CHALMERS

# Centralsystem för Informationsskärmar

Examensarbete inom Högskoleingenjörsprogrammet i mekatronik

Robin Bandgren
Viktor Engström

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

EXAMENSARBETE

# Centralsystem för Informationsskärmar

Robin Bandgren
Viktor Engström

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2015

**Centralsystem för informationsskärmar**

Robin Bandgren
Viktor Engström



Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

## Sammanfattning

Informationsskärmsbranschen växer fortfarande och behovet av mer avancerade produkter ökar ständigt. Vid växling mellan företagsinformation, till att presentera larminformation från exempelvis fastighetslarm eller produktionslarm, så är marknaden relativt outforskad. Det här projektet går ut på att undersöka marknaden efter befintliga produkter för att se om det redan existerar en produkt med dessa egenskaper. Om inte det finns ska en egen prototyp tillverkas för informationsskärmar. Skärmarna skall bläddra mellan internnyheter, lunchmenyer eller välkomstmeddelanden, tills ett larm sker. Skärmarna slår då snabbt över och visar en bild med larminformation istället. Det skulle exempelvis kunna vara en bild på var i fastigheten larmet har gått, eller på vilken station i en produktion det är stopp.

Lösningen som togs fram var en prototyp. Denna prototyp fungerar som så att bilder automatiskt hämtas från mappar på en serverenhet och presenteras i ett bildspel på en webbsida. Om det skulle gå ett larm så placeras informationen från systemövervakningsprogrammet i en särskild sökväg, som sedan prioriteras av informationshämtningssystemet och läggs upp på skärmarna. Alla programvaror och system som krävs ligger på servern där informationsskärmssystemet sköts ifrån. Skärmvisningsytan går att öppna via en vanlig webbläsare eller via applikationer på smarta produkter så som Smart-phones eller Smart-TV.

## Abstract

The information display industry is still growing and the demand is constantly growing. However, when it comes to the possibility to be able to switch between internal company news or lunch menus, to information about ongoing alarms in a building, or production process, the market is unexplored. The project aims to solve that issue and to make it possible for information displays to quickly switch between regular everyday information to a, for example, fire alarm information image, with information about where the alarm is and the fire escape routes.

The solution is to automatically get images or files that are to be displayed on the screens from folders on a server unit and upload them in to a webpage. If an alarm would go off in the system the process supervising program will upload information in to a higher priority folder. From there the information display system will get the information and put it up on the display, and in the same instance remove the regular information slideshow. The information screen is possible to open in a regular web browser or in smart-products like a phone or TV through an application.

## Förord

Vi skulle vilja tacka alla på AcobiaFLUX för all hjälp och stöd under projektets gång och rikta ett särskilt tack till Andreas Karlsson som har hjälpt oss med allt från mjukvaror till att varit med och bollat idéer. Vi skulle även vilja tacka vår handledare Rolf Snedsböl på Chalmers Tekniska Högskola för all hjälp med rapportskrivandet. Ett stort tack även till alla andra som på ett eller annat sätt har hjälpt till under detta projekt.

## Begrepp

| | |
|---|---|
| CMS | Content Manegment Sytem |
| DIS | Dynamic Info Screen |
| HTML | Hypertext Markup Language |
| JS | JavaScript |
| PHP | Hypertext Preprocessor |
| SSSP | Samsung SmartSign Platform |

# Innehåll

# 1. Inledning

## 1.1 Bakgrund

Företaget där examensarbetet skall utföras heter AcobiaFLUX AB [Aa] och ligger på Lindholmen i Göteborg. Bolaget är medelstort med ca 60 anställda runt om i Sverige och Norge. AcobiaFLUX grundades 1995. Företaget inriktar sig på organisationer som behöver produkter så som automatisk styrning, övervakning, planering och spårbarhet. Deras lösningar används exempelvis inom tillverkningsprocesser och fastighetsstyrning. AcobiaFLUX har även blivit utnämnd till världens bästa Citectintergratör [Ab]. CitecScada är ett SCADA-program (*Supervisory Control And Data Acquisition*) som kan styra och övervaka pågående automatiserade processer.

## 1.2 Syfte

Examensarbetet har kommit till på grund av att AcobiaFLUX hade fått förfrågan av ett antal företag om de hade ett mer centraliserat system för informationsskärmar att erbjuda. Idag kräver dessa skärmar att en lokal dator finns till respektive skärm där datorn har de bilder som ska visas. IT-avdelningen har som uppgift att sköta och kontrollera att bilder laddas upp till varje enskild dator, på varje separat skärm. Detta vill man komma ifrån och ha ett mer centralt system så att hela informationssystemet, kan hanteras från en och samma plats.

## 1.3 Mål

Målet är att undersöka om det finns några aktuella lösningar på marknaden. Dessa skall vara funktionella och smidiga att använda. Vidare skall en grundläggande applikation för en smart–TV plattform tas fram. Applikationen ska kunna hämta fram bilder från en central databas. Bilderna kan vara lunchmenyer, välkomstmeddelande till gästande kunder, dynamisk och produktionsnära information. Detta kan till exempel vara produktionsmål, produktionstakt och dylikt. Applikationen ska även hantera en larm-funktion. När ett larm sker, ska information om detta visas på visningsskärmen. Meddelandet kan då vara en larm-text och vilka tillgängliga brandvägar som finns att tillgå under ett brandlarm.

## 1.4 Avgränsningar

Examensarbetets fokus ligger på Smart-TV teknologin och inte andra smarta produkter, som surfplattor och mobiltelefoner. En annan avgränsning är att prioritera funktionalitet framför den grafiska designen på slutprodukten.

# 2. Metod

Examensarbetets mål är att ta fram en applikation för Smart-TV som ska ha den önskade funktionalitet som är nämnd i inledningen. För att uppnå detta mål är denna process uppdelad i fem steg:

- En marknadsundersökning av befintliga produkter/lösningar skall göras.
- En teknisk utvärdering av de befintliga produkterna/lösningar skall göras.
- Framtagande av prototyp
- Utvärdering av prototyp
- Fastställande av prototyp

## 2.1 Processens tillvägagångssätt

Marknaden söks av för att se om det finns befintliga produkter exempelvis kompletta informationsskärmar. Undersökningen begränsas till en viss period eller tills att ett visst antal produkter hittas. Bland annat, ska Content Management System undersökas, även kallat CMS. De kompletta och kompletterade system som hittas kommer att undersökas för att se hur deras funktionalitet ser ut. Systemens för- och nackdelar skall jämföras med varandra.

Efter undersökningen ska en prototyp tas fram. Denna ska bestå av en applikation till en Smart-TV. Prototypen ska hämta och ta emot information, som senare ska skickas till skärmen och läggas upp i visningsyta för enkel avläsning. Prototypen ska även kunna lägga ut bilder på ett flertal skärmar, som då kan visa olika saker beroende på var den befinner sig. Sådan information kan vara processinformation som visas på en skärm.

## 2.2 Tidsplanering

En tidsplanering är gjord i form av ett Gantt-schema. Detta för att ha en uppfattning hur mycket tid som ska läggas ner och när de olika delmomenten ska vara klara. Tidsplaneringen behöver inte följas till punkt och pricka, utan den är till för att få en helhetsbild. Se appendix A.

## 2.3 Mjukvara

Vid framtagandet av applikationen används programmeringsspråken Java, C# och HTML5. Dessa programmeringsspråk är utöver de som finns inom mekatronikprogrammet på Chalmers. Därför kommer information införskaffas om de nämnda programmeringsspråken. Här kommer bland annat studielitteratur användas så som Jan Skansholms böcker *Java med Swing* [Jsj] och *Skarp programmering med C#* [Jsc].

# 3. Teknisk bakgrund

I detta kapitel presenteras information om de olika programvaror och hårdvaror som används i detta arbete.

## 3.1 Raspberry Pi

Raspberry Pi är en enkortsdator, som inte har något inbyggt minne för operativsystem och filer. Istället använder den sig av ett externt SD-kort för fillagring. Den har flera USB-portar där det går att koppla in en USB-adapter, för exempelvis en extern Wifi-pol. Förutom USB I/O-portar finns det för nätverkskabel, HDMI och AUX som kopplas in i valfri skärm med HDMI port, det finns även Raspberry Pi med RCA port för äldre skärmar. Raspberry Pi är Linux baserad [Rp].

## 3.2 Intel Computer Stick

Intel Computer Stick är en komplett mini-PC. Den är placerad på en form av USB-sticka, fast i stället för en USB-kontakt så har den en HDMI-kontakt. Till skillnad från Raspberry Pi så är Intel Computer Stick en Windows baserad dator. Utrymmet ligger på 2 GB internminne och 32 GB lagringsutrymme. Denna mini-PC har tre I/O-portar för USB, Micro SD och mini-USB för laddning av enheten. Intel Computer Stick har även Wifi och Bluetooth. Själva enheten är ganska liten, endast 103*37*12 mm stor. I denna skulle det gå att köra ett separat program, till exempel en webbläsare [Ic].

## 3.3 HTML

HTML är ett språk för hypertext som används för att bygga webbsidor och utgör grunden för majoriteten av alla webbsidor. HTML fick sitt genombrott i början av 1990 talet, det i samband med att internet spreds. Språket har kontinuerligt utvecklats och alla utvecklare har inte alltid hållit sig till samma standard. På grund av det så kan vissa webbsidor inte öppnas i vissa webbläsare. HTML kan inte hantera dynamisk information utan endast statisk. Om det ska visas dynamisk information så krävs ett annat program eller script [Ht].

## 3.4 JavaScript

JavaScript är ett scriptspråk som används vanligtvis tillsammans med HTML-sidor för att exempelvis hantera bilder eller kontroll av ifyllda fält innan det skickas till servern. Det används även för att bygga upp dynamiska webbsidor med material som rör sig, eller bläddrar. Med andra ord så att en webbsida inte bara blir en statisk bild utan lite mer levande. [Jps].

## 3.5 PHP

PHP är ett scriptspråk som används på webbservrar för att driva webbsidor med dynamiskt innehåll. Vilket är standard på Linux servrar. Däremot kan PHP inte köras på Windows datorer utan att det installeras som ett tillägg. PHP används idag för att hantera dynamisk information och läsa ifrån bland annat databaser m.m. [Afp].

## 3.6 ASP.NET

ASP.NET eller Active Server Pages.NET, är en utvecklingsmiljö som används för att skapa dynamiska webbsidor. ASP.NET är utvecklat av Microsoft och släpptes sensommaren 2012. ASP.NET bygger på .NET Framework som innehåller förkodade program för till exempel databaser m.m. ASP.NET är Microsoft svar på JAVA [Mka].

## 3.7 Content Management System

Content Management System eller CMS som det även kallas är ett innehållshanteringssystem som är ett förenklat sätt att publicera information och hantera administrativa uppgifter på en webbplats vi ett webbgränssnitt och låta koden autogenereras allt eftersom webbsidan byggs ut eller om. Detta ger en person som inte har några större erfarenheter av att programmera hemsidor, en möjlighet att kunna bygga upp och sköta hemsidor [Cs]. Med CMS så syftar man idag oftast på vad som tidigare kallades för WCMS eller Web CMS där man utvecklar sin hemsida direkt i webbläsaren. Skall med andra ord inte förväxlas med DMS (dokumentshanteringsystem) som används för att administrera bilder, dokument och annan elektroniskdata.

De flesta stora webbsidor använder sig av någon form av CMS för att driva deras webbsidor, detta då det underlättar för t.ex. journalister att publicera och uppdatera sina artiklar från var de än befinner sig.

## 3.8 Smart-TV

Smart-TV är en TV som kan vara anslutna till internet för hämtning av information. Smart-TV kan innehålla applikationer och program som kan utföra olika uppgifter exempelvis titta på, film, sociala medier och andra funktioner som den kan utföra då den är uppkopplad mot internet [Stv].

## 3.9 Webbapplikation

En webbapplikation är en programvara som inte är installerad på en enhet för att köras som t.ex. en PC.  Istället körs applikationen via en webbläsare där programvaran befinner sig. På så sätt kan även applikationen underhållas via webbläsaren. Webbapplikationen är skapad i en webbläsare som stöder det språket den är programmerad i som t.ex. JavaScript & HTML[Wa].

# 4. Genomförande

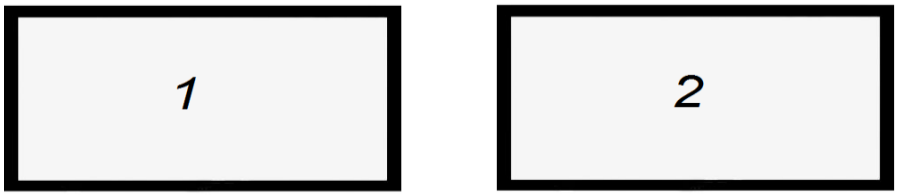
I detta kapitel beskrivs genomförandet av detta projekt. Genomförandet är uppdelat i olika moment som krävs för att nå resultatet av detta arbete. Först söktes marknaden av efter relevanta produkter. Utav de produkter som ansågs mer intressanta undersöktes de mer detaljerat med hjälp av uppsatta krav och önskemål som produkterna skulle uppfylla, där kraven var tvunget att uppfyllas. Om ingen produkt uppfyllde alla krav skulle en egen prototyp tillverkas. Det fanns olika lösningar på hur prototypen skulle se ut och utav de fyra som ansågs mest relevanta användes för att gå vidare. Slutligen tillverkades prototypen utifrån den lösning som ansågs bäst.

## 4.1 Undersökning av befintliga produkter

I början av arbetet undersöktes marknaden på lösningar till informationsskärmar och hur de befintliga produkternas funktioner såg ut. Produkter som var intressanta visade bilder, filmer, texter och andra datavisualiseringar. Samtliga av dessa produkter krävde inga tidigare erfarenheter inom programmering eller mjukvarebehandling för att underhålla de administrativa uppgifterna. Alla produkter hade egna program som tog hand om uppladdningen och visningen av de administrativa uppgifter man ville behandla. Principen som samtliga produkters program utgick ifrån var att först bestämma vad som skulle visas, sedan bestämdes i vilken ordning och hur länge det skulle visas.

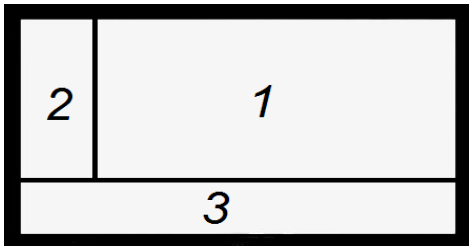
Utav de produkter som ansågs intressanta kategoriserades de som mjukvaror eller hårdvaror. Mjukvarorna var de produkter som kom löst utan att programvaran var inprogrammerad i en skärm. I stället kan programvaran användas med hjälp av en PC eller via en extern enhet som är kopplad till en skärm. Hårdvaror var de produkter som har programvaran inprogrammerad i skärmen och de produkterna definierades som komplettinformationsskärmar. Mjukvaror som hittades och ansågs vara intressanta var SmartSign [Ssig], SpinetiX [Sx], Dynamic Info Screen [DIS], Demorize [Dze] och Playipp [Pi]. Hårdvaror som ansågs vara intressanta var Samsungs SmartSign Platform [Sstv] och LGs SuperSign TV [Lgd], som är kompletta informationsskärmar. Alla ovan nämnda produkter kunde utföra dessa uppgifter. Däremot varierade det om de kunde ha olika bilder/bildspel på olika skärmar. Några klarade av olika bildspel på olika skärmar, vissa hade samma bildspel på alla skärmar. Nedan finns information om varje produkt, med en jämförelse gentemot kravspecifikationen som visar de önskade egenskaperna som ansågs vara relevanta i undersökningen. Marknadsandelen för dessa produkter och informationsskärmar överlag kommer inte nämnas i detta arbete pga. att ingen pålitlig information fanns att hitta. Följande egenskaper är önskade hos produkterna:

*Olika material på olika skärmar*: Produkten skall hantera flera olika uppgifter på ett flertal skärmar. I ett företag kan exempelvis visningsytan på skärm 1 vid entrén visa ett välkomstmeddelande eller ett nyhetsflöde, samtidigt som programvaran kan hantera visningsytan på skärm 2 i en möteslokal som visar schema och dagordningen. Se figur 4.1.




Figur. 4.1 – Två visningsytor på två olika skärmar styrda av samma produkt

*Delad visningsyta*: Skärmen hanterar flera visningsytor på samma gång och på samma skärm. Dock inte som ett bildspel som bläddrar mellan uppgifter, utan den visar alla olika uppgifter samtidigt på skärmen. Se figur 4.2.




Figur 4.2 – Skärm med flera visningsytor

*Stöd för widgetar*: Produkten ska hantera länkar från separata webbsidor som kan visas på visningsytan samt hantera RSS.

*Larm prioritering*: En funktion som hanterar att ta emot information om ett alarm. Detta kan sedan visas i visningsytan. Det kan vara allt från ett fel i en maskin i en industrifabrik till en brand i en skola.

*Läsa ur databas*: Programvaran kan från en separat databas hämta information som i sin tur kan visas på visningsytan.

*PC behövs inte*: I de flesta fall använder sig programvaran av en PC per skärm för att köras. Detta vill man undvika och här undersöks om programvaran bevaras på annat sätt än i en separat PC. Alternativen är att programvaran finns i en extern enhet exempelvis en HDMI-sticka eller mini-PC som Raspberry Pi eller Intel Computer Stick, som i sin tur kan kopplas till en skärm. Ett andra alternativ är att använda sig av en PC-box och det tredje är att programvaran är inprogrammerad i en dator i skärmen som definieras som en komplett informationsskärm i detta arbete.

Figur 4.3 – Exempel på externa enheter. Till vänster mini-PC & till höger PC-box.

## 4.2 Mjukvara

### 4.2.1 SmartSign

SmartSign [Ssig] är ett av de märken som har funnits längst på marknaden. SmartSign AB grundades 1998 och sedan dess lanserat produkter som kan behandla flertalet skärmar med olika material. SmartSign tillverkar många olika produkter som är anpassade till olika arbetsuppgifter. SmartSigns informationsskärmar används bl.a. på sjukhus, restauranger, företag, gym m.m. De produkter SmartSign lanserar kan även lägga upp information, bilder, kalendrar, klocka m.m. Stöd för portabla enheter finns där SmartSigns mobila applikation kan ladda upp material via mobilenheten till andra skärmar. Tillsammans med Samsung har SmartSign tagit fram en gemensam produkt, Samsung SmartSign Platform, även kallad SSSP. SSSP behandlas inte som SmartSign i tabell 4.1 utan det är SmartSigns programvara som behandlas i undersökningen och visas på tabellen. SSSP nämns mer i Samsung avsnittet.

### 4.2.2 Demorize

Demorize [Dze] är en produkt som Dual Heights Software har tillverkat och den kan hämta data från andra källor automatiskt. Demorize är den enda produkten som Dual Heights har att erbjuda. Dock behöver Demorize en PC för att användas. Här skapas ett bildspel på en tidslinje, där bilder placeras ut vid bestämd tidpunkt. Dessutom bestäms var på skärmen bilderna ska visas under den bestämda tiden. Det Demorize kan är att lägga upp bilder, video, PDF, Excel med mera. Den kan även visa ett twitterflöde och webbsidor på sitt bildspel som är live-uppdaterat. Se tabell 4.1.

### 4.2.3 Playipp

Playipp AB [Pi] är ett företag som funnits sedan 2006 och tillverkar informationsskärmar som är anpassade till olika miljöer. Playipps produkter är lik SmartSign som kan hantera det flesta administrativa uppgifter. Playipp kan hantera flera skärmar samtidigt och visa olika material på skärmarna. Skärmen kan innehålla flera visningsytor som utför olika uppgifter. Playipp är inte beroende av en PC för att köras på en skärm, istället används en eller flera externa enheter som kopplas i det antalet skärmar som ska användas. Hämtning av information av separata databaser och använda sig av widgetar är inga problem för Playipp. Som SmartSign kan även Playipp ladda upp information från deras mobila applikation till en av skärmarna som körs. Playipp visade sig vara en av de starkaste produkterna i undersökningen. Se tabell 4.1.

### 4.2.4 SpinetiX

SpinetiX AG [Sx] grundades 2006 och deras produkt SpinetiX hanterar bildspel, video och scheman. Den kan inte lägga upp information från internet. Visningsytan kan visa flera saker samtidigt. SpinetiX kan hantera flera olika skärmar samtidigt och skapa en större visningsyta tillsammans med andra skärmar. SpinetiX kan inte hantera olika information på olika skärmar som andra nämnda produkter. PC behövs inte när SpinetiX används, utan det är en extern enhet som kopplas direkt till skärmen vilken den, tillsammans med Playipp är ensam om bland mjukvarorna. Se tabell 4.1.

### 4.2.5 Dynamic Info Screen (DIS)

DIS är tillverkad av XemiComputers Ltd och har funnits sedan 2008. Denna produkt har ingen extern enhet som Playipp och SpinetiX utan den körs på en PC som i sin tur är kopplad till en skärm. Dynamic Info Screen [Dis] hanterar bilder, video, text, internet webblänk och RSS-flöde. Här skapas ett bildspel som i Demorize. Detta bestämmer vilka administrativa uppgifter som ska visas och när. Dynamic Info Screen hanterar flera skärmar samtidigt med olika uppgifter eftersom olika bildspel kan spelas upp parallellt. Se tabell 4.1.

## 4.3 Hårdvara - Kompetta Informationsskärmar

### 4.3.1 Samsung

Samsung [Sstv] har, som tidigare nämnts, ett samarbete med SmartSign där de har tillverkat SSSP TV. Deras system erbjuder en möjlighet att bygga upp en skärmvägg för att få en större visningsyta. Systemet kan även ta emot information från en portabel enhet, till exempel ett USB-minne som SSSP kan visa i sin skärm. Denna lösning erbjuder även en möjlighet att lägga upp olika material på olika skärmar från samma distribueringssystem. I systemet kan det dessutom välja att visa bilder vid vissa specifika klockslag. Det går också att stänga av och sätta på systemet automatiskt vid specifika klockslag eller via fjärrstyrning. SSSP går under namnet Samsung i undersökningen. Anledningen till det är att SSSP kan läsa ur databaser, som inte SmartSigns mjukvaror kan göra och därför går den under namnet Samsung. Se tabell 4.1.

### 4.3.2 LG

LG [Lgd] erbjuder informationsskärmlösningar tillsammans med Smart-TV eller som en extern enhet. LGs externa enhet kan vara uppkopplad till en eller flera skärmar samtidigt. Den kan även hantera flera skärmar, som då skapar en skärmvägg. LG lansera en produkt som heter LG SuperSign TV, vilken har funktioner som påminner om Playipp och som innebär att en skärm har några specifika ändamål som visas och allt visas samtidigt. Se tabell 4.1.

## 4.4 Sammanfattning av produkterna

Vid granskning av resultatet ovan ansågs Demorize vara ett mindre bra val, då den inte fyllde något av de uppsatta funktionalitetsönskemålen. Produkten kan endast hantera widgetar, vilket även de övriga produkter kan. Bortsett från Demorize kan samtliga produkter ha en delad visningsyta. Det varierade mellan produkterna huruvida de är kapabla att hantera olika material på olika skärmar och detsamma gällde läsning ur en separat databas. Av dessa ovan nämnda produkter ansågs Playipp och Samsungs SSSP TV vara de mest intressanta. Dessa hade ett enkelt program för att publicera material. De hade också möjlighet att kommunicera med olika databaser. Både Playipp och SSSP TV uppfyllde flest av de uppsatta önskemålen. Inte någon av produkterna krävde PC för att användas. Istället levererades de som en extern enhet eller komplett informationsskärm. Det var ingen av produkterna som uppfyllde önskemålet om en larmprioritering. Se tabell 4.1.

| | Olika material på olika skärmar | Larm prioritering | Läsa ur databas | Dela visningsytan | Stöd för widgetar | PC behövs inte |
|---|---|---|---|---|---|---|
| SmartSign | X | | | X | X | |
| Playipp | X | | X | X | X | X |
| Demorize | | | | | X | |
| SpinetiX | | | | X | X | X |
| DIS | X | | X | X | X | |
| Samsung | X | | X | X | X | X |
| LG | X | | | X | X | X |

Tabell 4.1 – Sammanställning av produkterna

## 4.5 Olika lösningar

Det finns kompletta informationsskärmar på marknaden men de saknar vissa vitala delar som är relevanta i detta projekt. Lösningar som är aktuella är dessa.

- Smart-TV med ett CMS-system och extern applikation
- Smart-TV med ett CMS-system och använda webbsidor
- Smart-TV med enbart en applikation
- En webbsida

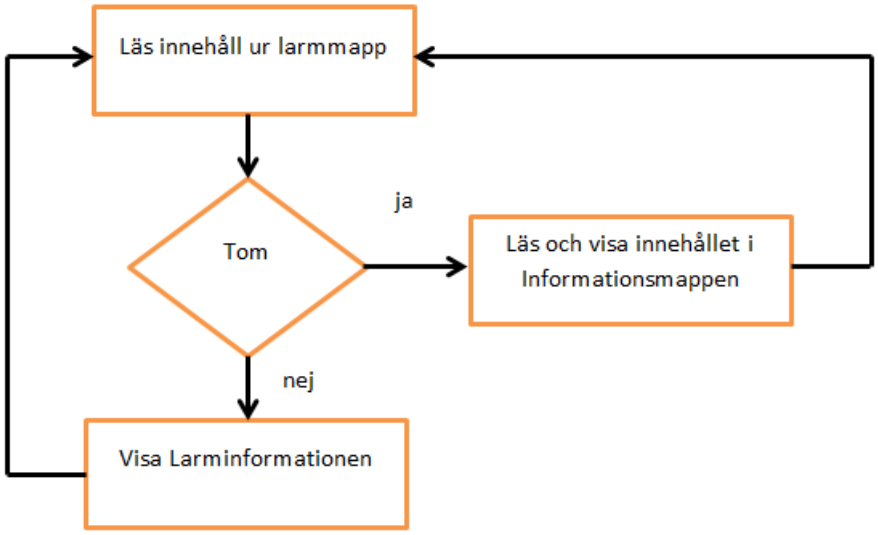
Det framgick relativt snabbt att CMS-system inte var ett alternativ. Med CMS-system fanns ingen möjlighet för ett automatiserat datorprogram att lägga upp och ändra på innehåll samt layout på materialet som skall visas på informationsskärmarna. Därmed blev dessa alternativ uteslutna.

En ren applikation till Smart-TV var länge ett alternativ och att överföra enskilda bilder från en server till Smart-TV var relativt enkelt. Problem uppstod dock när det var dags att automatisera systemet och dessutom visa autogenerade bilder i bildspelet. Ett annat problem som uppstod var att det inte gick att lösa en larmprioritering. Orsaken var att Smart-TV var tvungen att spara alla bilder på sin hårddisk och sedan hämta dem till bildspelet vilket medförde att det inte gick att kontrollera och ändra vad som visades på skärmarna. På grund av detta så uteslöts detta alternativ.

Lösningen med en webbsida bedömdes som enklast och valdes till bästa lösning, då all programvara som behövs för att köra systemet skall vara samlat på en och samma plats, på en lokal server. Kommunikation med externa produkter blev onödig och hanteringen av bilder sköttes från ett och samma ställe. Tidigt i projektet framgick det att det även går att ha applikationen till en annan hårdvara än Smart-TV. Exempel på annan hårdvara är Intel Computer Stick, som inkopplad till en skärm som får möjlighet att köra ett program som visar

bildspelet. En annan är Raspberry Pi, då det erbjuder nästan samma möjlighet som en applikation till en Smart-TV.

Det fanns två förslag på hur applikationens uppbyggnad skulle se ut. Första lösningen var att applikationen ska bestå av två olika mappar, där en av mapparna innehåller de bilder som normalt visas och den andra mappen ska användas i händelse av larm. Tanken är att applikationen hämtar upp bilder från en mapp där kunden själv ska kunna välja vilka bilder som ska visas. Kunden ska kunna lägga in och ta bort bilder och sedan visa den bild som är i mappen på en visningsyta. Vid ett larm ska applikationen prioritera en annan mapp som innehåller varningstext, brandplan och liknande. Se figur 4.3.




Figur 4.4 – Flödesschema till lösning 1

Den andra lösningen är att programmet ska säkerhetskopiera mappar vid larm. Då ett larm sker ska de mappar som används kopieras till en annan mapp. Mappen som sköter vilka bilder som ska visas på skärmarna ska i det läget motta andra bilder från ett systemövervakningsprogram som läggs upp på skärmarna. Lösningen verkar mer komplicerad än den först nämnda och kodningen till denna lösning blir betydligt längre och mer komplicerad.

## 4.6 Framtagande av hemsida

Till att börja med testades att göra ett enkelt bildspel där programmet bläddrade genom ett antal bilder. Enklaste sättet att göra det på var att använda sig av ett stycke HTML-kod och ett JavaScript. I detta test innehöll Javascriptet ett fält (engelska: array) med adresser till de bilder som skulle visas. Dessa bläddrades igenom med hjälp av en loop. Se figur 4.5.

```
//Från javascriptet slideshow.js
//Arrary med bilder till bildspel
var imageArray=["image1.jpg","image2.jpg","image3.jpg"];
//Pekare som pekar på vilken bild som ska visas
var imageIndex=0;
function changeImage(){
      //Bestäm vilken bild som ska visas
      myPhoto.setAttribute("src", imageArray [imageIndex]);
      imageIndex++                                  //Öka pekaren med 1
      if(imageIndex >= imageArray.length){          //Om pekaren är större än
                                                    arrayen =0
      imageIndex=0;
            }
      }
```

Figur 4.5 – Kod ur Javascriptet slideshow.js

Webbsidan hämtar informationen från Javascriptet så att bilderna visas på webbsidan. Med hjälp av kommandot <script> kan HTML koden anropa ett externt JavaScript som hämtar bilderna som är aktuella. Se figur 4.6.

```
<script src="slideshow.js"> </script>
```

Figur 4.6 – Script kommandot

Applikationen hämtar bilder från en mapp. I denna kan kunden bestämma fritt vad som skall skrivas ut på visningsskärmen. Mappen skall automatiskt genomsökas och uppdatera innehållet på webbsidan. Problemet som uppstod här var att JavaScript på grund av ett internetsäkerhetsprotokoll inte ger möjligheten att söka av och hämta filer automatiskt från en eller flera allokerade platser. För att kunna visa de bilder som önskades i bildspelet var de deklarerade direkt i Javascriptet, i detta fall i fält. För att kunna göra inläsningen av bilder till Javascriptet så krävdes det att javascriptet i stället anropade ett PHP script som läser inläsningen av filer och skickar över en matris med det allokerade filerna till Javascriptet och webbsidan. Detta bedömdes vara en icke fungerande metod då de lokala testservrarna använde sig av Windows som operativsystem och med det användes ASP.NET.

På samma sätt som JavaScript kan C# också användas av HTML kod. HTML-koden hanterar även här uppladdningen av bilderna. C# koden hanterar hämtningen av bilderna från mappen och själva roterandet av dessa i bildspelet. Här hämtar sidan automatiskt bilder från en sökväg och lägger in dessa i ett bildspels fält. Detta möjliggjorde att sidan enkelt kunde hantera förändringar i sökvägen, som tillägg och borttagning av bilder. Dessutom kan den ögonblickligen uppdatera innehållet på webbsidan. Koden skrevs så att den hanterar en prioritering av en annan mapp, som då skulle kunna vara en larmhanterings mapp. Om det finns något innehåll i den sökvägen så ska dessa ha prioritet över den vanliga. Se figur 4.7.

```
//Från Image.ascx.cs filen

  protected void Timer1_Tick(object sender, EventArgs e)
  {
      //Kolla i larm-mappen
      System.IO.DirectoryInfo larmDir = new
System.IO.DirectoryInfo(larm_folder);
                  int raknare = 0;        //Initiera en räknare
                  larm.Clear();           //Rensa innehållet i gamla arryen
                  foreach (FileInfo eachfile in larmDir.GetFiles()){
                  //Läs in varje fil från mappen till arrayen.
                            larm.Add(eachfile.ToString());
                            raknare += 1;
                        }
      Session["raknare"] = count1;
      Session["images1"] = larm;
if (larm.Count == 0)    //Om larm mappen är tom hämta bildet här i stället
  {......}
else
   {.....}
```

Figur 4.7 kod ur Image.ascx.cs filen

Vid ett alarm ska de andra bilderna i bildspelet inte visas. Istället hänvisas programmet till en annan mapp med bilder som informerar om till exempel var i byggnaden larmet har uppkommit. Detta larm fungerar genom att en om bild läggs i larm-mappen så visas enbart den mappens bilder. Se figur 4.3. Webbsidan kördes på ett internt nätverk med hjälp av ISS server manager och en virtuell dator placerad på en server. Detta gjorde det möjligt att öppna webbsidan för alla som var anslutna till det lokala nätverket men inte för några utomstående.

## 4.7 Framtagande av Applikationen

### 4.7.2 Android Applikation

Applikationen som tillverkades var tänkt att placeras i Samsungs Smart-TV. Först utvecklades en mobil-applikation för androidmobiler. Detta på grund av att det var enklare att hitta information rörande Android än Samsung.

Den första applikationen kunde hämta en adress från en webbsida och sedan visa sidan på skärmen. URL-fältet i webbläsaren togs bort, för att på så sätt få en snyggare och bättre visningsyta av webbsidan.  Det skedde genom att lägga till denna kod bit i programmet. Se figur 4.8.

```
//Från MainActivity.java filen
mWebView.setWebViewClient(new WebViewClient());
mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.getSettings().setDomStorageEnabled(true);
```

Figur 4.8 – Kod ur MainActivity.java filen

Med detta tillägg så inaktiveras URL-adress baren på webbsidan så att den inte syns senare på applikationen. Med denna kodning lyckas inte applikationen utnyttja hela skärmen, utan det blir en svart ram runt skärmen. Genom att ta bort kodraden som visas i figur 4.9 kunde hela skärmen utnyttjas.

```
//Från activity_main.xml filen
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
```

Figur 4.9 – Kod som togs bort från activity_main.xml filen

Webbadressen från prototypen användes till applikationen på samma sätt som föregående applikation. Skillnaden här var att prototypens webbadress låg på ett lokalt nätverk och applikationen hämtade en webbsida som var global. Om detta ska fungera måste mobiltelefonen vara uppkopplad till samma nätverk som prototypens webbadress ligger på. Genom att lägga till .ax.local/ på URL-adressen adresseras den till webbsidor som ligger i det lokala nätverket. Detta gäller generellt för alla URL-adresser och inte enbart för Java-programmering. Se figur 4.10.

```
//Från MainActivity.java filen
mWebView.loadUrl("http://exjobb-vrobin/");
```

Figur 4.10 – Kod ur MainActivity.java filen

Webbadressen http://exjobb-vrobin/ är den lokala prototypsidan som ligger på företagets server. När applikationen installerades på mobiltelefonen, var bilderna från prototyp-webbsidan inte anpassad för skärmen på mobiltelefonen. Webbsidan är anpassad för vanlig webbsida-besök för mobiltelefoner, men i applikationen fungerade det inte. När en webbsida adresseras till en applikation kan det hända att bilderna från sidan inte är anpassad för visningsskärmen. Applikationen måste anpassa de bilder den visar så att den matchar i visningsyta.

### 4.7.2 Samsung Smart-TV Applikation

För att ta fram en applikation till Samsung Smart TV används Eclipse [Sstv]. Med Eclipse är det möjligt att använda sig av en Samsung Smart TV emulator som simulerar applikationen som har tagits fram. På så sätt få en bild på hur det kan se ut vid en körning på en riktig Smart-TV. Applikationen skall visa prototypens webbsida på visningsskärmen. Det krävdes inte mycket kod för att visa webbsidan på visningsskärmen. Denna rad kod lades till i HTML-filen. Se figur 4.10.

```
<meta http-equiv=”refresh” content=”0, url=http://exjobb-vrobin/” />
```

Figur 4.11 – Kod i Smart-TV applikationen

I Android-applikationen var URL-adressfältet tvunget att tas bort, vilket inte behövdes vid framtagandet av Samsung applikationen. Här är URL-adressfältet redan avvecklat. Vid framtagning av denna Smart-TV applikation uppstod det problem när koden skulle

kompileras. När koden kompilerades rapporterades inga felmeddelande trots att det fanns fel i koden. Den kunde byggas och köras trots de fel som fanns. Felsökning av koden var svår fram tills denna rad av kod hittades.

# 5. Resultat

Projektet blev framgångsrikt. Uppdragsgivaren blev nöjd med resultatet, vilket visar att det inte finns någon lämplig färdig produkt att använda. Undersökningen visade att det finns många kapabla produkter för informationsskärmar, men att de inte klarar av att hantera och prioritera processlarm eller fastighetslarm [se tabell 4.1, sida 8]. Alternativet blev därför att utveckla ett par prototyper till ett eget system. För att det skulle vara möjligt så krävdes det att kunskap inhämtades inom kodspråken JAVA, C# samt HTML.

Prototyperna som utvecklades var två olika som bygger på samma teknologi. Den ena är en webbsida (se avsnitt 4.6.1) och den andra en applikation (se avsnitt 4.6.2) till smarta produkter som hämtar innehållet från webbsidan. För att applikationen skall fungera kräver den att det även finns ett program som gör majoriteten av allt jobb. Detta medför att när det sker en ändring eller en uppdatering av systemet eller gränssnittet behövs det endast ändras på ett ställe och det är på webbsidan. Sker detta så kommer samma ändringar följa med i applikationen. I och med att produkten bygger på ASP.NET, finns det en bred bas av personer som är kunniga i att vidareutveckla och ändra allt eftersom ett företag expanderar eller lägger om fokus.

Prototyperna som togs fram följer överlag de krav som ställdes. En del som saknas, är möjligheten att ladda upp och visa olika material på olika skärmar. Prototyperna kan visa upp hur det är tänkt att det ska fungera, men saknar vissa tidigare planerade funktioner. En annan del som saknas är möjligheten att bläddra till andra sidor, som är tänkt att vara olika sektioner i en fastighet, eller stationer i en produktionsanläggning. Möjligheten att visa något annat än olika bild-format saknas även det. Upplägget för hur bilderna presenteras är heller inte helt komplett men fyller sin funktion för en demonstrationsprodukt. Prototyperna som existerar idag frågar hela tiden servern om det finns en uppdatering för bildspelet som visas på skärmen. Om nya bilder existerar kommer de hämtas och läggas upp på skärmarna Detta går att ändra så att den frågar efter uppdateringar så ofta man vill. Detta för att minimera datatrafiken över det lokala nätverket.

# 6. Slutsats

## 6.1 Återkoppling av kravspecifikation

I kravspecifikationen står det att projektet ska baseras på Smart-tv konceptet. Se appendix B. Tanken var att en applikation skulle utvecklas till Smart-TV plattformen. Detta gjordes genom att applikationen öppnar en webbsida. Lösning valdes som den lättaste och bästa då det efter ett tag framgick att direktuppdatering och synkronisering i realtid mellan server och den externa enheten i fristående applikationsform var svår att lösa på ett bra sätt. För att kunna skicka bilder från en server till en extern enhet och sen visa dessa i ett bildspel så krävs det att bilderna sparas på klienten. Som det framgår i föregående avsnitt så har vissa av kraven inte riktigt uppfyllts. Detta på grund av brist på tid och lägre prioritering. Att kunna visa olika material på olika skärmar anses vara en relativt lätt sak att lägga till. En möjlighet är att skapa flera webbsidor eller skapa olika så kallade flikar som man kan bläddra mellan. Ett annat önskemål var enkel uppdatering av det material som ska visas på skärmarna i systemet. Detta är delvis löst genom att lägga bilder i särskilda mappar på serverenheten som hanterar webbsidan. Detta går dock att utveckla vidare. Tanken är att det ska skapas ytterligare en webbsida där det laddas upp och tas bort material som ska visas.

## 6.2 Applikationen

Att applikationen var uppbyggd på så sätt att den visade endast en hemsida där hemsidan utför det största arbetet låter inte det som en avancerad lösning. En applikation som skulle utföra samma uppgifter som de tillverkade prototyperna utan en webbsida kanske anser som ett bättre alternativ. Men med tanken på hur brett användningsområdet är och hur smidigt det är att endast sköta en webbsida var valet lätt att använda sig av en. Lösningen här är däremot kanske inte avancerad men den utför de uppgifter den ska och möjligheten till vidareutveckling är stor.

## 6.3 Återkoppling till avgränsningar

I avgränsningarna står det "*fokus ligger på Smart-TV teknologin och inte andra smarta produkter så som surfplattor och mobiltelefoner*". Denna avgränsning bröts då det utvecklades en applikation till den lokala webbsidan för en android mobil. Efter framtagningen av prototypen skulle en applikation tillverkas men tillgängligheten av mobiltelefoner var större än Smart-TV apparater. Utifrån det scenariot valdes att programmera en applikation för android mobil. Därifrån upptäcktes även användbarheten kring emulatorer. Istället för att köra det på en fysisk produkt så kördes det virtuellt med hjälp av en emulator som sparade tid.

## 6.4 Marknaden idag och framtid

Informationsskärmsbranschen idag, är fortfarande under stark utveckling och blir mer och mer vanligt. I takt med att allt digitaliseras, ökar ständigt behovet av bra och smidiga informationsskärmar som utför informativa uppgifter beroende på om den befinner sig på en restaurang, ett företag eller fabrik etc. Färdiga produkter finns, exempelvis Samsungs SSSP och klienter som SmartSign och Playipp. Dessa produkter är bara en handfull av de alternativ

som existerar idag. Informationsskärmar kan även användas till vardagliga uppgifter i hemmet där skärmen visar klocka, kalender, nyheter och mycket annat.

# Referenser


[Aa] AcobiaFlux (2015a).*Hemsida*. [Elektronisk]. Tillgänglig: http://www.acobiaflux.se/ [2015-04-20]

[Ab] AcobiaFlux (2015b). *Störst i Europa på Citect*. [Elektronisk]. Tillgänglig: http://www.acobiaflux.se/pdf/referenser/Storst_i_Europa_pa_Citect.pdf [2015-04-20]

[Afp] Freeman, Adam. 2010. *Introducing Visual C# 2010 chapter 34*. 1:a uppl. New York: Apress. [2015-06-03]

[Cs] Wikipedia (2015). *Content management system.* https://en.wikipedia.org/wiki/Content_management_system [2015-06-15]

[Dis] Dynamic Info Screen (2014). *Digital Signage Software*. http://www.dynamicinfoscreen.com/index.php [Hämtad 2015-06-02]

[Dze] DualHeights (2015). *Demorize*. http://www.dualheights.se/demorize/se/ [Hämtad 2015-06-02]

[Ht] Whitehead, Paul, Russel, James H. *HTML*. 1:a uppl. Indianapolis: Wiley Publishing, Inc.

[Ic] Intel Corporation (2015). *Intel Compute Stick*. http://www.intel.com/content/www/us/en/compute-stick/intel-compute-stick.html [Hämtad 2015-06-02]

[Jps] Pollock, John. 2013. *JavaScript*. 4:e uppl. New York: McGraw-Hill/Osbourne

[Jsc] Skansholm, Jan. 2008. *Skarp programmering med C#*. 1:3 uppl. Lund: Studentlitteratur AB

[Jsj] Skansholm, Jan. 2014. *Java direkt med swing*. 8. uppl. Lund: Studentlitteratur AB

[Lgd] LG (2015). *LG Commercial Display: Digital Signage*. http://www.lg.com/us/commercial/display [Hämtad 2015-06-02]

[Mka] Kofler, Michael 2005. *The Definitive Guide to MySQL5*. 3:e uppl. New York: Apress.

[Pi] Playipp (2014). *Digital Signage*. https://playipp.com/sv/ [Hämtad 2015-06-02]

[Rp] Wikipedia (2015). *Raspberry_Pi*. http://sv.wikipedia.org/wiki/Raspberry_Pi [Hämtad 2015-06-02]

[SSig] SmartSign (2015). *Homepage*. http://www.smartsign.se/ [Hämtad 2015-06-02]

[Stv] Wikipedia (2015). *Smart TV*. *https://en.wikipedia.org/wiki/Smart_TV* [Hämtad 2015-06-02]

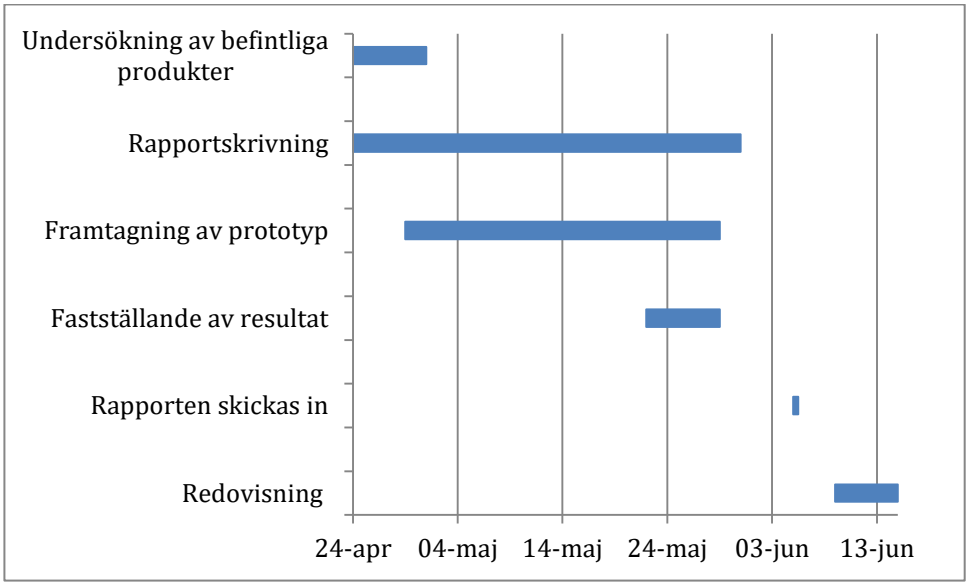
[Sstv] Samsung (2015). *Smart-TV*. http://www.samsung.com/us/business/displays/digital-signage/ [hämtad 2015-06-02]

[Sx] SpinetiX (2015). *Digital Signage*. http://www.spinetix.com/ [Hämtad 2015-06-02]

[Wa] Wikipedia (2015). *Web Application*. https://en.wikipedia.org/wiki/Web_application [Hämtad 2015-07-016]

# Appendix A: Gantt-schema

Detta Gantt-schema är en uppskattad tidsplan över detta projekt. Arbetet startade 24 april 2015 och beräknades vara färdigt 15 juni samma år. Detta schema är endast till för att få en överblick hur uppläget av projektet kan se ut, med andra ord är detta Gantt-schema inte bindande. Figuren nedan är hämtat ur planeringsrapporten som gjordes innan projektet startade.



# Appendix B: Kravspecifikation på prototypen

I projektet kom företaget AcobiaFLUX och författarna översens om en kravspecifikation. Kravspecifikationen utvecklades efter kunderna som hade efterfrågat ett sådant här informationsskärmssystems önskemål. Projektets två krav. Kravspecifikation på projektet.

**Larm Prioritering**: Vid ett larm ska visningsytan visa information om vad för typ av larm det är. Är det exempelvis ett brandlarm ska hela visningsytan utnyttjas till att visa varningstext och brandplan på bygganden som den ska bläddra mellan.

**Läsning ur databas**: I en separat databas kan innehålla information in exempelvis produktion.  I arbetet söks det efter programvaror som kan hämta information från en separat databas som i sin tur kan visas på visningsytan.

# Appendix C: Tabell över informationsskärmsprodukter

De 7 undersökta produkterna med det önskemåls funktioner som det uppfyller.

| | Olika material på olika skärmar | Larm prioritering | Läsa ur databas | Dela visningsytan | Stöd för widgetar | PC behövs inte |
|---|---|---|---|---|---|---|
| SmartSign | X | | | X | X | |
| Playipp | X | | X | X | X | X |
| Demorize | | | | | X | |
| SpinetiX | | | | X | X | X |
| DIS | X | | X | X | X | |
| Samsung | X | | X | X | X | X |
| LG | X | | | X | X | X |