa lösningsförslag på problemet och jobba vidare med delmomentet.

3.5 Kvalitetssäkring
Kontinuerligt under arbetet genomförs kvalitetssäkring för att upptäcka eventuella buggar i programmen för att sedan ta bort dem. Kvalitetssäkringen görs i två steg. I första steget testas alla nya funktioner i programmet och i det andra steget gås programmet igenom och alla olika kombinationer av funktioner testas för att slumpmässigt upptäcka buggar. Under slutskedet av arbetet fokuseras det mer på kvalitetssäkring.

3.6 Dokumentation
Kontinuerligt under arbetets gång dokumenteras det även i en rapport för att underlätta i slutskedet av arbetet. I slutet av arbetet fokuseras det mest på att skriva klart rapporten.

4. Analys

Under uppstarten av projektet så utfördes grundliga analyser och tester av de program som vid tillfället fanns. I detta kapitel redovisas de analyser och hur lösningen kan implementeras.



Figur 2: Flödesschema över hur de olika programmen samarbetar. Markerat är den delen som huvudfokus ligger på.

4.1AX Parameter Manager

AXPM är ett program för övervakning- och säkerhetskopiering av parametrar för automationssystem. Programmet hämtar parametrar med dess värden ifrån Citect och utför säkerhetskopiering på dem efter valt intervall. Ett flödesschema på hur de olika programmen är kopplade med varandra kan ses på figur 2.

Programmet är uppdelat i ett par delprogram, de två delarna som utgör huvudprogrammet är en klient och en server. Servern är det programmet som utför säkerhetskopieringen samt kommunicerar mellan Citect och SQL-servern (databasen). Kommunikation mellan Citect och servern görs med hjälp av ett kodbibliotek som AcobiaFLUX själva skapat. Säkerhetskopieringen sker efter den konfiguration som görs av konfigurationsverktyget. Servern körs i bakgrunden som ett program i kommandotolken eller som en Windows service, servern måste alltid vara igång för att man ska ha tillgång till data i klienten, då servern är det program som levererar data till klienten.

Klienten är ett grafiskt gränssnitt där användaren kan övervaka parametrarna, se backuper och se andra händelser. Ifrån servern får klienten all data som den presenterar i ett lättförståeligt format. Klienten ger användaren möjligheten att övervaka parametrars värden, händelseförlopp och ladda in en backup. Första intrycket av användargränssnittet på klienten är att det är lätt och överblickbart, men saknar bra flikhantering vilket gör att användaren lätt tappar översikt på vad man som händer.

4.2 AX Parameter Manager Configuration Tool

AX Parameter Manager Configuration Tool (konfigurationsverktyget) är det program som utför säkerhetskopieringen efter en bestämd inställningsfil. När konfigurationsverktyget startas läser den in inställningar ifrån en förbestämd inställningsfil där den hämtar vilka villkor som programmet ska filtrera ut parametrar efter.

Inställningsfilen är en fil som innehåller information om hur konfigurationsverktyget ska koppla sig till databasen, inställningar gällande hur säkerhetskopieringen ska vara och efter vilka villkor programmet filtrerar parametrar. För att ändra inställningarna för konfigurationen krävs det att inställningsfilen öppnas manuellt och därefter görs de önskade ändringarna.

Det första som händer när konfigurationsverktyget startar är att den läser in alla inställningar ifrån inställningsfilen. Utifrån inställningarna väljer verktyget mellan två lägen, att ta bort eller uppdatera parametrar. Båda lägena fungerar men måste köras separat, verktyget kan inte uppdatera parametrar och ta bort parametrar samtidigt. När verktyget ska uppdatera vilka parametrar den ska ta backup på söker den genom alla projekt i Citect, projekt som är givna ifrån inställningsfilen. Därefter så jämför verktyget alla namnen på parametrarna i de givna projekten med reguljära uttryck som fås ifrån inställningsfilen, ifall ett namn på en parameter innehåller ett av de reguljära uttrycken så läggs den parametern till på en lista över parametrar som ska säkerhetskopieras.

Därefter konfigurerar konfigurationsverktyget servern så att den säkerhetskopierar de valda parametrarna. Sedan ställer verktyget in hur ofta en backup ska ske och hur länge en säkerhetskopia ska sparas.

Om användaren vill så kan konfigurationsverktyget köras i ett rapporteringsläge där den endast genererar en rapport på vad som kommer ändras ifall man skulle köra verktyget skarpt. Detta läge väljs i inställningsfilen. Oberoende om man kör verktyget i rapporteringsläge eller inte så loggas alla händelser som hänt under konfiguration i en loggfil.

4.3 Konfigurationsguiden

Sedan tidigare har AcobiaFLUX en prototyp på en metod där man ställer in konfigurationsverktyget. Prototypen (konfigurationsguiden) är framtagen i formatet WPF och är utförd som en vanlig installationsguide i Windows.

I bilaga A finns det bilder på hur konfigurationsguiden ser ut, sida för sida. Guiden är utformad så att användaren successivt sida för sida anger den information som konfigurationsverktyget kommer behöva. Den information som matas in på varje sida i guiden sparas undan temporärt inom programmet men lagras inte i någon fil eller används inte till något. Ett par funktioner och inmatningssätt i guiden är inte riktigt intuitiva och användaren kan ha svårt att förstå varför de ska matas in. Just nu så utför guiden inte mycket och om den ska fungera så behövs det mycket arbete.

Prototypen av konfigurationsguiden är långt ifrån fullständig och är främst ett grafiskt skal. För att den ska fungera behövs samtliga funktioner kodas. Dessutom behövs det grafiska gränssnittet utvecklas för att vara mer intuitiv och lättförståelig.

5. Problembeskrivning

Huvudproblemet är att skapa en intuitiv metod för användaren att välja parametrar och inställningar för konfigurationen. Problemet är att välja hur konfigurationsmetoden ska vara utförd, att analysera olika alternativ och väga dem mellan varandra. Följande kapitel beskriver problemen, målen och kraven i mer detalj för delproblemen.

5.1 Användargränssnitt

För att användaren lätt och utan större eftertanke kan utföra konfigurationen av AXPM behövs ett enkelt användargränssnitt till konfigurationsmetoden. Detta kan antingen implementeras i konfigurationsverktyget eller läggas som fristående program som kommunicerar med konfigurationsverktyget. Främsta kravet på användargränssnittet är att det är intuitivt, att användaren utan större kunskap ska ha full förståelse. Andra krav är att användaren, med hjälp av gränssnittet, ska ha möjligheten att göra alla inställningar för konfigurationen för att eliminera behovet att ändra i programfiler.

5.2 Konfiguration

Ett problem med konfigurationen är att avväga hur mycket av konfigurationen som bestäms av konfigurationsmetoden via användargränssnittet och hur mycket som ligger i programfiler. Att hitta en avvägning över vad som är viktigt och hur mycket som är bra att ha i konfigurationsmetoden utan att den blir för komplicerad och omständlig.

Ett annat problem är hur man rent tekniskt löser konfigurationen. Problemet är hur man konfigurerar inställningarna, viktigt är att hålla ordning på vad som sker i bakgrunden i .NET och vad sker i löpande i löpande i programmet.

5.3 Kommunikation

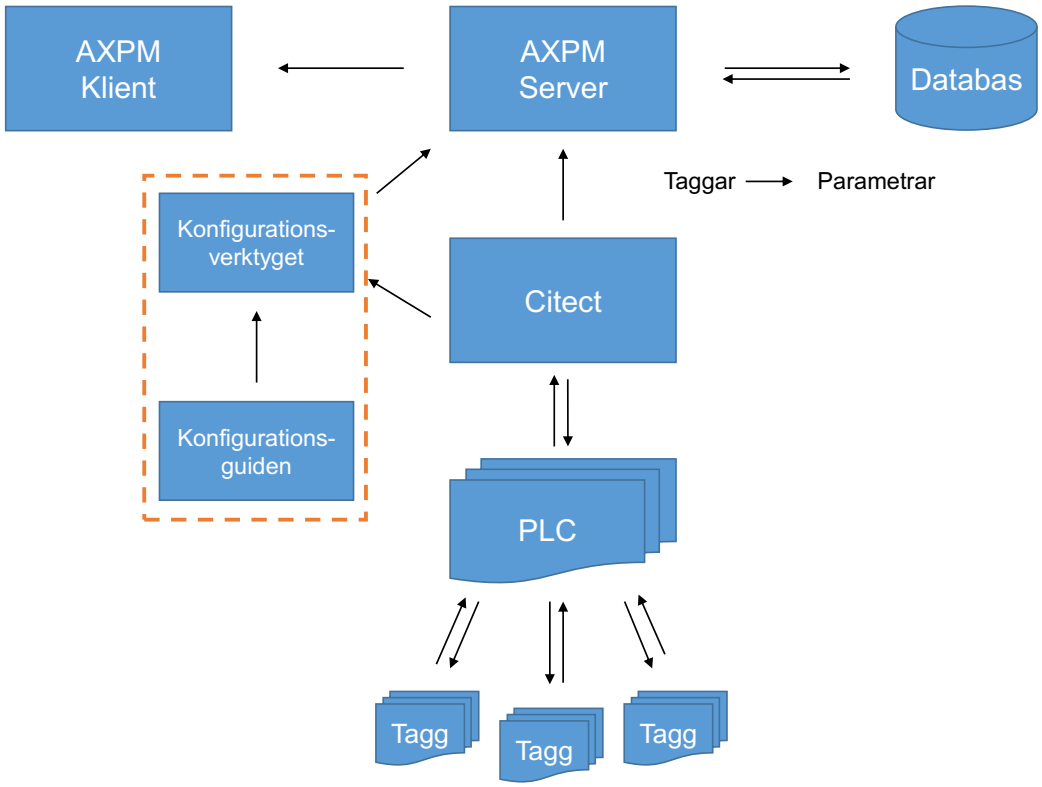
Oberoende om konfigurationsmetoden är implementerad i konfigurationsverktyget eller om den är fristående så krävs det kommunikation mellan metoden och verktyget. Viktigt är att kommunikationen mellan programmen sker på ett effektivt sätt och att det finns möjlighet att köra konfigurationsverktyget fristående för att kunna köra den som en schemalagd händelse. Dessutom är det viktigt att på ett säkert sätt kommunicera mellan konfigurationsverktyget och konfigurationsmetoden, det för att ingen data förloras och för att ha en bra felhantering.

5.4 Tilläggsarbete

Ändringar på AXPM är inte prioriterat men görs ifall tid finns och de ändringar som görs sker efter de ändringar som ökar användarvänligheten på AXPM mest. AcobiaFLUX har sedan innan en lista på rekommenderade ändringsförslag på AXPM och problemet är att välja samt utvärdera vilka ändringsförslag som kommer ge mest inverkan på programmet.

6. Val av utförande

De två stora valen i projektet har varit hur utförandet av konfigurationen ska se ut och ifall konfigurationen ska göras som fristående eller inbyggt i det nuvarande programmet.



Figur 3: Flödesschema över hur de olika programmen samarbetar med tillagd konfigurationsguide.

6.1 Fristående eller implementerat i konfigurationsverktyget

Det har funnits flera val hur man tillämpar konfigurationsmetod för konfigurationsverktyget, där de främsta alternativeten som utforskades på var att låta konfigurationsmetoden vara fristående ifrån konfigurationsverktyget och det andra var att implementera den i konfigurationsverktyget. Båda alternativen har sina fördelar och respektive nackdelar.

Att ha konfigurationsmetoden fristående ger konfigurationsverktyget mer kraft genom att verktyget kan köras själv. Man undviker komplikationer i att sammankoppla två program i ett program. Ett problem som däremot uppstår att man behöver kommunicera mellan konfigurationsmetoden och konfigurationsverktyget, vilket kan vara svårt att genomföra. Man får dessutom två program vilket kanske inte är det effektivaste, men man har också kvar möjligheten att, som ursprungligt, köra konfigurationsverktyget fristående.

Det andra alternativet med att ha konfigurationsmetoden implementerad i konfigurationsverktyget gör att man slipper ha många program, man får allt på ett och samma ställe. Kommunikationen mellan konfigurationsmetoden och verktyget är både lättare och effektivare. Problemet däremot är att man måste implementera koden i verktyget, vilket kan vara problematiskt. Dessutom är det då svårare att köra konfigurationsverktyget fristående.

Efter mycket övervägning så valdes det att ha konfigurationsmetoden fristående ifrån konfigurationsverktyget, dels för att det var lättare att genomföra men också för att ha kvar möjligheten att kunna köra ursprungliga verktyget fristående sågs vara mycket önskvärd.

6.2 Konfigurationsmetod för konfigurationsverktyget

Alternativen för konfigurationsmetod till konfigurationsverktyget har inte varit många. Det två stora alternativen har varit att metoden är i form av en installationsguide eller i form av ett inställningsfönster. Båda metoder med sina fördelar och nackdelar. Det som valdes var att låta metoden vara i form av en installationsguide, vilket är det alternativ som AcobiaFLUX sedan innan har jobbat på.

Anledningen till att välja en guide som konfigurationsmetod är att det är lätt att se till så att användaren inte glömmer någon viktigt inställning och att det dessutom är lätt att få en guide vara intuitiv. En stor del i beslutet att välja en installationsguide som konfigurationsmetod var i att AcobiaFLUX sedan tidigare har utvecklat en prototyp av en guide. Valet att som konfigurationsmetod välja en guide gör att konfigurationsmetoden i fortsättningen benämns konfigurationsguide.

7. Utförande
I detta kapitel redovisas det som har utförts. Utförandet är uppdelat i de program som det har arbetats på och därefter uppdelat på vad som skett i respektive program.

7.1 Konfigurationsguiden
Arbetet med konfigurationsguiden har varit det huvudsakliga arbetet och har tagit upp mestadels av tiden i projektet. Tillvägagångsättet för arbetet har varit att bearbeta en sida i guiden i taget tills en fungerade modell fanns. Därefter har modifieringar och förbättringar skett efter det behov som funnits. Till slut har kvalitetssäkring gjorts för att säkerställa så att inga buggar finns och att det grafiska gränssnittet är intuitivt samt ser snyggt ut. Detta sker genom att testa de olika funktionerna slumpmässigt för att se om något avvikande förekommer.

Detta avsnitt kommer behandla det som utförts på konfigurationsguiden och beskrivningarna kommer ske sida för sida genom hela guiden.

7.1.1 Startsida
Primärt hade första sidan i konfigurationsguiden två syften, det ena syftet var att informera användaren om vad guiden var till för och det andra syftet var att vara grafiskt snyggt. Senare uppkom ett till syfte, att initiera processen att konfigurera inställningarna.

En liten ändring på startsidan ifrån prototypen är att designen har uppdaterats för att vara mer konsekvent under hela guidens gång samt för att följa den grafiska profilen som AcobiaFLUX har. Därefter har lite text adderats för att informera användaren om att den ska välja att starta guiden i något av de tre lägen som finns.

Den stora ändringen ifrån prototypen är en funktion för att starta guiden i tre olika lägen; läge ett är att skapa en konfiguration helt ifrån början, läge två är att ladda in en redan existerande konfiguration och ändra på inställningar, läge tre är att ladda in en redan existerande konfiguration och köra den. Vid utförandet av denna funktion uppkom problemet i hur inställningar skulle sparas. Till detta problem diskuterades två lösningsförslag fram och vägdes mot varandra. Första lösningen var att kontinuerligt under guidens gång spara inställningarna till en fil. Fördelarna med första lösningen är att ifall konfigurationsguiden kraschar eller stängs ned, sparas inställningarna och användaren har möjlighet att fortsätta ifrån där den var sist. Andra lösningen var att temporärt lagra alla inställningar i minnet för att sedan spara dem i slutet av guiden. Den största fördelen med den andra lösningen är att den är mer resurseffektiv då guiden inte behöver att öppna samt stänga filströmmar hela tiden. I slutändan var det lösningsförslag två som valdes för att den är lättare samt mer resurseffektiv.

För att paketera och packa upp inställningarna för att lagra undan samt läsa dem ifrån en fil så behövdes det ett format som hanterade det på ett bra sätt. Till detta valdes formatet JSON för att det är både lättläst för människa samt maskin och att det är lätt att arbeta med. Ett format som undersöktes var XML eftersom att det är formatet som kommer som standard i Visual Studios, men JSON ansågs lättare att arbeta med.

```
using (StreamReader file = File.OpenText(path))
{
            JsonSerializer serializer = new JsonSerializer();
            serializer.MissingMemberHandling = MissingMemberHandling.Error;
            konfigSettings = (KonfigurationSettings)serializer.Deserialize(file,
            typeof(KonfigurationSettings));
}
```

Figur 4: Kod vid deserialisering vid öppning av en JSON-fil

När användaren ska starta guiden så får den tre val, om användaren väljer att skapa en ny konfiguration skapas ett nytt objekt där alla inställningar som användaren anger sparas i och som sedan i slutet av konfigurationsguiden serialiseras (packa ner) till JSON och sparas i en fil. Ifall användaren väljer att läsa in en fil görs detta genom att läsa in en fil med JSON-format för att sedan deserialisera (packa upp) innehållet till ett objekt. I figur 4 så kan man se hur deserialisationen ser ut i koden med hjälp av funktioner ifrån kodbiblioteket Newtonsoft.Json. Det man ser i figur 4 är att programmet först anger filströmmen till den plats inställningsfilen finns, sedan skapar den ett serialisationsobjekt som hanterar deserialiseringen, därefter deserialiseras objektet och lagras i ett annat objekt.

Vid inläsning kontrollerar guiden så att alla inställningar är rätt och att kommunikationen med databasen fungerar samt att den hittar de angivna Citect projektet. Det för att säkra att inte filen har blivit korrupt eller att något ändrat sig sedan filen först blev skapad.

Figur 5: Första sidan i guiden.

Sammanfattningsvis resulterade första sidan i en enkel sida som ger information om vad programmet behöver och att guiden tillhör AcobiaFLUX som ses på figur 5. Sidan skapar ett nytt objekt för inställningar, alternativt laddar den in inställningar till ett objekt som sedan kommer förvalda genom hela guiden. Trots att sidan inte har så många funktioner så gör den konfigurationen effektivare för användaren genom att möjliggöra för användaren att återgå till gamla konfigurationen och redigera dem.

7.1.2 SQL-databasanslutning

Anslutningen till databasen är en väsentlig del för att kunna börja konfigurera AXPM-servern. Därför så konfigurerar man anslutningen så tidigt som möjligt i guiden. Andra sidan i guiden har därför som syfte att se till så att användaren säkrar anslutningen till databasen. Dessutom har sidan som syfte att se till så att användaren väljer det system som konfigurationen skall ligga på.

I prototypen så ska användaren mata in anslutningssträngen, en textsträng som specificerar hur anslutningen ska ske till databasen själv. När man skriver in anslutningssträngen själv så är det lätt att det blir ett fel och det kan vara svårt att förstå formateringen. Därför valdes det göra det enklare för användaren genom att låta det finnas flera alternativ för användaren att välja mellan. Det användaren behöver skriva blev sökvägen till databasen. Därefter måste användaren välja vilken autentiseringsmetod, inloggningsmetod, den vill ha för att ansluta till databasen. De alternativen som ges för metod på autentisering är att köra autentisering genom att logga in som en Windows-användare eller att mata in inloggningsuppgifter för att logga in som en SQL-användare.

```
// Save the current values of the textboxes into a connectionstring
BuildConnectionString();
// Start a task to check if database is online
Task.Factory.StartNew(() => TestMethod());
```

Figur 6: Start för testa anslutningen parallellt med huvud-tråden

För att säkerhetsställa att alla uppgifter som användaren angivit är korrekta och att anslutningen är giltig testas anslutningen till databasen. För att göra det tydligt för användaren om hur man testar anslutningen så finns det en testknapp som startar testfunktionen. I figur 6 kan man se hur man anropar testfunktionen. Innan man anropar testfunktionen körs en metod för att bygga ihop anslutningssträngen, det metoden gör är att den sammanställer alla de uppgifter användaren har angivit till en sträng, det vill säga autentiseringsmetod, inloggningsuppgifter och sökväg till databasen. Ett problem uppstod vid skapandet av metoden för att bygga anslutningssträngen. Problemet var hur man ändrar anslutningssträng när man kör programmet, eftersom att programmet per automatik hämtar anslutningssträngen till databasen ifrån .NET ramverket och programmet saknar skrivrättigheter till där anslutningssträngen finns i .NET-filen under körtid. Lösningen till problemet som kan ses i figur 7 var att kringgå att programmet läser anslutningssträngen ifrån .NET och tvinga den att läsa anslutningssträngen virtuellt ifrån minnet.

```
public static void ChangeDatabase(
            string initialCatalog,
            string dataSource,
            string userId,
            string password,
            bool integratedSecuity)
{
    // Update the connectionstring, but only in the memory
    var config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
    var connectionStringsSection =
(ConnectionStringsSection)config.GetSection("connectionStrings");

    // Add values to the connectionstring with the name AXPM
    connectionStringsSection.ConnectionStrings["AXPM"].ConnectionString = "Data
Source=" + dataSource + "; Integrated Security=" + integratedSecuity + "; User
Id=" + userId + "; Password=" + password + "; Initial Catalog=" + initialCatalog +
";";
    config.Save();
    ConfigurationManager.RefreshSection("connectionStrings");

}
```

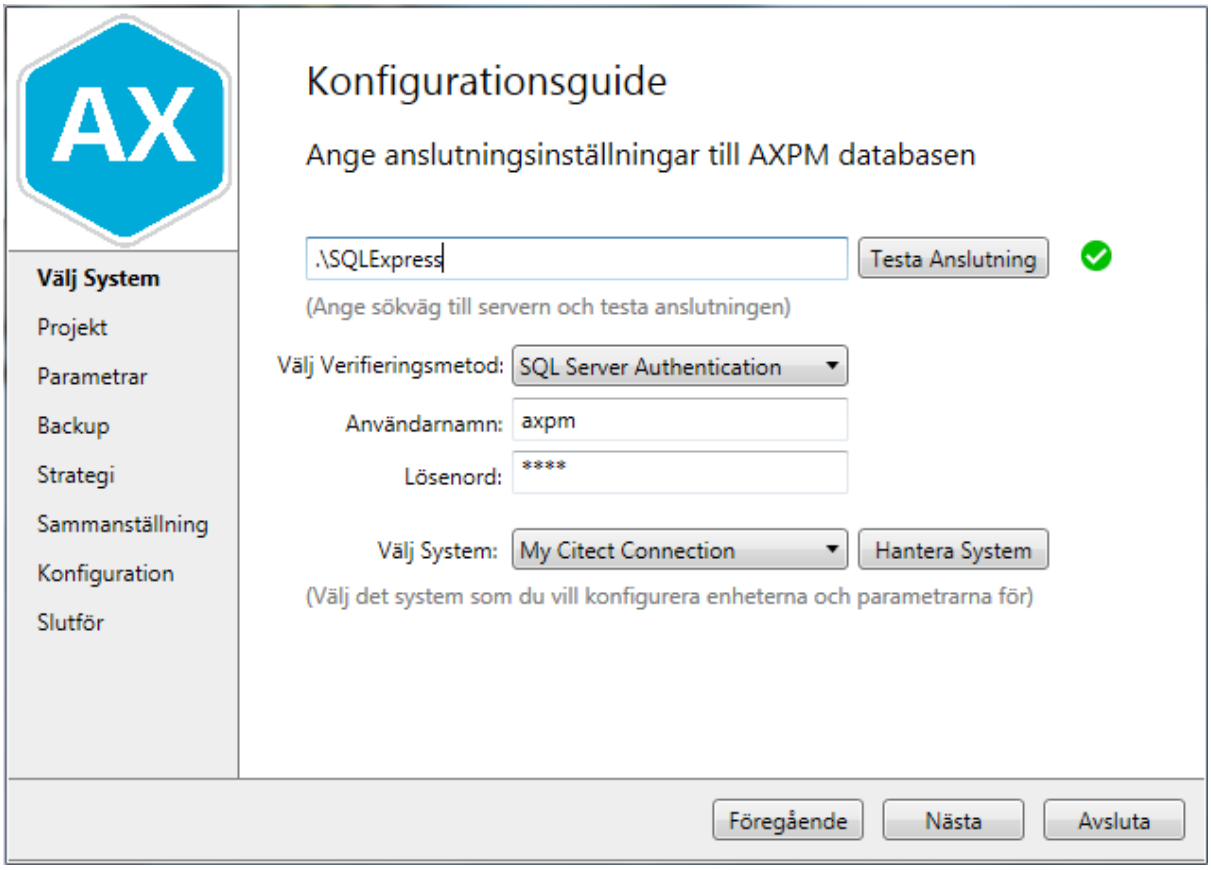
Figur 7: Kod för att tvinga programmet att använda anslutningssträngen från det virtuella minnet

Efter att anslutningssträngen är sammanställd testas det ifall den är korrekt, det genom att försöka ansluta sig till databasen. Eftersom att validering hos en anslutning är resurstung fryser sig det grafiska gränssnittet vilket kan tolkas av användaren att programmet har kraschat. Som lösning på att programmet fryser infördes användningen av flera trådar med hjälp av ”tasks”. Med ”tasks”, som möjliggörs av kodbiblioteket TPL, kan validering och det grafiska gränssnittet köras parallellt med varandra vilket innebär att programmet inte fryser sig. För att ge användaren mer feedback på att validering sker, infördes en symbol som snurrar under tiden då programmet arbetar vilket ger användaren feedback på att något händer och när arbetet är klart visas en symbol att arbetet lyckats eller misslyckats beroende på resultatet av valideringen.

```
using (var db = new AXPMContext())
{
    var queery = from connection in db.Connections orderby connection.Name select
connection;
    foreach (Connections item in queery)
    {
        serverList.Add(item);
    }
}
```

Figur 8: Sökning av existerande system i databasen

I samband med att anslutning säkras, väljs också vilket system som ska användas. System är en intern inställning i AXPM för att kunna dela upp olika säkerhetskopieringar, det för att lättare kunna sortera dem. Att söka genom systemen görs möjligt av biblioteket System Data Entity och en variant på dess klass DbContext. Figur 8 visar hur programmet hittar systemen som finns på databasen, först öppnar programmet anslutningen till där alla system finns, sedan söker den genom och hämtar alla system för att därefter lägga dem i en lista som användaren kan bläddra mellan.



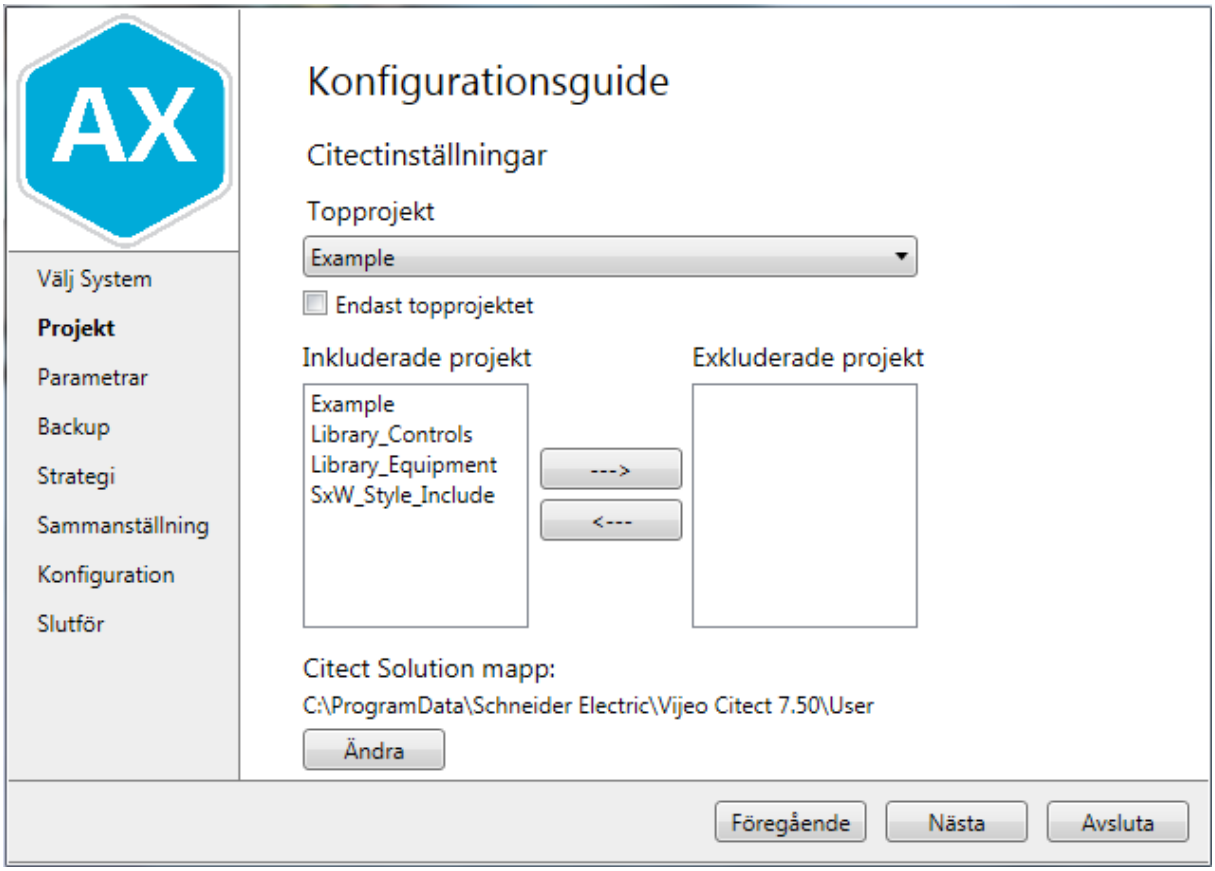
Figur 9: Sida nummer två, anslutning- och systemval.

Som en sammanfattning är andra sidan lite mer komplicerad än första sidan, både för användaren och i koden bakom. Sidan sammanställer anslutningen till SQL-databasen och väljer det system säkerhetskopieringen ska ske på, vilket syns i figur 9. Det första användaren behöver göra är att ange sökvägen till databasen och sedan ska användaren välja autentiseringsmetod. Därefter ska användaren välja det system som ska användas, om inte något system existerar finns det en metod för att skapa ett, det genom knappen ”Hantera System”. Om användaren vet sökvägen till SQL-databasen, vilket får antas, har sidan ett bra flöde och är relativ enkel att förstå. Resultatet av sidan är att den på ett enkelt sätt sammanställer anslutningen till databasen utan att användaren behöver kunna jättemycket om anslutningssträngar. Dessutom ges användaren ett par alternativ hur den vill ansluta.

7.1.3 Projekt

Parametrar sparas av Citect projektvis, där ett topprojekt kan ha flera underliggande projekt. Ett projekt kan t ex vara en hel fastighet eller ett processflöde. För att säkerhetsställa relevansen på de parametrar som säkerhetskopieras måste användaren ange vilket projekt den ska säkerhetskopiera på. Det fanns olika alternativ på hur man skulle sortera mellan projekten, ett av alternativen var som standard att ge användaren alla projekten och sen låta den få exkludera de projekt som inte var relevanta, ett annat alternativ var att låta användaren ange de projekt som skulle inkluderas. I slutändan blev det så att användaren får bläddra bland de olika topprojekten och efter användaren valt ett topprojekt får den välja vilka av de underliggande projekt som skulle inkluderas samt exkluderas. För att göra det lättförståeligt så visas alla topprojekt i en rullgardinsmeny (eng. drop down menu) och när ett topprojekt är valt så visas alla underliggande projekt i två listor under menyn. Listorna är en lista för inkluderade och en för exkluderade projekt, där kan användaren flytta de underliggande projekten mellan de två listorna för att på ett lätt sätt välja vilka projekt som ska med eller inte.

Vid sökningen efter projekt letar den efter en specifik sökväg, som anges av användaren och som går till den plats där informationen om alla projekt finns. Filen från Citect ”MASTER.DBF” innehåller informationen om projekten och information om var parametrarna hittas. Inläsningen av projekten görs genom att öppna projekten med hjälp av kodbiblioteket AX.Citect.Configuration som är skapat av AcobiaFLUX.

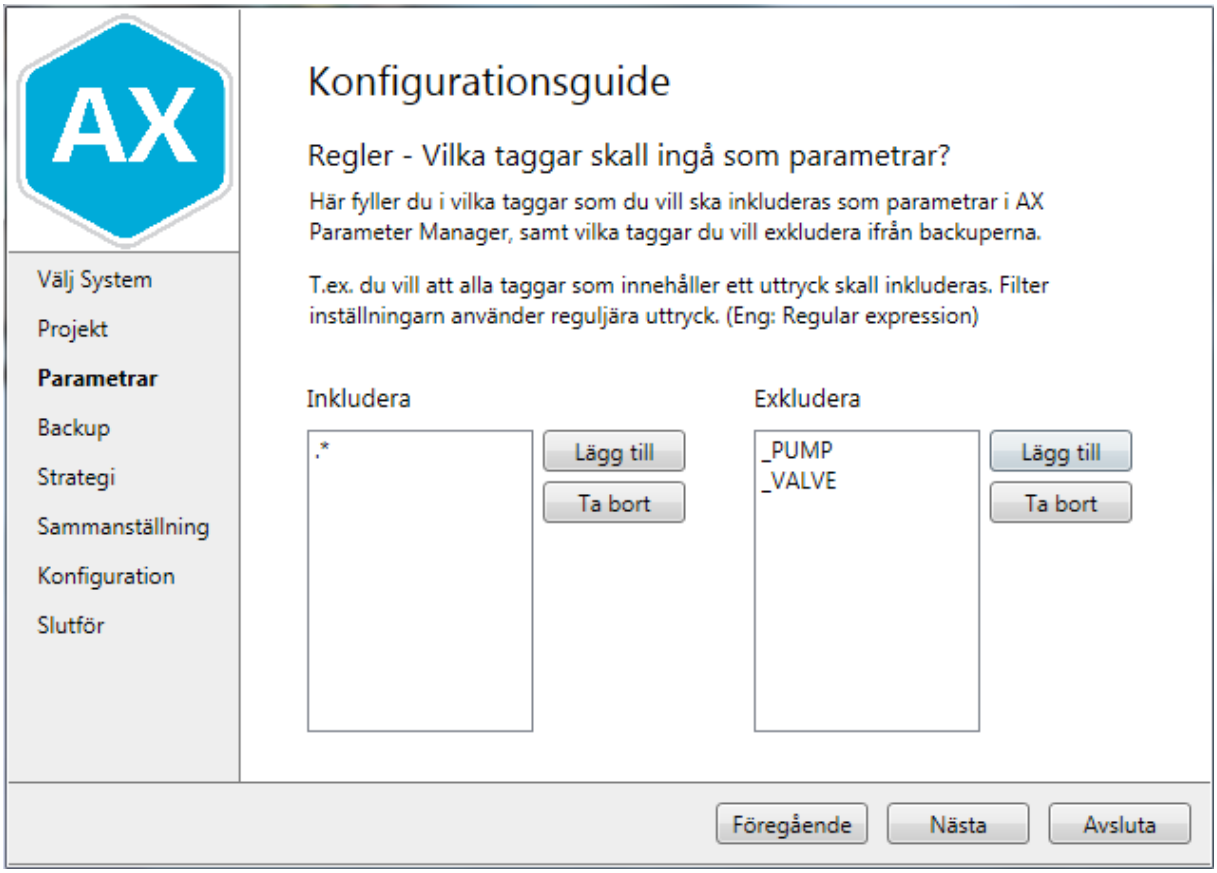


Figur 10: Sida nummer tre, Citectprojekt

I figur 10 kan man se resultatet av sidan, flödet av guide bryts lite då användaren måste välja rätt mapp till Citect. Förutom det är denna sida lättförståelig och användaren ges alla alternativ på huvudprojekt i en rullgardinsmeny. Därefter kan användaren lätt välja mellan att inkludera eller exkludera underliggande projekt.

7.1.4 Filter till parametrar

Likt vid sidan för att välja projekt ska användaren sortera vilka parametrar den vill inkludera eller exkludera. Den stora skillnaden är att användaren själv måste ange vad som ska inkluderas eller exkluderas, till skillnad ifrån projekten där alla projekt är inlästa och förvalda. Användaren kan med hjälp av en textruta ange de reguljära uttryck som den vill filtrera parametrar efter och sedan får den välja ifall parametrarna med uttrycket ska inkluderas eller exkluderas. Anledningen till att användaren själv inte väljer vilka parametrar som ska med istället för att filtrera efter reguljära uttryck är att projekt kan innehålla flera tusen parametrar och det är alldeles för ineffektivt att filtrera dem för hand. I figur 11 kan man se hur den resulterade sidan ser ut och hur några filter med reguljära uttryck kan se ut.

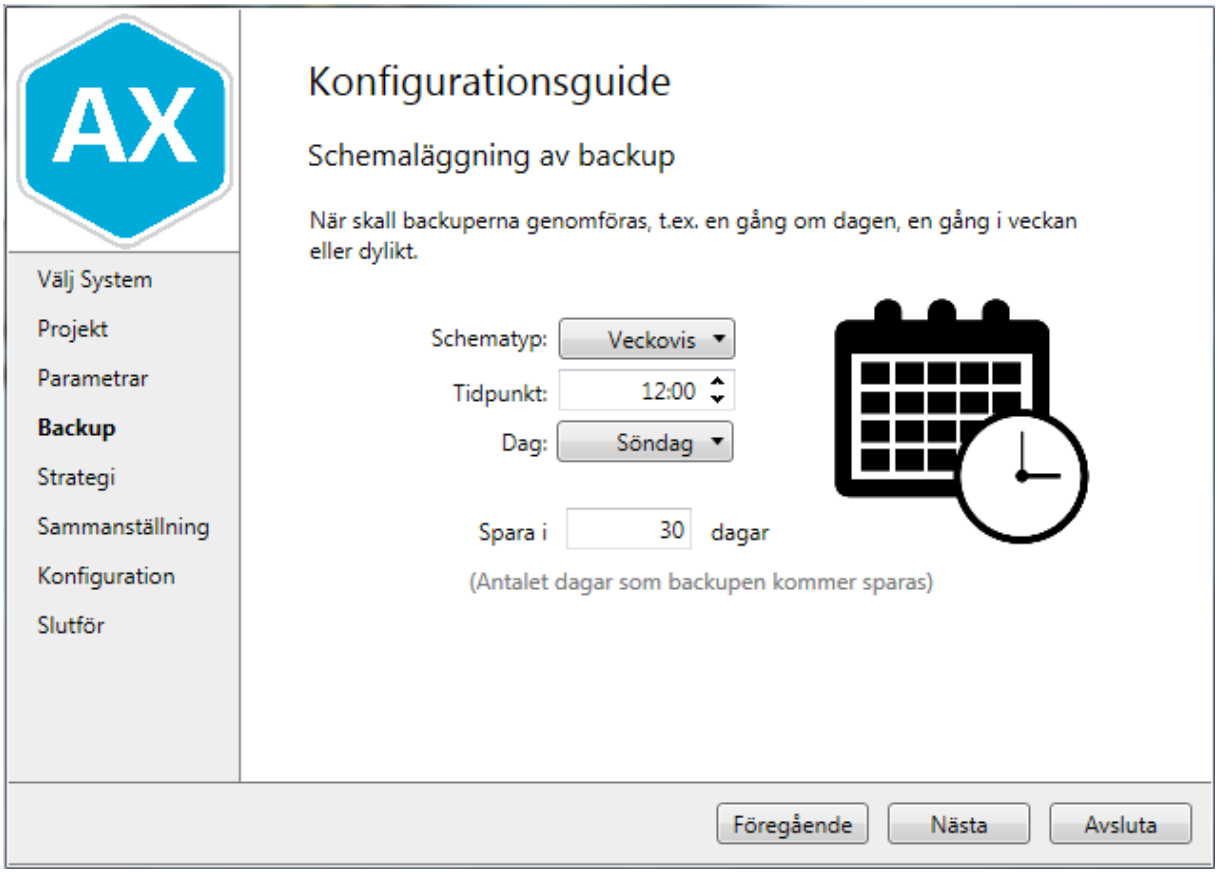


Figur 11: Sida nummer fyra, parameter-filter

Om användaren skulle missa att ange ett filter ställs frågan om alla parametrar ska inkluderas till användaren och om användaren vill det så läggs ett filter för det in. Detta för att säkerhetsställa att det missas att inkludera några parametrar.

7.1.5 Schema

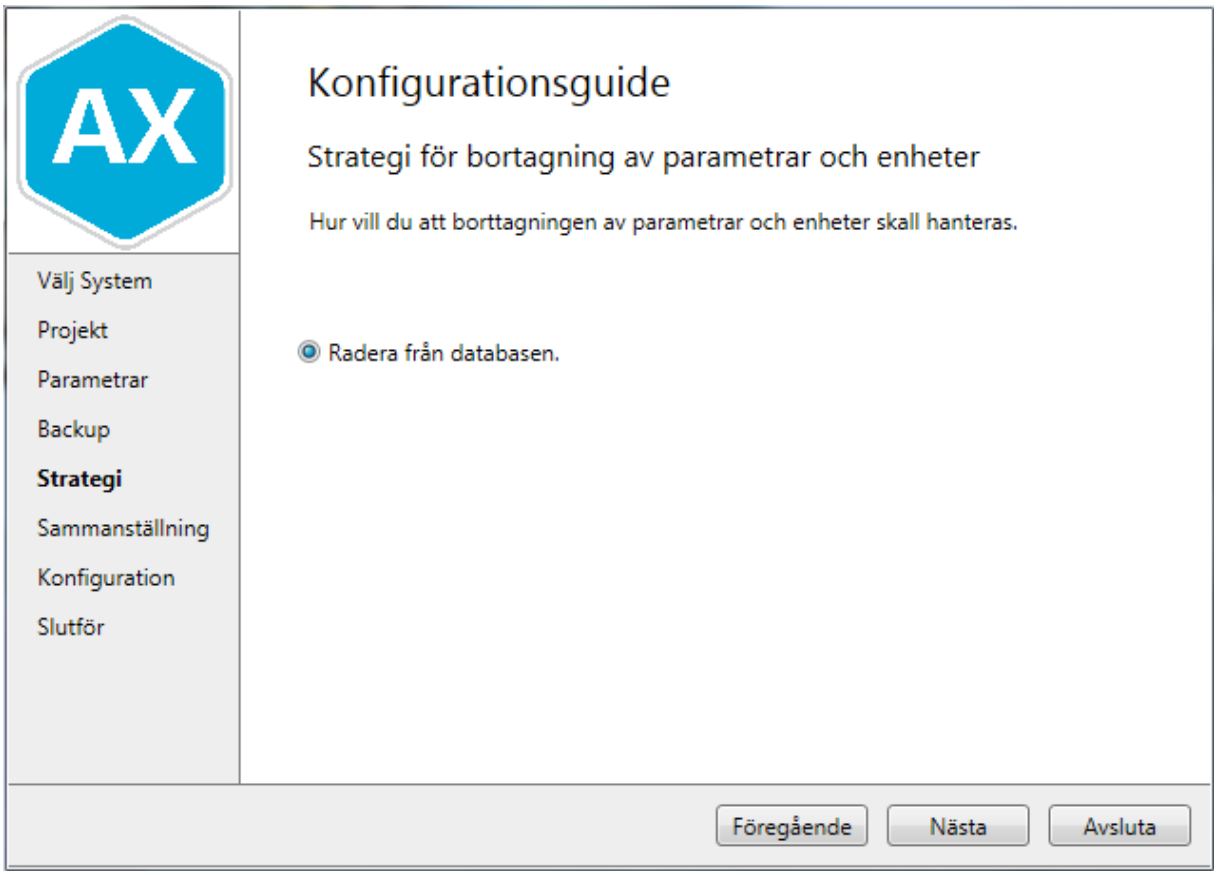
Denna sida i guiden har som syfte att få användaren att välja det schema som säkerhetskopieringen ska ske efter. Schemat bestäms av en meny där man får välja vilken tidpunkt säkerhetskopieringen ska ske, användaren får välja mellan minutvis, daglig, veckovis eller månadsvis. Med varje val visas de tillhörande inställningar specifikt för valet, i figur 12 syns valet ”veckovis” med tillhörande inställningar. Efter man valt när säkerhetskopieringen ska ske formateras det om till ett Cron-uttryck för att lätt kunna använda det senare. Man sparar dessutom i hur många dagar en säkerhetskopiering ska sparas innan den tas bort.



Figur 12: Sida nummer fem, val av schema

7.1.6 Strategi vid borttagning

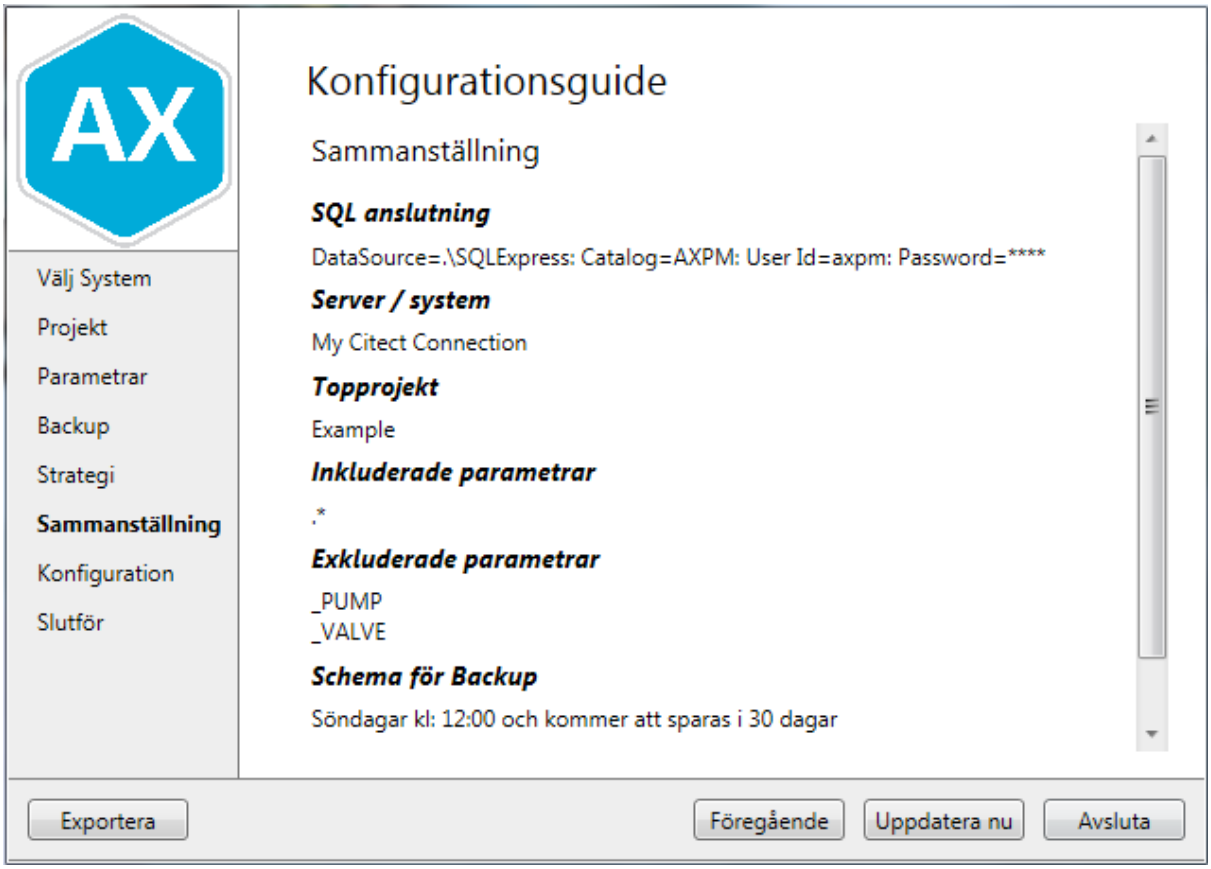
Vid förekomst av gamla parametrar behövs ett alternativ om/hur de ska tas bort från databasen. En gammal parameter är en parameter som funnits när konfigurationen skapats med tagits bort ifrån Citect i efterhand. Det kan finnas flera sätt att hantera en sådan situation och denna sida ger användaren valmöjligheten över hur den vill hantera det. Det diskuterades vilka alternativ som skulle finnas och tre alternativ togs fram; att gamla parametrar ska tas bort permanent, inte alls eller bara märkas som borttagna. Som syns i figur 13 finns endast alternativet att ta bort parametern permanent, detta beror på tidsbrist att implementera de andra funktionerna. Trots det finns möjligheten kvar att implementera de alternativen i en senare version av guiden.



Figur 13: Sida nummer sex, Strategi för borttagning

7.1.7 Sammanställning

Sammanställningssidan finns till för att användaren ska få en översikt på vad som kommer konfigureras. Det för att användaren ska kunna säkerhetsställa så att allt stämmer och att den inte har missat någon inställning. Här går det också att exportera alla inställningar ifall de ska sparas, användas på annan plats eller vill ha flera olika versioner. Exporteringsknappen kan ses i nedre vänstra hörnet i figur 14. Det är också hit användaren kommer om de valde på första sidan om de skulle ”Ladda och kör”, där användaren får säkerhetsställa att inställningarna är rätt innan den utför konfigurationen.



Figur 14: Sida nummer sju, Sammanfattning av alla inställningar

7.1.8 Konfiguration

På denna sida utförs konfigurationen av säkerhetskopieringen, här anropas konfigurationsverktyget. Det guiden gör, med hjälp av kodbiblioteket Newtonsoft.Json, är att serialisera och spara ned all information som användaren angivit till en fil. Därefter anropas konfigurationsverktyget, som i sin tur läser in filen och utför konfigurationen.

Eftersom det beslutades att konfigurationsguiden och konfigurationsverktyget skulle vara två fristående applikationer uppkom problemet att kommunicera mellan de två programmen, det för att i konfigurationsguiden få information om vad konfigurationsverktyget utför för att sedan kunna presentera det för användaren.

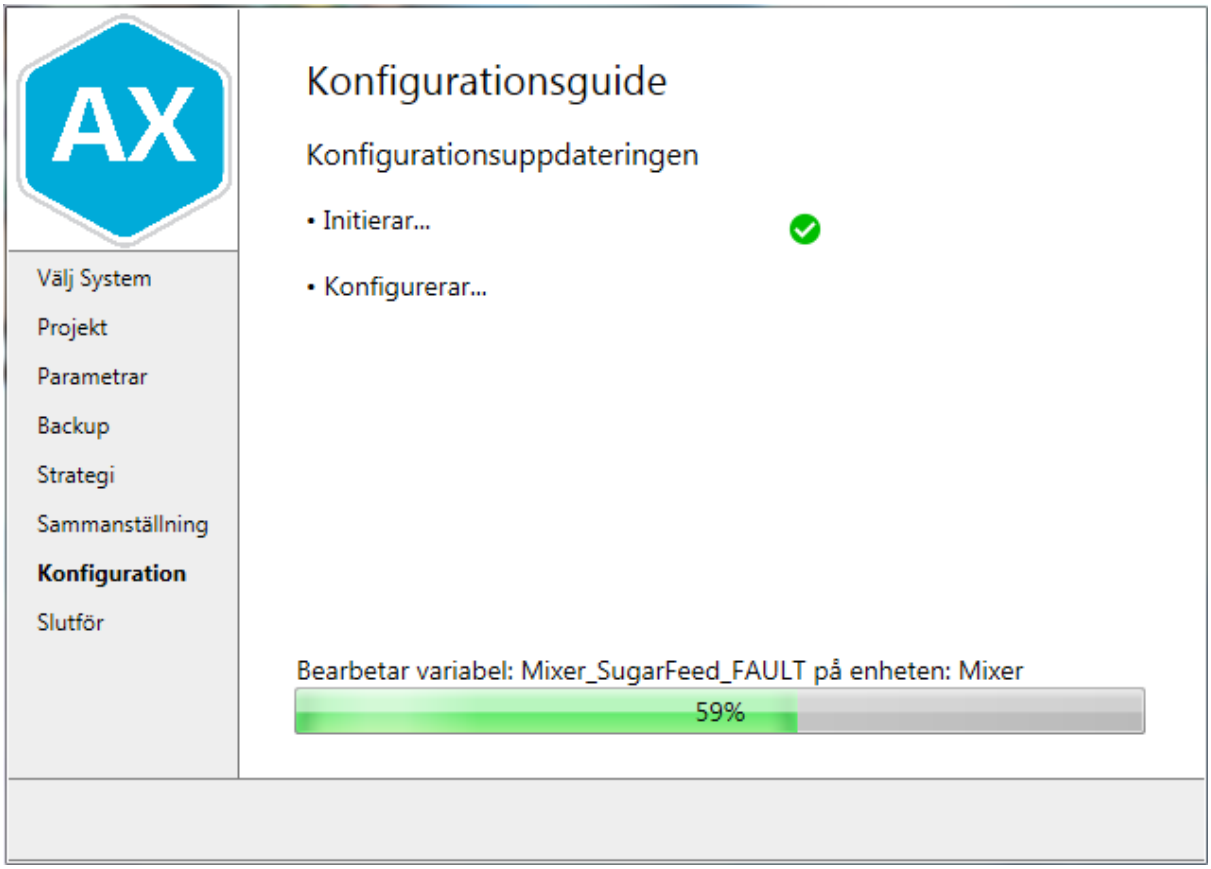
Flera IPC:er undersöktes som kommunikationskanal mellan programmen, varav flera IPC:er som var föråldrade och några som inte levde upp till de krav som ställts. Ett alternativ som fanns var minnesdelning vilket är resurseffektivt men det finns stor risk att det blir fel vid implementeringen. Minnesdelning är dessutom komplicerat att implementera då det behövs både tillgång till minnet och synkronisering mellan flera trådar i olika program. Ett annat alternativ som ansågs bra var pipelines vilket är en beprövad teknik och är enkel att implementera men är inte lika resurseffektiv som minnesdelning. I tabell 3 listas för- och nackdelar mellan minnesdelning och pipelines.

Alternativ	Fördelar	Nackdelar
Pipelines	• Enkelt att implementera • Klarar nätverkskommunikation • Relativt effektivt	• Lite äldre teknik
Minnesdelning	• Effektivt	• Komplicerad att implementera • Måste synkronisera mellan trådar

Tabell 3: För- och nackdelar med olika IPC:er

Efter övervägande valdes pipelines som metod då den ansåg nog resurssnål och säker, men främsta anledningen för att metoden är lätt att implementera. Pipelines fungerar genom att guiden initierar en pipeline-server som ska ta emot data och sedan startar guiden konfigurationsverktyget. Därefter väntar guiden tills verktyget har initierat en pipeline-klient och att en anslutning har säkerhetsställts mellan programmen. När data skickas ifrån klienten till servern, tar servern emot datan och packar upp den för att sedan presentera informationen för användaren på ett tydligt format.

Vid implementationen av pipelinen så upptäcktes det att skicka rådata utan att packa ned det i något format var ineffektivt och att det blev komplicerat att veta vad som skickades samt hur data skulle tas emot och bearbetas. Efter undersökning framkom det att JSON även här kunde implementeras för att paketera och packa upp data.

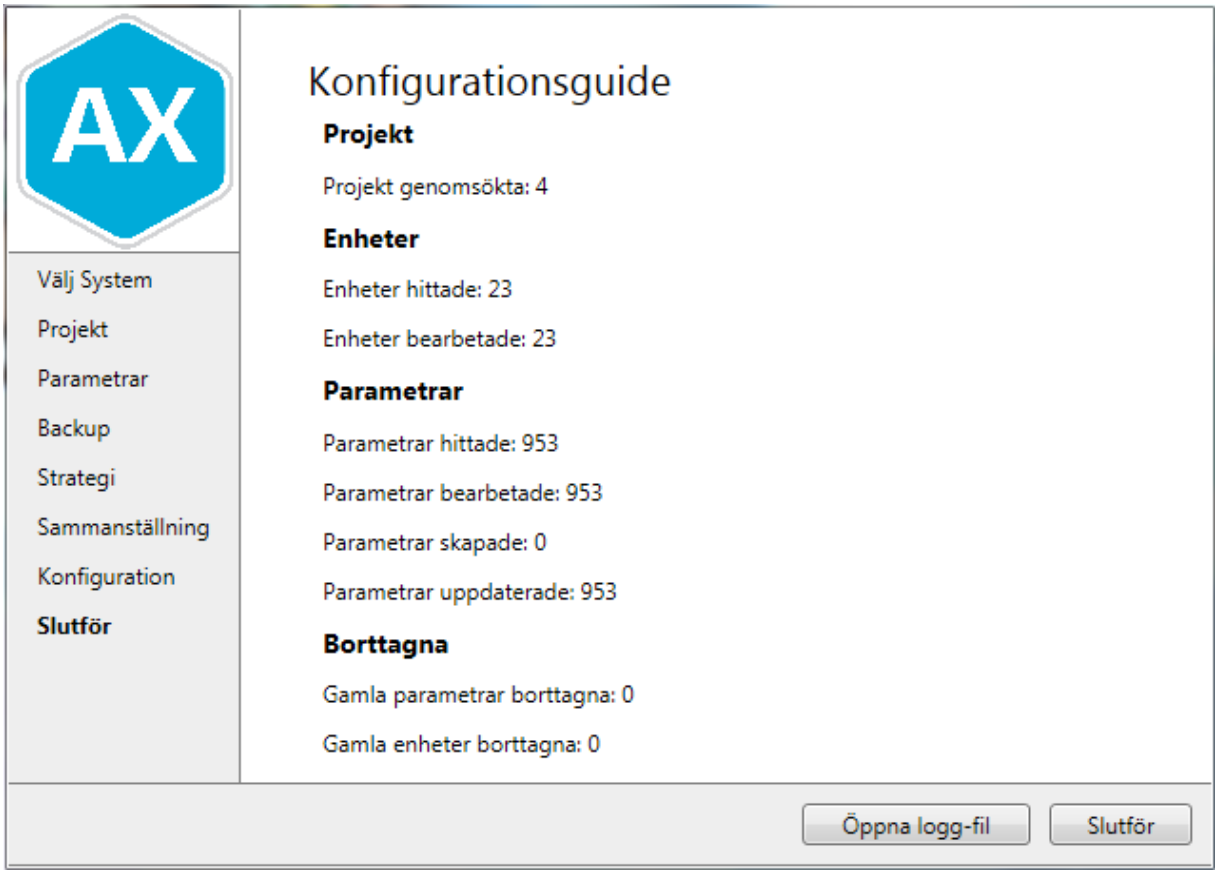


Figur 15: Sida nummer åtta, Konfigurationen

Sammanfattningsvis resulterade det i sidan som hittas i figur 15 där en förloppsindikator visar procentuellt hur mycket arbete som genomförts tillsammans med vilken parameter och enhet som i stunden bearbetas. När guiden startar verktyget kan den välja mellan att starta verktyget i bakgrunden eller i ett eget kommandotolkfönster som visar mer detaljerad information. Om man öppnar verktyget i ett kommandotolksfönster visas den information som loggas i loggfilerna kontinuerligt under konfigurationen. Hela sidans XAML-kod finns i bilaga B och C#-kod i bilaga C.

7.1.9 Resultatsida

På denna sida, som kan ses i figur 16, sammanställs resultatet, den data som skickades ifrån konfigurationsverktyget till guiden. Sidan ger också användaren möjligheten att öppna loggfilen om den skulle vilja få mer detaljerad information om vad som hände under konfigurationen.



Figur 16: Sida nummer nio, Slutrapport på verktygets arbete.

7.2 Konfigurationsverktyget

Efter analysen framgick det att konfigurationsverktyget behövde en uppdatering. Den primära uppdateringen var att få verktyget att läsa in alla inställningar ifrån en fil. En annan uppdatering var en omstrukturering i ordningen verktyget letar och bearbetar parametrar för att på ett bra sätt kunna skicka information till konfigurationsguiden om progressionen av konfigurationen. Dessutom behövdes gammal och utdaterad kod tas bort.

7.2.1 Inläsning

En stor uppdatering på konfigurationsverktyget var att verktyget istället för att läsa in standardinställningar skulle läsa in ifrån en inställningsfil genererad av konfigurationsguiden. När verktyget startar läser den in en inställningsfil med ett valt förbestämt namn och packar upp informationen som är i JSON-format.

7.2.2 Skicka data

För att få verktyget att skicka data på progressionen av konfigurationen, implementerades en pipeline-klient för att ansluta sig till konfigurationsguidens pipeline-servern. Initieringen av klienten kan ses i figur 17.

```
// Init Pipe Client
var client = new NamedPipeClientStream("AXPMPipe");
try
{
    client.Connect(5000);
}
catch (TimeoutException)
{
    Log.Info("Could not connect to Pipe server");
}
```

Figur 17: Initiering av pipeline-klienten

Därefter så skickar klienten data till servern som ses i figur 18 genom att packa ner data till JSON-format. Men eftersom konfigurationsverktyget också ska kunna köras fristående, infördes det att verktyget testar ifall det finns en anslutning till konfigurationsguiden, det görs varje gång innan något ska skickas för att säkerhetsställa att det finns en anslutning.

```
public void SendJsonToServer(ConfigurationProgress progress)
{
    string data = JsonConvert.SerializeObject(progress);
    SendToServer(data);
}

private void SendToServer(string toWrite)
{
    if (Client.IsConnected)
    {
        Writer.WriteLine(toWrite);
        Writer.Flush();
    }
}
```

Figur 18: Konfigurationsverktygets kod för att skicka data till konfigurationsguiden

För att få verktyget att kunna sammanställa informationen som ska skickas till guiden behövdes en omstrukturering av koden. Omstruktureringen var att ändra på hur och när verktyget läser in alla parametrar. Ändringen innebar att verktyget blev något långsammare, men efter tester så kunde det bestämmas att det inte var något större problem.

7.2.3 Exkludering av parametrar

Användaren har i guiden två val gällande parametrar; att inkludera eller exkludera dem. Konfigurationsverktyget har sedan tidigare bara ett läge att inkludera parametrar men behovet att kunna exkludera fanns också. Metoden för att exkludera parametrar är likt den metod som för att inkludera parametrar och båda metoder görs på samma ställe. Det metoden gör är att kontrollera ifall namnet på en parameter matcher de reguljära uttryck användaren angivit för att exkludera och om den matchar ska inte parametern läggas till.

7.2.4 Borttagning av gamla parametrar

Det undersöktes länge hur problemet med att uppdatera och ta bort gamla parametrar samtidigt skulle lösas. Lösningen som togs fram ansluter till databasen och listar alla parametrar som redan finns i databasen. Listan jämförs sedan med de parametrarna som ska läggas in i säkerhetskopieringen, om någon inte stämmer överens tas de bort som kan ses i figur 19. Eftersom att parametrar tas bort kontrolleras det även om någon enhet saknar parametrar, ifall den gör det så tas även den bort.

```
if (VariNameList.Contains(parameter.Name) == false)
{
    Log.Trace("Removing parameter '{0}' on device '{1}'", parameter.Name,
parameter.Device.Name);
    context.Parameters.Remove(parameter);
    configProgress.ParametersRemoved++;
}
```

Figur 19: Borttagning av parametrar i konfigurationsverktyget

Ett problem som uppstod vid inläsning av de gamla parametrarna var att verktyget tog med alla enheter och parametrar, även dem som inte var i samma system. Problemet löstes genom att använda enheterna direkt från anslutningen till databasen istället för databaskontexten.

8. Extra funktionalitet

Detta kapitel behandlar allt kring extra funktioner till AXPM. Det fanns många ambitioner att lägga till nya funktioner till AXPM, dels funktioner som var önskade sedan tidigare och dels funktioner som uppkom under analys samt arbetets gång. Men på grund av tidsbrist så genomfördes inga direkta extra funktioner.

En funktion som diskuterades mycket var att förenkla installationsprocessen genom att skapa en installationsfil. Det utforskades en del på ett par lösningar på detta men framför allt ”WiX Toolset”, ”Squirrel” och idén att utveckla en helt egen installationsguide. För- och nackdelar på lösningarna som togs fram går att se i tabell 4.

Alternativ	Fördelar	Nackdelar
Squirrel	• Har stöd för automatiska uppdateringar • Relativt lätt att implementera	• Svårt att lägga till egna inställningar som ska utföras under installationen • Liten frihet i användargränssnittet
WiX Toolset	• Mer inställnings möjligheter än ”Squirrel” • Frihet i användargränssnittet	• Saknar stöd för automatiska uppdateringar • Annat programmeringsspråk
Egen utvecklad	• Full frihet i inställningar och användargränssnitt • Kan göras enhetligt med AXPM guiden	• Svår utvecklat • Tidskrävande

Tabell 4: För- och nackdelar med olika installations metoder

”WiX Toolset” och ”Squirrel” är tillägg till Visual studios för att skapa en installation till applikationer. ”WiX Toolset” är utvecklat av ”FireGiant” och programmeras i XML-kod medan ”Squirrel” är utvecklat av Paul Betts och programmeras i C#. [22], [23]

Andra funktioner som diskuterades och några av de som AcobiaFLUX redan listat sedan innan. En av funktionerna var att lägga till bättre flikhantering i AXPM-klienten så att en operatör på ett lättare sätt kan ha en översikt på programmet.

9. Resultat

Arbetet resulterade i en fungerande samt en mer användarvänlig och intuitiv metod för att konfigurera AXPM-servern. Konfigurationen sker genom två delprogram, varav ena programmet ställer in de inställningar som behövs för konfigurationen och det andra som utför konfigurationen. Programmet som utför konfigurationen är en modifierad version av det redan existerande konfigurationsverktyget och för att ställa in inställningar till konfigurationen tillkom en guide som på ett intuitivt sätt leder användaren genom de inställningar som behövs ställas in.

Konfigurationsguiden är lätt att manövrera i och sidorna i guiden är lätta att förstå, användaren leds genom guiden på ett intuitivt sätt och behöver inte någon större förkunskap för att genomföra guiden. Enda sidan som är lite komplicerad för användaren är den sida där man väljer vilka projekt som säkerhetskopiering ska ske på, då användaren manuellt måste ange sökvägen till Citects användarmapp, efter det är det däremot lätt att inkludera samt exkludera projekt. Det är också lätt att lägga till och ta bort reguljära uttryck som används vid parameter identifieringen både för inkludering och exkludering. Med dessa argument går det att svara ”ja” på de tre första målen uppsatta i kapitel 1.4. Däremot på mål nummer fyra så utvecklades inga extra funktioner till programmet och inga nya funktioner kan presenteras.

10. Slutsatser & diskussioner

Inledningsvis skulle vi säga att projektet har gått bra och vi har lärt oss väldigt mycket vägen. Mycket av tiden har gått åt att lära sig förstå de programvaror och –språk som varit essentiella för projektet.

Det här projektet var vårt första projekt i C# vilket skapade lite problem och gjorde hela processen lite långsammare än var det borde. Den kunskapen vi har haft sedan innan är inom C-programmering och bara en av oss hade sedan innan läst objektorienterad programmering inom språket JAVA. Den andre tog sin fritid innan och mer tid under projektet för att lära sig objektorienterad programmering, då mer specifikt C# för att lära sig det mer ordentligt för att förstå och kunna arbeta på projektet. I uppstarten av projektet gick vi dessutom igenom en stor del av en träningsmanual till Citect för att både lära oss hur programmet fungerar och för att få ett Citect-projekt som vi kunde utföra tester på. Inlärningen gick bra, vi fick mycket kunskap och kännedom som kom till användning senare i projektet, som när konfigurationsguiden skulle söka efter Citect-projekt. Hela projektet har varit extremt lärorik både om Citect och C# för oss båda, men dessutom om SQL och .NET.

En del val av lösningsmetod har skett på grund av att vi har velat lära oss mer verktyg och har kanske inte varje gång varit den enklaste lösningen. Vid en lösning använde vi oss av JSON istället för XML, med främsta anledningen för att vi ville lära oss ett populärt och mångsidigt verktyg. Sen så blev vi rekommenderade av vår handledare Daniel att testa JSON. Att använda sig av flera trådar var något som inte var ett krav men något som vi implementerade för att vi tyckte det var intressant och att det bidrog till användarupplevelsen. Sammanfattningsvis har många val av lösningar i arbetet berott på att vi själva har velat lära oss mer, vissa lösningar har visat sig vara effektivare än vad vi trott och vissa har bidragit till att användarupplevelsen har blivit bättre

Vi hade rätt stora problem med några saker under projektets gång. Ett problem har varit att byta anslutningssträng till SQL-databasen under drift och få det att fungera, tiden som tog för att hitta en lösning på problemet syns kanske inte tydligt i resultatet men det var något som vi bearbetade länge och vi rådfrågade många av de anställda på företaget efter hjälp. Sedan tog analysen av de existerande programmen (AXPM-klienten, AXPM-servern och konfigurationsverktyget) mycket längre tid än planerat. Detta tror vi beror på att det var mycket att ta in och att det är svårare att sätta sig in ett redan existerande program och förstå skaparens bild av programmet, än att ha en egen bild och skriva från början. En annan anledning är att vi inte har analyserat ett program i denna skala innan, vilket gjorde planering svår.

I programmet finns det fortfarande vissa funktioner som vi inte har hunnit att tillämpa men som vi har tänkt på. Den funktionen vi helst hade velat tillämpa hade varit att ge fler alternativ vid borttagning av parametrar. Vi tänkte också på att lägga till ett ”Avancerat läge” på schemasidan så att användaren kunde manuellt lägga till ett Cron-uttryck. Detta eftersom om man manuellt skapar uttryck så ges man möjligheten skapa mer detaljerade och specificerade uttryck. Det fanns dessutom ambitioner om att hitta sökvägen till Citect-mappen automatiskt för att få ett bättre flöde i guiden, men det visade sig vara för komplicerat och tidskrävande att få fungerande. Det finns också funktioner som inte har testats fullt ut, som att ansluta till en databas som inte ligger lokalt på den dator som verktyget körs ifrån, det är på

grund av nätverksrestriktioner. Vi känner att det alltid går att förbättra produkter och idéer men en linje behöver dras någonstans, så vi fick vid ett tillfälle nöja oss med utvecklandet. Överlag är vi nöjda med slutprodukten och det vi har kvalitetstestat fungerar som tänkt, men det kan alltid finnas situationer som inte är testade och då kanske inte allt fungerar som det ska.

Under vissa tider under projektet, då projektet stod still på grund av olika anledningar, har vi både utforskat och analyserat en del förbättringsarbeten för att göra AXPM bättre. En förbättring som vi undersökte mycket var att utveckla en installationsguide för att göra installationsprocessen effektivare och enklare, anledningen till att vi undersökte detta berodde på att vi dels var intresserade hur utvecklingen av en installationsguide går till men dessutom tyckte vi att själva installationsprocessen var för svår. Att införa bättre flikhantering i AXPM-klienten var en förbättring vi ville göra då det var en förbättring som skulle innebära en stor skillnad för användarvänligheten hos klienten. I slutändan gjordes inga förbättringsarbeten då tiden tyvärr inte fanns.

Vad har projektet resulterat i? Vi tror att AcobiaFLUX nu har en produkt som är lättare att få ut på marknaden då det är en produkt som är betydligt enklare att implementera. Vad kan programmet ha för betydelse? Det som programmet direkt kommer göra är att det kommer vara lättare att implementera AXPM. Indirekt kan det innebära att fler företag utför säkerhetskopiering på deras system som de har justerat på länge. Om en fabrik eller annan stor anläggning förlorar sin data på något sätt (t.ex. datorkrasch eller brand) kan det ta lång tid att återgå till samma effektivitet som troligtvis fanns innan olyckan. Det kan medföra stora kostnader för miljön och samhället, men om det finns en säkerhetskopiering kan kostnaden och den negativa miljöpåverkan minskas rejält. Så man skulle kunna dra slutsatsen att indirekt kan detta program i miljösynpunkt hjälpa omvärlden.

Sammanfattningsvis har projektet gått väldigt bra och vi är nöjda med vår insats och vårt resultat. Detta trots att vi inte hann med allt vi hade ambitioner för och inlärningen samt analysen av projektet tog längre tid än förväntat. Något som vi kunde gjort bättre är att ha gjort en bättre tidsuppskattning och tidsplanering, vi skulle ha räknat med att saker kan ta längre än förväntat. Utöver det har det gått bra och vi har lärt oss program som kan vara väldigt användbara i framtiden i arbetslivet.

Referenser

[1] Nationalencyklopedin, “Automatisering.” [Online]. Available: http://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/automatisering. [Accessed: 20-May-2016].

[2] W. Bolton, *Programmable logic controllers*. Newnes, 2015.

[3] Schneider Electric, “SCADA-system.” [Online]. Available: http://www.schneider-electric.se/sites/sweden/sv/produkter-tjanster/industriell-automation/scada-system.page. [Accessed: 22-Apr-2016].

[4] Schneider Electric, “CitectSCADA.” [Online]. Available: http://www.schneider-electric.se/sv/product-range/60288-citectscada/?filter=business-1-industriell-automation&parent-category-id=5100. [Accessed: 20-May-2016].

[5] C. Schaschke, *A Dictionary of Chemical Engineering*. Oxford University Press, 2014.

[6] MSDN Microsoft, “Welcome to Visual Studio 2015.” [Online]. Available: https://msdn.microsoft.com/en-us/library/dd831853.aspx. [Accessed: 30-Mar-2016].

[7] S. Sumathi and S. Esakkirajan, *Fundamentals of Relational Database Management Systems*, vol. 47. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[8] A. Butterfield and G. E. Ngondi, Eds., *A Dictionary of Computer Science*. Oxford University Press, 2016.

[9] MSDN Microsoft, “Overview of the .NET Framework.” [Online]. Available: https://msdn.microsoft.com/sv-se/library/zw4w595w.aspx. [Accessed: 04-May-2016].

[10] MSDN Microsoft, “Getting Started with the .NET Framework.” [Online]. Available: https://msdn.microsoft.com/library/hh425099.aspx. [Accessed: 04-May-2016].

[11] MSDN Microsoft, “Introduction to the C# Language and the .NET Framework.” [Online]. Available: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx.

[12] M. MacDonald, *Pro WPF 4.5 in VB*. Apress, 2012.

[13] MSDN Microsoft, “Introduction to WPF.” [Online]. Available: https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx.

[14] Newtonsoft, “Json.NET.” [Online]. Available: http://www.newtonsoft.com/json.

[15] JSON.org, “Introducing JSON.” [Online]. Available: http://www.json.org/. [Accessed: 06-Apr-2016].

[16] MSDN Microsoft, “Task Parallelism (Task Parallel Library).” [Online]. Available: https://msdn.microsoft.com/en-us/library/dd537609(v=vs.110).aspx. [Accessed: 05-Apr-2016].

[17] MSDN Microsoft, “Interprocess Communications.” [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365574.aspx. [Accessed: 04-May-2016].

[18] MSDN Microsoft, “Pipe Operations in the .NET Framework.” [Online]. Available: https://msdn.microsoft.com/en-us/library/bb762927.aspx. [Accessed: 05-May-2016].

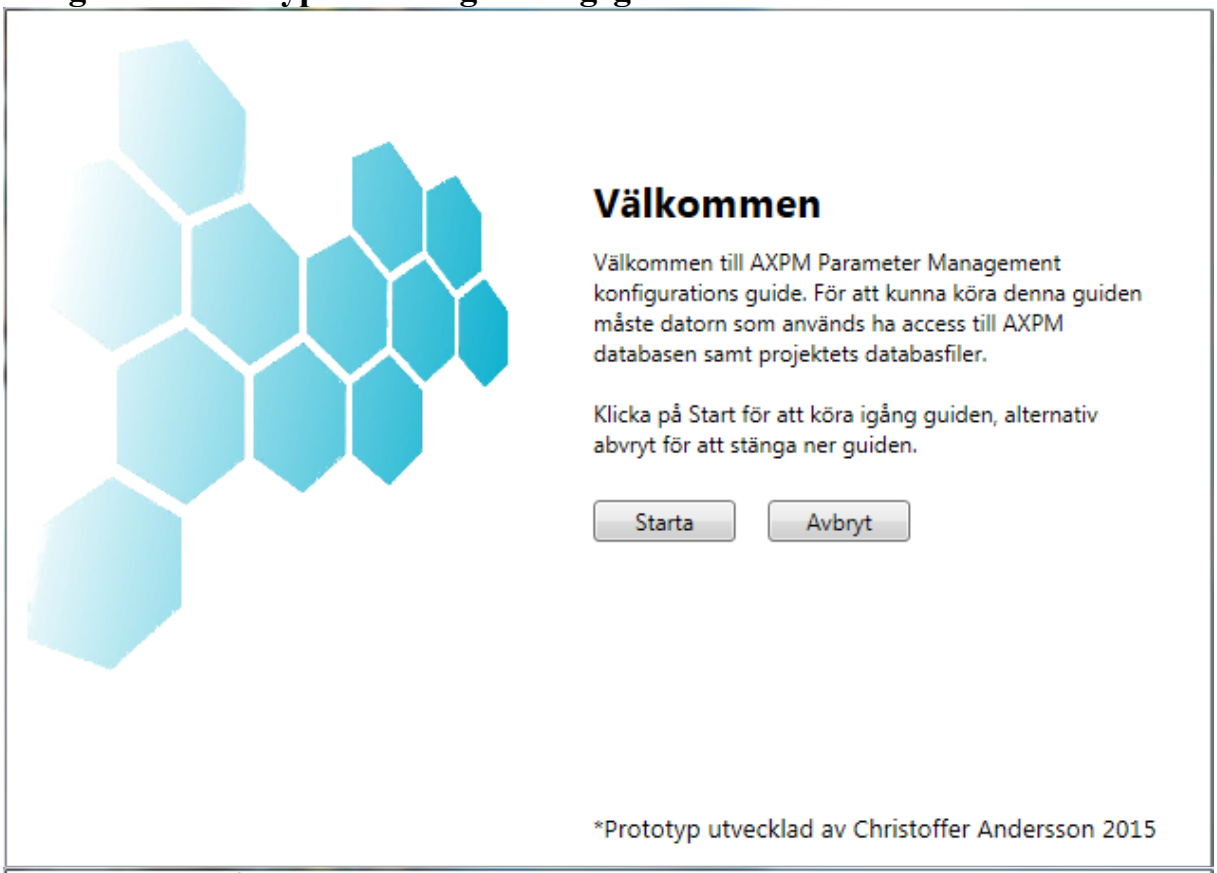
[19] MSDN Microsoft, “Regular Expression Language - Quick Reference.” [Online]. Available: https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx. [Accessed: 08-Apr-2016].

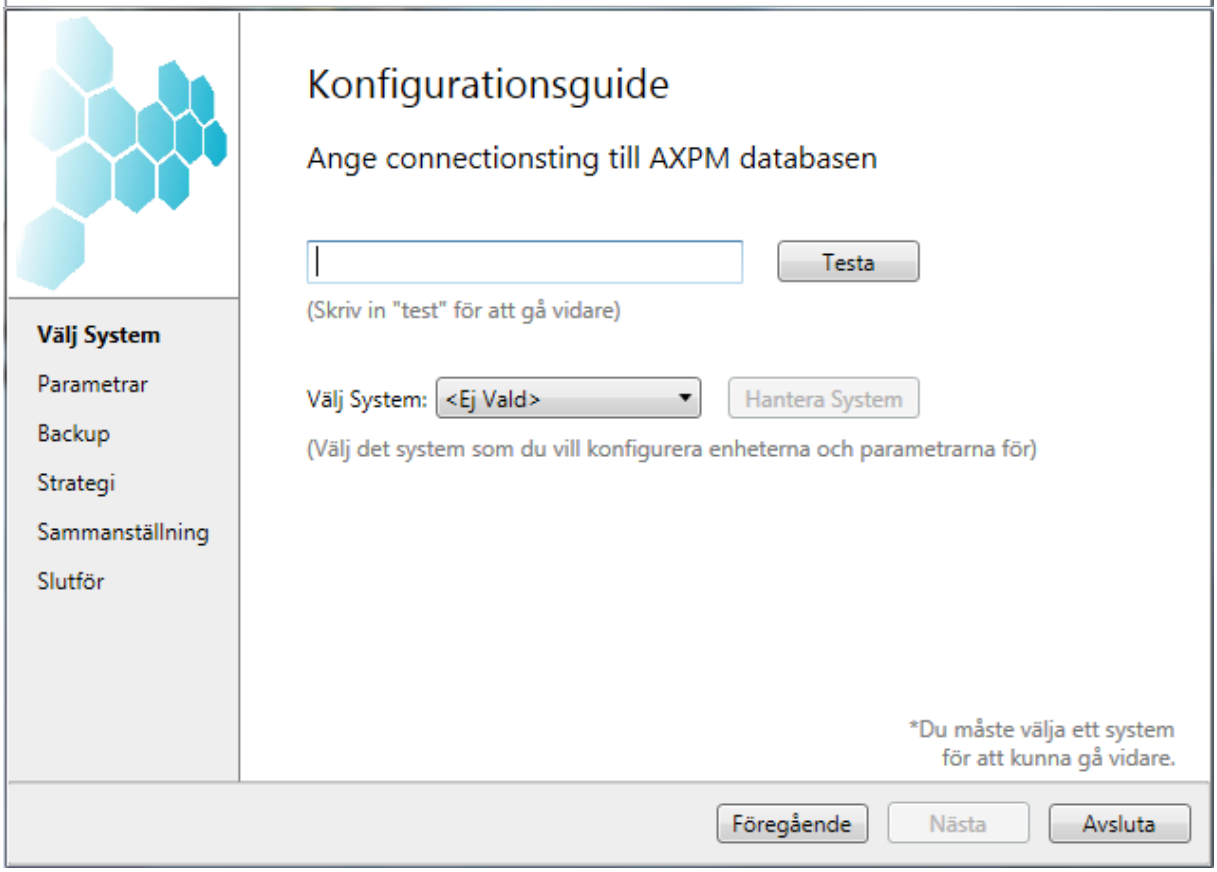
[20] C. Fabrizio, P. Huliker, and A. Prakash, “A Cron Expressions,” 2009.

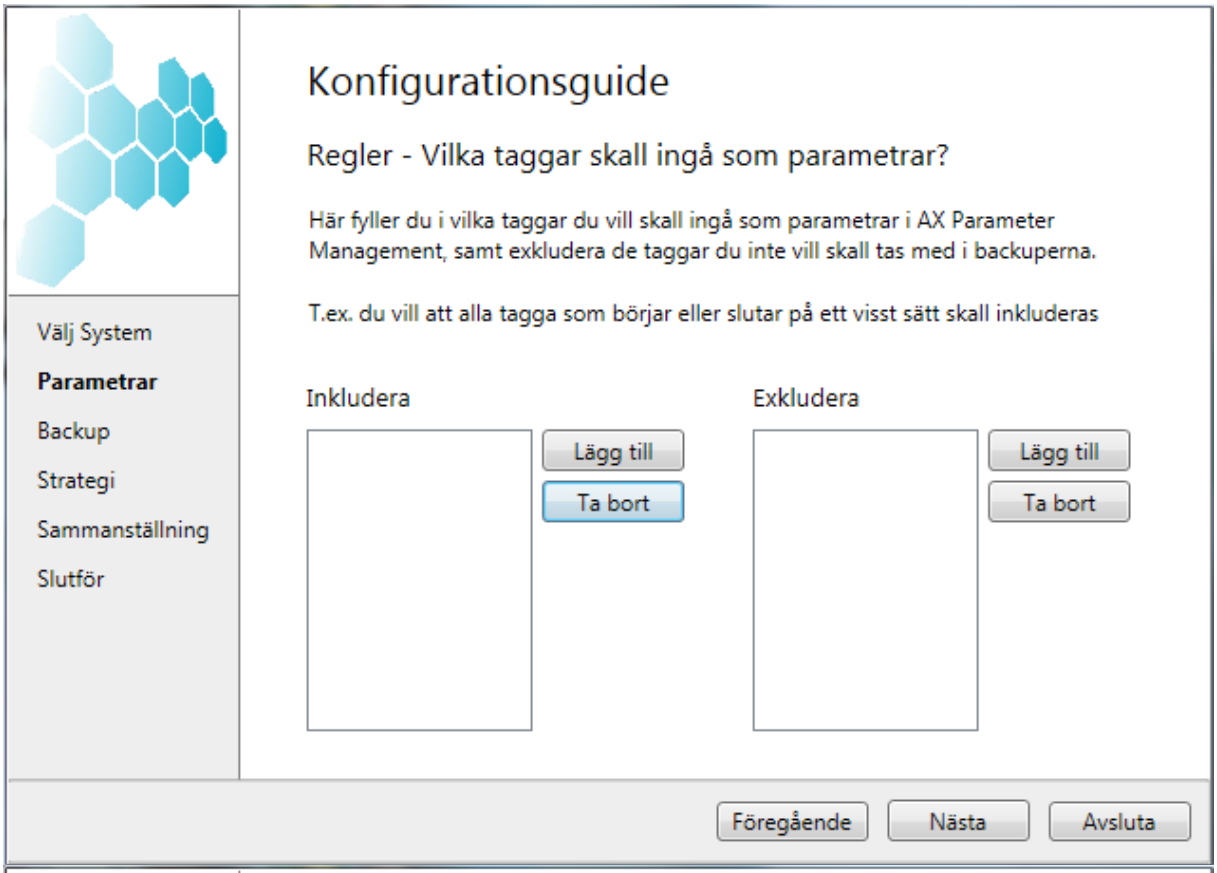
[21] Quartz, “CronTrigger Tutorial.” [Online]. Available: http://www.quartz-scheduler.org/documentation/quartz-1.x/tutorials/crontrigger. [Accessed: 15-Apr-2016].
[22] FireGiant, “WiX.” [Online]. Available: https://www.firegiant.com/wix/. [Accessed: 29-Apr-2016].
[23] P. Betts, “Squirrel.Windows,” 2016. [Online]. Available: https://github.com/Squirrel/Squirrel.Windows. [Accessed: 29-Apr-2016].

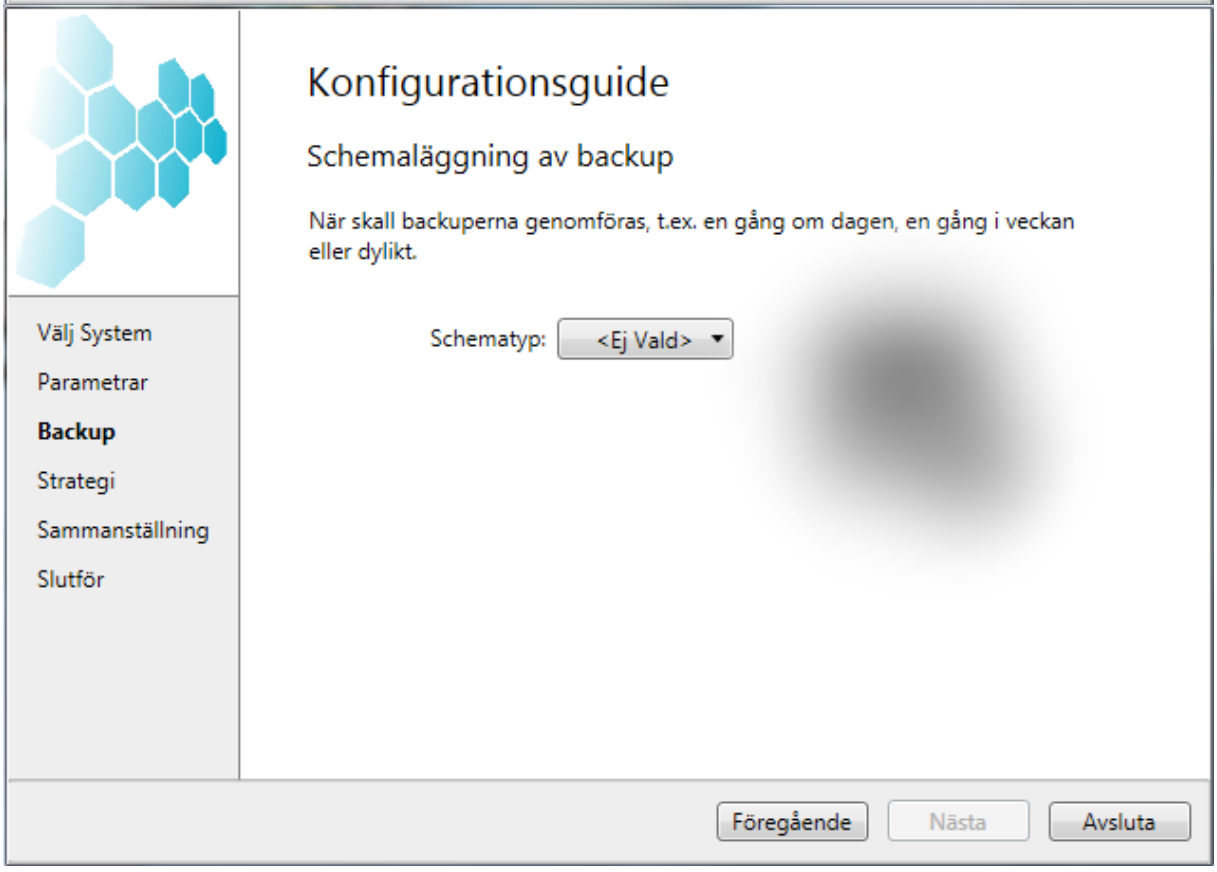
Bilagor

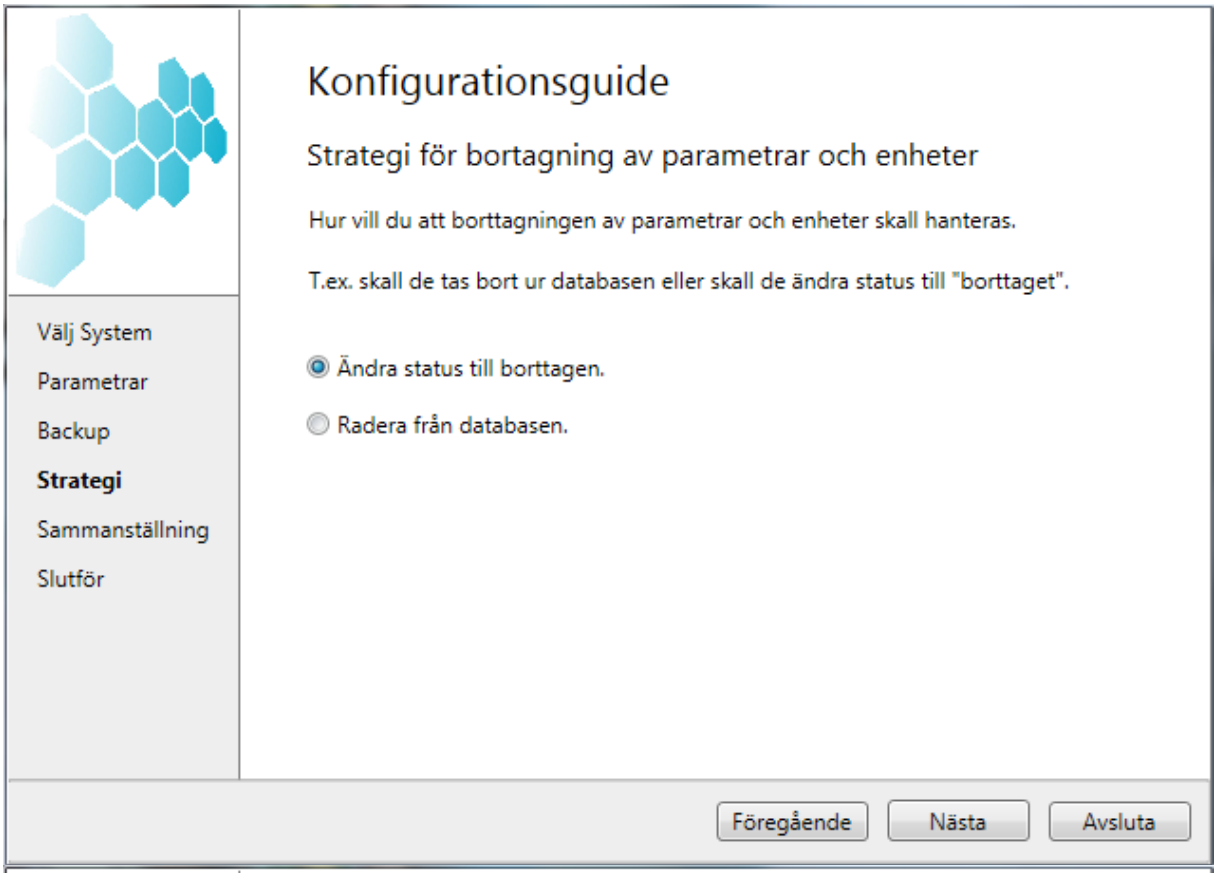
Bilaga A – Prototyp av konfigureringsguide

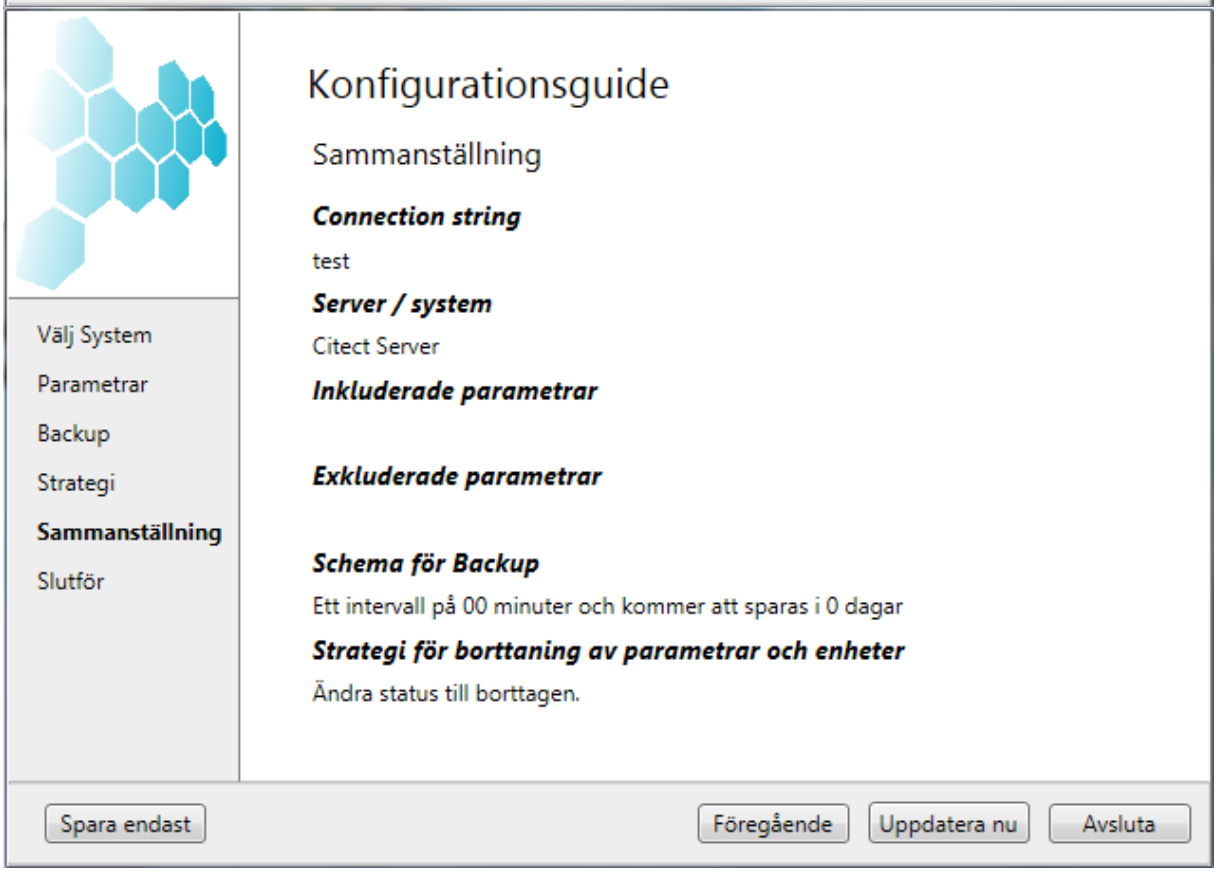


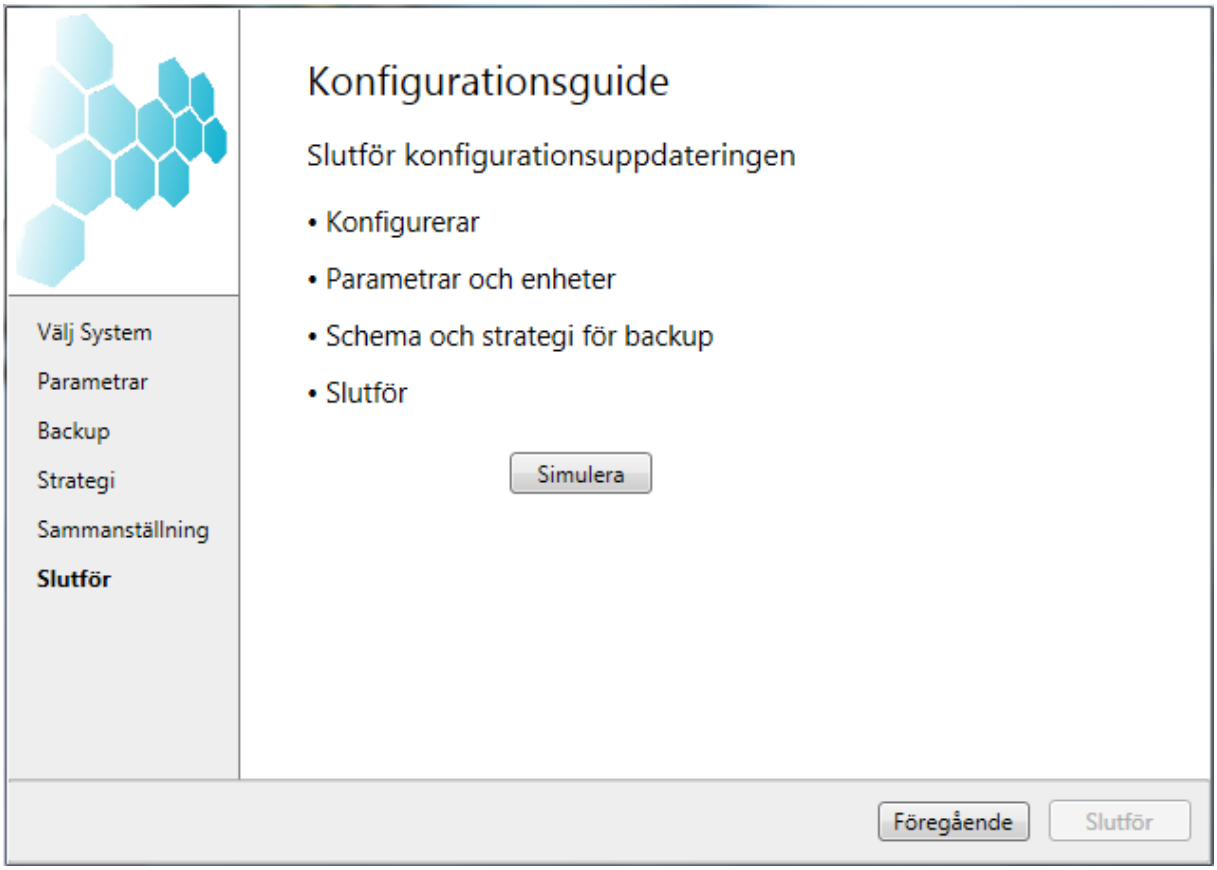



Konfigurationsguide
Regler - Vilka taggar skall ingå som parametrar?
Här fyller du i vilka taggar du vill skall ingå som parametrar i AX Parameter Management, samt exkludera de taggar du inte vill skall tas med i backuperna.
T.ex. du vill att alla tagga som börjar eller slutar på ett visst sätt skall inkluderas
Inkludera
Lägg till
Ta bort
Exkludera
Lägg till
Ta bort
Välj System
Parametrar
Backup
Strategi
Sammanställning
Slutför
Föregående
Nästa
Avsluta


Konfigurationsguide
Schemaläggning av backup
När skall backuperna genomföras, t.ex. en gång om dagen, en gång i veckan eller dylikt.
Schematyp:
<Ej Vald>
Välj System
Parametrar
Backup
Strategi
Sammanställning
Slutför
Föregående
Nästa
Avsluta


Konfigurationsguide
Strategi för bortagning av parametrar och enheter
Hur vill du att borttagningen av parametrar och enheter skall hanteras.
T.ex. skall de tas bort ur databasen eller skall de ändra status till "borttaget".
Ändra status till borttagen.
Radera från databasen.
Välj System
Parametrar
Backup
Strategi
Sammanställning
Slutför
Föregående
Nästa
Avsluta


Konfigurationsguide
Sammanställning
Connection string
test
Server / system
Citect Server
Inkluderade parametrar
Exkluderade parametrar
Schema för Backup
Ett intervall på 00 minuter och kommer att sparas i 0 dagar
Strategi för borttaning av parametrar och enheter
Ändra status till borttagen.
Välj System
Parametrar
Backup
Strategi
Sammanställning
Slutför
Spara endast
Föregående
Uppdatera nu
Avsluta


Konfigurationsguide
Slutför konfigurationsuppdateringen
• Konfigurerar
• Parametrar och enheter
• Schema och strategi för backup
• Slutför
Simulera
Välj System
Parametrar
Backup
Strategi
Sammanställning
Slutför
Föregående
Slutför

Bilaga B – XAML kod för sida sju, konfigurationssidan

```
<Page x:Class="KonfigurationsGuideWPF.Page7"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:KonfigurationsGuideWPF"
      xmlns:wfi="clr-
namespace:System.Windows.Forms.Integration;assembly=WindowsFormsIntegration"
      xmlns:winForms="clr-
namespace:System.Windows.Forms;assembly=System.Windows.Forms"
      mc:Ignorable="d"
      d:DesignHeight="{StaticResource PageHeight}" d:DesignWidth="{StaticResource
PageWidth}"
      Title="Page7" Loaded="Page_Loaded">

    <Grid Width="Auto" Height="Auto" Background="White">
        <Label x:Name="title_label" Content="Konfigurationsguide"
HorizontalAlignment="Left" Margin="152,17,0,0" VerticalAlignment="Top" Width="214"
FontSize="21.333"/>
        <Label x:Name="title2_label" Content="Konfigurationsuppdateringen"
HorizontalAlignment="Left" Margin="152,60,0,0" VerticalAlignment="Top"
FontSize="16"/>
        <Label x:Name="label_init" Content="• Initierar..."
HorizontalAlignment="Left" Margin="152,96,0,0" VerticalAlignment="Top"
FontSize="14"/>
        <Label x:Name="label_konfig" Content="• Konfigurerar..."
HorizontalAlignment="Left" Margin="152,131,0,0" VerticalAlignment="Top"
FontSize="14" Visibility="Hidden"/>

        <Border Height="60" Width="{Binding ElementName=textblock_progress,
Path=Width}" Margin="152,298,0,0" VerticalAlignment="Top"
HorizontalAlignment="Left">
            <TextBlock x:Name="textblock_progress" Text="Arbetar..."
VerticalAlignment="Bottom" HorizontalAlignment="Left" FontSize="14" Width="449"
TextWrapping="Wrap"/>
        </Border>

        <ProgressBar x:Name="progressBar" Minimum="0" Height="25" Width="450"
VerticalAlignment="Top" HorizontalAlignment="Left" Margin="151,358,0,0"
IsIndeterminate="True"/>
        <!--Border to keep percent in middle of progress bar-->
        <Border Height="{Binding ElementName=progressBar, Path=Height}"
Width="{Binding ElementName=progressBar, Path=Width}" Margin="{Binding
ElementName=progressBar, Path=Margin}" VerticalAlignment="Top"
HorizontalAlignment="Left">
            <TextBlock x:Name="textblock_percent" Text=""
VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="14"
Margin="0,0,0,0"/>
        </Border>
        <Label x:Name="label_finish" Content="• Slutför..."
HorizontalAlignment="Left" Margin="152,166,0,0" VerticalAlignment="Top"
FontSize="14" Visibility="Hidden"/>
        <wfi:WindowsFormsHost x:Name="winhost1" Margin="411,106,201,324"
Width="20" Height="20" Foreground="{x:Null}">
            <winForms:PictureBox x:Name="pictureBoxLoading1"
SizeMode="StretchImage">
            </winForms:PictureBox>
        </wfi:WindowsFormsHost>
```

```
        <wfi:WindowsFormsHost x:Name="winhost2" Margin="411,141,201,289"
Width="20" Height="20" Foreground="{x:Null}">
            <winForms:PictureBox x:Name="pictureBoxLoading2"
SizeMode="StretchImage">
            </winForms:PictureBox>
        </wfi:WindowsFormsHost>
        <wfi:WindowsFormsHost x:Name="winhost3" Margin="411,176,201,254"
Width="20" Height="20" Foreground="{x:Null}">
            <winForms:PictureBox x:Name="pictureBoxLoading3"
SizeMode="StretchImage">
            </winForms:PictureBox>
        </wfi:WindowsFormsHost>
        <Border BorderThickness="0,0,1,0" HorizontalAlignment="Left" Width="122"
Margin="0,0,0,44" Background="#FFEEEEEE" BorderBrush="Gray">
            <Grid>
                <Image x:Name="image" HorizontalAlignment="Left" Height="128"
VerticalAlignment="Top" Width="121"
Source="/KonfigurationsGuideWPF;component/Images/Hexagon_AX.png"
RenderTransformOrigin="0.532,0.221"/>
                <Border BorderBrush="Gray" BorderThickness="0,0,0,1"
HorizontalAlignment="Left" Height="128" VerticalAlignment="Top" Width="121"/>
                <DockPanel LastChildFill="False" Background="#FFEEEEEE"
Margin="0,128,0,0" >
                    <Label x:Name="progress1_label" Content="Välj System"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress2_label" Content="Projekt"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress3_label" Content="Parametrar"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress4_label" Content="Backup"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress5_label" Content="Strategi"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress6_label" Content="Sammanställning"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                    <Label x:Name="progress7_label" Content="Konfiguration"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top" FontWeight="Bold"/>
                    <Label x:Name="progress8_label" Content="Slutför"
HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Width="101"
DockPanel.Dock="Top"/>
                </DockPanel>
            </Grid>
        </Border>
        <Border BorderBrush="Gray" BorderThickness="1" Height="44"
VerticalAlignment="Bottom">
            <DockPanel Height="42" LastChildFill="False" Background="#FFEEEEEE"
Margin="-1,0" VerticalAlignment="Bottom">
            </DockPanel>
        </Border>
    </Grid>
</Page>
```

Bilaga C – C# kod för sida sju, konfigurationssidan

```
using System;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using Newtonsoft.Json;
using System.IO.Pipes;
using System.Diagnostics;

namespace KonfigurationsGuideWPF
{
    public partial class Page7 : Page
    {
        GuideWindow guideWindow;
        KonfigurationSettings konfigSettings;
        Page pagePrev;
        Page8 page8;
        ConfigurationProgress progress = new ConfigurationProgress();

        public Page7(GuideWindow window, Page prev, KonfigurationSettings konfig)
        {
            InitializeComponent();
            guideWindow = window;
            konfigSettings = konfig;
            pagePrev = prev;
        }

        private void previous_button_Click(object sender, RoutedEventArgs e)
        {
            //  Navigates to previous page.
            guideWindow.frame.Navigate(pagePrev);
        }

        private void StartWork()
        {
            StartPipeServer();
            ProcessStartInfo start = new ProcessStartInfo();
            start.FileName = Properties.Settings.Default.PathToTool;
            if (Properties.Settings.Default.HideToolCMD)
            {
                start.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
            }

            try
            {
                Process client = Process.Start(start);
            }
            catch (Exception f)
            {

                page8 = new Page8(guideWindow, progress, true);
                page8.textBlock.Text = "Ett fel uppstod! Starta om guiden och
testa igen. Fel meddelande: " + f;
                guideWindow.frame.Navigate(page8);
            }
        }
```

```
        private void StartPipeServer()
        {
            var task = Task.Factory.StartNew(() =>
            {
            NamedPipeServerStream server = new NamedPipeServerStream("AXPMPipe");

            try
            {
                server.WaitForConnection();
                StreamReader reader = new StreamReader(server);
                StreamWriter writer = new StreamWriter(server);

                var onlyOnce = true;
                while (true)
                {
                    if (server.IsConnected)
                    {
                        var dataFromTool = reader.ReadLine();
                        if (dataFromTool != null)
                        {
                            progress =
JsonConvert.DeserializeObject<ConfigurationProgress>(dataFromTool);
                        }
                            if (progress != null)
                            {
                                if (progress.SearchFinished == false)
                                {
                                    textblock_progress.Dispatcher.Invoke(() =>
(textblock_progress.Text = progress.Info));
                                }
                                else if (progress.SearchFinished &
(progress.UpdateFinished == false))
                                {
                                    if (onlyOnce)
                                    {

this.textblock_progress.Dispatcher.Invoke(() => (textblock_progress.Visibility =
Visibility.Visible));
                                        this.label_konfig.Dispatcher.Invoke(() =>
(label_konfig.Visibility = Visibility.Visible));
                                        this.winhost1.Dispatcher.Invoke(() =>
(pictureBoxLoading1.Image =
System.Drawing.Image.FromFile(@"Images/CheckMark1.png")));
                                        this.winhost2.Dispatcher.Invoke(() =>
(pictureBoxLoading2.Image =
System.Drawing.Image.FromFile(@"Images/loading.gif")));
                                        this.progressBar.Dispatcher.Invoke(() =>
(progressBar.IsIndeterminate = false));
                                        this.progressBar.Dispatcher.Invoke(() =>
(progressBar.Maximum = progress.ParametersSaved));

                                        onlyOnce = false;
                                    }
                                    var percent =
Math.Round(((double)progress.ParametersProcessed /
(double)progress.ParametersSaved) * 100);
                                    this.progressBar.Dispatcher.Invoke(() =>
(progressBar.Value = progress.ParametersProcessed));
```

```
                                    this.textblock_progress.Dispatcher.Invoke(()
=> (textblock_progress.Text = progress.Info));
                                    this.textblock_percent.Dispatcher.Invoke(() =>
(textblock_percent.Text = (percent + "%")));
                                }
                                else if (progress.SearchFinished &
progress.UpdateFinished)
                                {
                                    this.progressBar.Dispatcher.Invoke(() =>
(progressBar.IsIndeterminate = true));
                                    this.textblock_progress.Dispatcher.Invoke(()
=> (textblock_progress.Text = "Tar bort gamla parametrar och enheter"));
                                    this.textblock_percent.Dispatcher.Invoke(() =>
(textblock_percent.Text = (""));
                                }
                            }

                            if (progress.IsFinished == true)
                            {
                                break;
                            }
                        }
                        else
                        {
                            this.winhost1.Dispatcher.Invoke(() =>
(pictureBoxLoading1.Image =
System.Drawing.Image.FromFile(@"Images/RedCross.png")));
                            this.winhost2.Dispatcher.Invoke(() =>
(pictureBoxLoading2.Image =
System.Drawing.Image.FromFile(@"Images/RedCross.png")));
                            this.winhost3.Dispatcher.Invoke(() =>
(pictureBoxLoading3.Image =
System.Drawing.Image.FromFile(@"Images/RedCross.png")));
                            throw new IOException("Lost connection with
configuration tool.");
                        }
                    }
                    server.Close();
                    this.winhost2.Dispatcher.Invoke(() => pictureBoxLoading2.Image
= System.Drawing.Image.FromFile(@"Images/CheckMark1.png"));
                    this.winhost3.Dispatcher.Invoke(() =>
(pictureBoxLoading3.Image =
System.Drawing.Image.FromFile(@"Images/loading.gif")));
                    return true;
                }
                catch (Exception f)
                {
                    MessageBox.Show("Something went wrong. Message: " + f);
                    server.Close();

                    this.Dispatcher.Invoke(() => { page8 = new Page8(guideWindow,
progress, true); });
                    page8.Dispatcher.Invoke(() => (page8.textBlock.Text = "Ett fel
uppstod! Starta om guiden igen och testa igen. Fel meddelande: " + f));
                    guideWindow.Dispatcher.Invoke(() =>
(guideWindow.frame.Navigate(page8)));
                    return false;
                }
```

```
            }).ContinueWith((t) =>
            {
                if (t.Result)
                {
                    this.Dispatcher.Invoke(() => { page8 = new Page8(guideWindow,
progress, false); });
                    this.winhost3.Dispatcher.Invoke(() => pictureBoxLoading3.Image
= System.Drawing.Image.FromFile(@"Images/CheckMark1.png"));
                    guideWindow.Dispatcher.Invoke(() =>
(guideWindow.frame.Navigate(page8)));
                }
            });
        }

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            pictureBoxLoading1.Image =
System.Drawing.Image.FromFile(@"Images/loading.gif");
            Save_Config();
            StartWork();
        }

        private void Save_Config()
        {
            try
            {
                using (StreamWriter file =
File.CreateText(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicati
onData) + Properties.Settings.Default.PathToConfig))
                {
                    JsonSerializer serializer = new JsonSerializer();
                    serializer.Serialize(file, konfigSettings);
                }
            }
            catch (Exception f)
            {
                System.Diagnostics.Debug.Write(f);
            }
        }
    }
}
```