Webbaseret alarmmanagement

Kjartan Døj

Kgs. Lyngby 2006
IMM-B.Eng-2006-63

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-B.Eng.: ISBN 87-643-XXXX-X

Indholdsfortegnelse

Kapitel 1

Forord

Denne rapport er udfærdiget som det afsluttende eksamensprojekt på IT diplom ved Institut for Matematisk Modellering (IMM) på Danmarks Tekniske Universitet (DTU), Lyngby.

Formålet med eksamensprojektet har været at løse en ingeniøropgave i samarbejde med virksomheden UH Communications (UHC) som dokumentation for brug af den under uddannelsesforløbet tilegnede viden til løsning af praktiske problemer.

Projektet har løbet over 10 uger, med start den 31. juli og aflevering den 6. oktober 2006.

Projektet er udført som en enkeltmandsopgave af:

Kjartan Døj
s022338, IT-diplom, DTU, Lyngby
Bag Kirken 16
2670 Greve

E-mail: kjartan@k2d.dk

Kapitel 2

Problemformulering

2.1 Baggrund

Q3ADE 3G er en applikation til centraliseret overvågning af netværksudstyr og andet kommunikationsudstyr.

Q3ADE 3G indeholder i dag en klient med en grafisk brugergrænseflade udviklet i Java, kaldet Q3Browser.

Q3Browser har funktioner til grafisk visning af oversigtsbilleder, hvor alarm situationen kan vises som grafiske kort og netværksudstyr som ikoner på disse kort. Desuden kan alarm operatøren via Q3Browser fremkalde detaljer om alarmer og udstyr i netværket, samt aktivt kvittere og slette alarmer.

2.2 Problemformulering

Udvikling af ny webbaseret alarmovervågnings GUI til UHCs produkt Q3ADE 3G
UHC ønsker at kunne tilbyde en række forbedringer i den grafiske visning af det overvågede udstyr/system, herunder zoomning og bedre integration med tredie parts grafiske værktøjer. UHC ønsker også at gøre det lettere for brugere at tilpasse og udvide funktionaliteten af den grafiske brugergrænseflade. Disse ønsker vil kræve betydelig udvikling i Java i den eksisterende Java-baserede Q3Browser teknologi. Derfor ønskes en ny grafisk brugergrænseflade til alarmovervågningsfunktionen baseret på fremtidige åbne web-baserede teknologier udviklet.

I den forudgående praktikperiode er der foretaget eksperimenter med forskellige nye webstandarder. Disse eksperimenter har vist at SVG standarden fremsat af W3C allerede i nuværende stadie rummer gode muligheder i forhold til ønsket om en flottere og mere skalerbar præsentation. På grundlag af erfaringerne fra praktikperioden ønsker UHC en prototype på en produkt-værdig løsning til den grafiske alarmovervågningsgrænseflade udviklet.

Den nye løsning skal så vidt mulig opfylde en række overordnede krav, herunder:

1. Alarm operatøren skal have de samme operationelle funktioner til rådighed som i den eksisterende Q3Browser
2. Der skal kun benyttes standard-komponenter, der kan hentes gratis på nettet
3. Der skal kun benyttes teknologier, som er de-facto eller forventede industri-standarder, herunder SVG, XML, HTML

4 Det skal kunne køre på Windows, Linux, Solaris og andre operativsystemer udstyret med industristandard web-browsere

5 Det skal aktiveres direkte fra en "tom" klient via web ved opkald til Q3ADE 3G's serverfunktion

2.3 Målsætning

2.3.1 Målsætning for eksamensprojektet

Følgende er en opstilling af den ønskede målsætning for eksamensprojektet. Eksamensprojektet skal omhandle to dele: en teoretisk del der dækker en hvis mængde research i forbindelse med valg af teknologier, understøttelse på forskellige platforme, fremtidsperspektiver for de forskellige teknologier og standarder, osv., og en praktisk del der bl.a. skal omhandle underbyggende eksempler fx mulighederne for brug af SVG i forskellige browsere, indtastningsmuligheder, interaktion, osv.

- Kravspecifikation for alarmmanageren
  Der skal opstilles en redegørelse for de detaljerede krav til alarmmanagerens funktionalitet, samt hvilke muligheder en alarmoperatør skal have. Dette skal gøres på baggrund af de muligheder man allerede har i UHCs nuværende alarmmanager Q3Browser (jvf. UHCs krav om minimum samme funktionalitet som der allerede tilbydes via Q3Browser)

- Use cases for alarmmanagerens funktionalitet
  Der skal opstilles use cases for alarmoperatørens arbejdsopgaver

- Kort redegørelse / baggrund for de forskellige webteknologi standarder
  Der skal kort redegøres for de forskellige udvalgte web industristandarders nuværende stadie, samt deres fremtidsperspektiver. Samtidig skal der redegøres for hvilke browsere der i nuværende / kommende udgaver understøtter / vil understøtte disse standarder, og i hvilken grad.

- Kompatibilitet imellem operativsystemer / browsere
  På baggrund af UHCs krav om et produkt der virker på flere platforme, skal der indhentes information om de valgte industristandarder og om de forskellige operativssystemers /browseres understøttelse af disse.

- Redegørelse for indtastnings muligheder i SVG
  Alarmoperatøren skal have mulighed for indtastning af oplysninger. Der skal undersøges og redegøres for de i forbindelse med SVG mulige indtastnings metoder, da en grafisk editor kræver indtastning af information.

Det er UHCs ønske at de på baggrund af den foretagne research inkluderet i den endelige rapport, kan træffe deres valg omkring hvilke værktøjer, man ønsker at satse på i en fremtidig web baseret version af en alarmmanager.

Den primære opgave vil være at indhente og redegøre for de detaljerede krav, hvilke overvejelser der ligger bag valget af specifikke løsninger i relation til de tekniske muligheder og begrænsninger der eksisterer, mens den sekundære opgave vil være at udvikle en løsning, der i videst mulig omfang realiserer kravene.

2.3.2 Udvidet målsætning

Multibrowser eksempler
Eksperimenterne foretaget under praktikperioden er primært testet og benyttet på Windows platformen under Internet Explorer (med Adobe SVG viewer plug-in). Alle eksempler udviklet under eksamensprojektet skal udvikles så de understøtter og kan afvikles under andre browsere (jvf. UHCs krav om understøttelse af flere operativsystemer / browsere).

Fuld funktionsdygtig demoversion
På baggrund af eksemplerne udviklet under eksamensprojektet og den under eksamensperioden indhentet research, skal en endelig version færdig implementeres.

Gennemgående test af demoversion
Fuld test af demoversionen på alle platforme under flere forskellige browsere.

Kapitel 3

Indledning

UHC er en software virksomhed der leverer værktøjer til udvikling af Network Management Systemer. Deres primære produkt Q3ADE indeholder en række forskellige komponenter der kan oversætte/mediere imellem forskellige management protokoller, således at management systemer baseret på forskellige teknologier kan interopererer. I dag har UHC som standard værktøj til konfigurering, administration. m.m. af Q3ADE en javaklient kaldet Q3Browser. I fremtiden ønsker man af kunne tilbyde kunder en webbaseret applikation til netop konfigurering, overvågning, administration, osv., samt standard komponenter til udvikling af webbaseret management systemer.

Dette projekt omhandler research og udvikling i forbindelse med en webbaseret brugergrænseflade til en alarmmanager. Opgaven er begrænset til at indeholde argumentation og overvejelser i forbindelse med valg af teknologier m.m. der kan bruges til netop dette, samt implementerede eksempler der viser formålet og brugbarheden med de forskellige teknologier. Alle eksemplerne er samlet til en mindre prototype.

Design afsnittet bygger i høj grad på analyse og research delen af rapporten. Designet er tiltænkt en fuldt funktionsdygtig version, hvorfor der kan være uoverensstemmelser med den mindre implementerede prototype.

Implementeringen er foretaget ud fra grundfilosofien om, at der hellere må være en mindre funktionalitet der virker i flere forskellige browsere, frem for mange funktioner der kun virker i en enkelt browser. Denne strategi er valgt med udgangspunkt i UHCs brede udvalg af kunder, der benytter flere forskellige operativsystemer og dermed også forskellige browsere.

Endvidere indeholder rapporten en gennemgående og dokumenteret test af den implementerede prototype.

3.1 Forudsætninger om læseren

Det forudsættes at læseren har et grundlæggende kendskab til web relateret programmering, samt kendskab til tankegangen bag objekt orienteret programmering. Endvidere forudsættes der kendskab til almene udviklingsmetoder inden for software udvikling, herunder brug af UML.

Kapitel 4

Planlægning

4.1 Risikostyring

Der er en lang række eksterne og interne faktorer, der kan have indflydelse på projektet.

1 Udeblivelse

1.1 Sygdom

- Eftersom det er et en-mands projekt, vil alt arbejde være stillestående ved sygdom. UHC skal underrettes så tidligt som muligt. Al underretning om sygdom sendes til Uffe Harksen.
- Såfremt det er muligt arbejdes der hjemmefra. Hvis der ikke er mulighed for dette, kan det blive nødvendigt at revidere tidsplanen.

1.2 Forsinkelse / udeblivelse

- Ved helt eller delvist forhindring i at fremmøde rettidigt, meddeles dette til UHC (Uffe Harksen).

2 Fil- og dokumentrevisionsstyring

2.1 Programmel

- Revisionsstyringsprogrammet Microsoft SourceSafe bruges til at kontrollere fil-revisioner i projektet.
- Såfremt det er muligt skal filer der kan håndteres som tekst tilføjes som tekst i MS SourceSafe, så der er mulighed for at bruge dennes funktioner mht. forskelle på forskellige fil-revisioner.
- Hvis en fil ikke kan håndteres som tekst i MS SourceSafe (binære filer) skal den nyeste revision af filen bruges som udgangspunkt for ændringer.
- Der skal udfyldes en beskrivende log-meddelelse for hver ny revision der tilføjes til projektet, således at man kan holde styr på de ændringer der foretages undervejs.

3 Datatab

3.1 Menneskelige fejl

- Brugen af et revisionssystem placeret på en server er sat som krav med henblik på at kunne afhjælpe denne slags situationer. Skulle det alligevel ske at data går tabt, må man hente den sidst gemte version og arbejde videre der fra. Hvis det ikke umiddelbart er

muligt at løse problemet på egen hånd, kan der tages kontakt til UHC (Søren Vejgaard-Nielsen) (se punkt 4 for kontaktinformation).

3.2 Maskinelle fejl

- Den lokale version der arbejdes på skal uploades til serveren dagligt, således at der altid tages backup af dagens arbejde.
- Yderligere sikring sker ved en ugentlig backup af serveren til bånd.

4 Kontaktinformation

4.1 Paul Fischer, vejleder IMM, DTU

- DTU, Lyngby - Bygn. 322 Lok. 113
- Tlf. arbejde: (+45) 45 25 37 13
- E-mail: paf@imm.dtu.dk

4.2 Søren Vejgaard-Nielsen, vejleder UHC

- Tlf. arbejde: (+45) 44 68 08 64
- E-mail: svn@uhc.dk

4.3 Uffe Harksen, UHC

- Tlf. arbejde: (+45) 44 68 08 64
- E-mail: uh@uhc.dk

4.4 Kjartan Døj, s022338, DTU

- Tlf. arbejde: (+45) 44 68 08 64
- Mobil: (+45) 20 20 82 31
- E-mail: kjartan@k2d.dk

4.2 Tidsplan

For at holde styr på projektet er der indledningsvis udarbejdet en tidsplan. Jeg har for så vidt muligt forsøgt at overholde de specificerede tidsfrister, dog er der løbende blevet lavet enkelte revideringer. Som udgangspunkt ville kravspecifikationen være det første der skulle laves, men pga. ferie hos UHC, er denne flyttet frem i tidsplanen.

Jeg har undervejs i projektet udformet forskellige diagrammer og taget noter til de forskellige afsnit af rapporten, samt løbende sørget for at skrive de relevante afsnit. Eftersom den primære del af rapporten er research, har det været naturligt at skrive mindre delkonklusioner og diverse teoriafsnit efterhånden som de forskellige kilder er blevet gennemgået.

Tidsplan

Uge 31		
Ma	31-07-2006	Opstart. Tidsplan. Risikostyring. Planlægning
Ti	01-08-2006	Redegørelse / baggrund for web industristandarder
On	02-08-2006	Redegørelse / baggrund for web industristandarder
To	03-08-2006	Redegørelse / baggrund for web industristandarder
Fr	04-08-2006	Redegørelse / baggrund for web industristandarder
Uge 32		
Ma	07-08-2006	Research / test af indtastningsmuligheder
Ti	08-08-2006	Research / test af indtastningsmuligheder
On	09-08-2006	Research / test af indtastningsmuligheder
To	10-08-2006	Research / test af indtastningsmuligheder
Fr	11-08-2006	Research / test af indtastningsmuligheder
Uge 33		
Ma	14-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
Ti	15-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
On	16-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
To	17-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
Fr	18-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
Uge 34		
Ma	21-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
Ti	22-08-2006	Undersøgelse af kompatibilitet, implementation / test af eksempler
On	23-08-2006	Kravspecifikation for alarmmanager
To	24-08-2006	Kravspecifikation for alarmmanager
Fr	25-08-2006	Kravspecifikation for alarmmanager
Uge 35		
Ma	28-08-2006	Kravspecifikation for alarmmanager
Ti	29-08-2006	Kravspecifikation for alarmmanager
On	30-08-2006	Use cases for alarmoperatørens funktionalitet
To	31-08-2006	Use cases for alarmoperatørens funktionalitet
Fr	01-09-2006	Use cases for alarmoperatørens funktionalitet
Uge 36		
Ma	04-09-2006	Integration af eksempler i demoversion
Ti	05-09-2006	Integration af eksempler i demoversion
On	06-09-2006	Integration af eksempler i demoversion
To	07-09-2006	Integration af eksempler i demoversion
Fr	08-09-2006	Integration af eksempler i demoversion
Uge 37		
Ma	11-09-2006	Integration af eksempler i demoversion
Ti	12-09-2006	Integration af eksempler i demoversion
On	13-09-2006	Integration af eksempler i demoversion

fortsættes...

Tidsplan (fortsat)		
To	14-09-2006	Integration af eksempler i demoversion
Fr	15-09-2006	Integration af eksempler i demoversion
Uge 38		
Ma	18-09-2006	Rapport
Ti	19-09-2006	Rapport
On	20-09-2006	Rapport
To	21-09-2006	Rapport
Fr	22-09-2006	Rapport
Uge 39		
Ma	25-09-2006	Rapport
Ti	26-09-2006	Rapport
On	27-09-2006	Rapport
To	28-09-2006	Rapport
Fr	29-09-2006	Rapport
Uge 40		
Ma	02-10-2006	Rapport
Ti	03-10-2006	Rapport
On	04-10-2006	Rapport. Sidste gennemlæsning
To	05-10-2006	Rapport. Sidste rettelser, trykning / samling
Fr	06-10-2006	Aflevering

Kapitel 5

Teori

Følgende afsnit giver en kort gennemgang af de forskellige udvalgte standarder. Dette afsnit er medtaget da flere af standarderne kun lige er blevet færdigdeklareret, eller stadig kun findes som udkast. Derfor er der også et begrænset kendskab til disse, da udbredelsen stadig er undervejs.

Fordelen ved at benytte standarder er primært, at det er nemmere at lave et produkt, der kan benyttes af mange forskellige mennesker uanset valg af fx browser. De fleste browserudviklere forsøger at overholde og understøtte standarder fremsat af fx World Wide Web Consortium (W3C), hvilket også er med til at øge konkurrencen om at understøtte mest muligt og derved igen med til at udbrede brugen af standarderne.

5.1 Scalable Vector Graphics (SVG)

I den moderne verden benyttes billeder og video som aldrig før. En hjemmeside uden en eller anden form for grafik, ryger direkte under definitionen "en kedelig hjemmeside". De fleste computer har heller ingen problemer med at afspille fx video, eller vise billeder i meget høj opløsning. Den største flaskehals i dag er netværksforbindelsen. Selvom (A)DSL linier og fiberlinier vinder større og større udbredelse, kan denne udbredelse dog ikke følge med det egentlige behov. Derfor er det naturligt at forsøge at udvikle et format der tillader billeder i alle opløsninger uden at det har nogen særlig indflydelse på størrelsen i MB. Svaret på dette findes i vektor grafik.

SVG er en platform til 2-dimensionel grafik. Den består af to dele: et XML baseret filformat og et programmerings API til grafiske applikationer.

SVG er en royalty-free, leverandør uafhængig, åben standard udviklet af World Wide Web Consortium (W3C). Royalty-free gør at alle frit kan benytte det, hvilket er med til at sikre en større og hurtigere udbredelse. Samtidig gør leverandør uafhængigheden og det faktum at det er en W3C standard, konkurrencen større, hvilket dermed er med til at fremskynde en hurtigere og større udbredelse.

SVG henter støtte fra mange af de største virksomheder indenfor computer industrien og flere af disse virksomheder har også bidraget med medarbejdere til standardiseringen af SVG. Virksomheder der har ydet støtte dækker bl.a. Adobe, Agfa, Apple, Canon, Corel, Ericsson, HP, IBM, Kodak, Macromedia, Microsoft, Nokia, Sharp og Sun Microsystems. Der er allerede sendt SVG viewers ud til over 100 millioner enheder og flere og flere enheder understøtter direkte brug af SVG.

SVG bygger på mange allerede velrenomerede standarder såsom XML (SVG grafik er tekst baseret og derfor nemt at skrive/redigere/søge i), JPEG og PNG som indsat grafik formater, DOM til brug af scripting og interaktivitet, SMIL (Synchronized Multimedia Integration Language) til animationer og

CSS til formatering.[3]

Ideen til SVG startede oprindeligt med henblik på mindre enheder så som PDA'er og mobiltelefoner. Salget af disse enheder er eksploderet inden for de sidste 10 år og en mobiltelefon nærmer sig hastigt hver mands eje.
Mobiltelefoner og PDA'er, har pga. mobiliteten ofte en begrænset båndbredde til rådighed. Derfor skulle filstørrelsen altså begrænses, helst uden at man gik på kompromis med kvaliteten. Samtidig er der nærmest ikke to enheder med samme skærmdimensioner og skærmopløsning, så formatet skulle altså også kunne tilpasses til alle opløsninger, med andre ord det skulle være skalerbart!
For at imødekomme kravet om et skalerbart grafikformat der kan afvikles på mindre CPUer, med en begrænset mængde RAM og lagerplads, udviklede W3C SVG. En SVG tegning skulle kunne skaleres til alle størrelser uden at man var nødt til at lave flere versioner, som det er tilfældet med almindelig raster grafik (JPG, PNG, GIF, osv.). Et normalt raster grafik billede kan også skaleres, men hvis det skal gælde både en meget høj opløsning og en meget lille opløsning, uden at man gemmer flere versioner af det samme billede, er man nødt til at benytte billedet med meget høj opløsning, men vise det meget småt. Denne visning fjerner ikke noget af billedets data, men skalerer det bare ned. Dvs. at man er tvunget til at hente en stor unødvendig mængde data, som i princippet aldrig vil blive brugt. Dette er en af de største fordele ved SVG. Et SVG dokument er egentlig en skabelon til hvordan man tegner lige netop denne ting. Dvs. at man kan vise dokumentet i en hvilken som helst skalering uden at man skal bruge mere data. Samtidig er det også den enkelte enhed der renderer SVG dokumentet og sørger for at tegne i lige netop den skalering der passer til den pågældende enheds skærm.

Som nævnt er SVG originalt tiltænkt mindre enheder, men der står intet til hindring for at benytte det på en almindelig moderne PC. Faktisk indeholder en moderne PC så meget processorkraft at man uden problemer kan lade sine SVG dokumenter indeholde animationer, gradiente farvelag, osv. Igen er SVG dokumentets force at man med vha. en række korte tekst linier beskriver hvordan animationen skal se ud, og så lader det være op til den enkelte enhed rent fysisk at lave beregningerne der får animationen til at ske.
På trods af SVGs fleksibilitet afviger enhederne meget i fx CPU kraft, RAM mængde, osv. Derfor har W3C inddelt SVG specifikationen i *profiler*. Hver profil dækker forskellige typer enheder. Sammenhængen findes i den hierarkiske opbygning af profilerne. Figur 5.1 viser denne opbygning de forskellige profiler. Den mindste profil er en underdeling af den overliggende, osv.

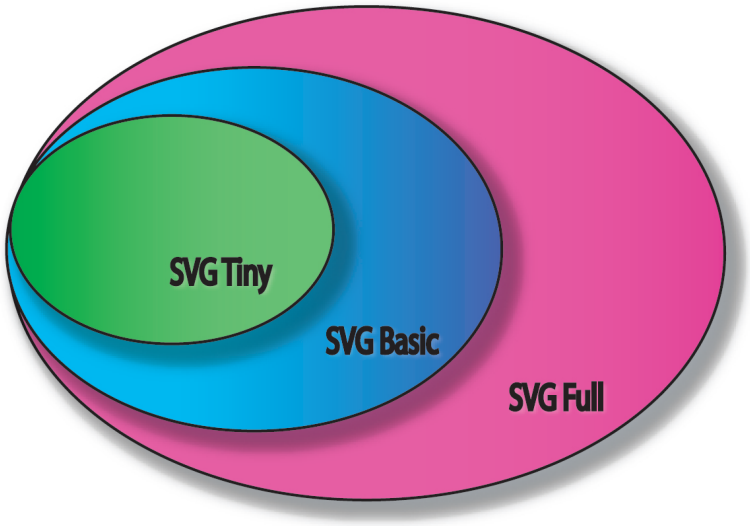
5.1.1 Mobile SVG Profiles

De følgende afsnit omhandler de enkelte forskellige SVG profiler fremsat af W3C. For en komplet oversigt over hvilke elementer de forskellige profiler dækker, se figur A.1 på Bilag A på side 68.

SVG Tiny og SVG Basic er beskrevet hos W3C under SVG Mobile. Både SVG Mobile og den fulde specifikation for SVG findes i skrivende stund som version 1.1 og blev officielt gjort til en W3C Recommendation 14. januar 2003. SVG 1.2 specifikationen er under udarbejdelse.

SVG Tiny (SVGT)

SVG Tiny er den profil der indeholder færrest elementer og henvender sig derfor også til de mindste enheder, nemlig mobiltelefoner. Bl.a. har mange af de nyeste Sony-Ericsson mobiltelefoner en indbygget SVG Viewer der bruges til animerede baggrunde, menuer, osv., samtidig med at den kan vise "almindelige" SVG billeder.



Figur 5.1: Den hierarkiske opbygning af SVG profilerne

Figur 5.2: Sony Ericsson W800i mobiltelefon med indbygget SVG Viewer

SVGT indeholder alle form elementer, dvs. man kan tegne både ellipser, cirkler, firkanter, osv. Den eneste undtagelse er buer tegnet med `<path>` elementet (kommandoen A eller a, arcs). Løven på figur 5.2 er fx tegnet vha. polygoner.

Animationer er også understøttet, dog kun deklarativ animation, da SVGT ikke indeholder nogen DOM model og derfor ikke understøtter nogen form for scripting ved brug af fx JavaScritp (eller andet

ECMA script kompatibelt sprog). SVGT understøtter brug af `<image>` elementet med raster grafik (fx PNG og JPEG filer), men ikke med andre SVG billeder.
SVGT understøtter kun solid fyldfarve (enkelt fyldfarve) og altså ikke nogle filtre eller gradiente fyldfarver.

SVG Basic (SVGB)

Som tidligere nævnt er SVG profilerne defineret hierarkisk (se figur 5.1). Dvs. at alt hvad SVGT understøtter, understøttes også af SVGB.

Derudover understøtter SVGB valgfri implementering af de samme elementer til brug af scripting som SVG 1.1. Herunder også tilgang til fx at starte animationer gennem scripting. Eftersom bruger input af forskellig slags på en PDA nærmer sig en almindelig PCs input, giver det også mere mening at kunne udføre scripts på en PDA end på fx en mobiltelefon.

SVGB understøtter en del filter effekter men ikke alle. Derimod er gennemsigtighed (opacity) understøttet for både omkredsne kanter af figurer og fyldfarve, og muligheden for at bruge gradiente fyldfarver er også medtaget.

5.1.2 SVG 1.1 (Full)

Den fulde SVG specifikation, SVG Full, dækker alt hvad SVG skal kunne. Dvs. både animation, gradiente fyld farver, alle former (cirkler, firkanter, ellipser, osv.), fonte, scripting og meget meget mere. Den fulde specifikation henvender sig til almindelige PCer, da fx animeringer af figurer med mange filtre og gradiente udfyldninger kræver en større mængde CPU kraft.

5.2 XForms

En ting er at en alarmmanager kan præsentere data, noget andet er at indhente data fra brugeren. Formularer er i dag en vigtig del af mange hjemmesider. De gør det muligt at modtage input fra brugeren. XForms er næste generation af de almindelige HTML formularer som man kender fra i dag. Almindelige HTML formularer tilbyder fortrinsvis begrænsede muligheder, mens XForms er designet til at tilbyde mange nye funktioner som man før var nødt til selv at udvikle i fx JavaScript.

XForms er en standard fremsat af W3C og den nuværende version er i øjeblikket XForms 1.0 (anden udgave). XForms bliver formular standarden i XHTML 2.0.

Som så mange andre nye standarder bygger XForms på XML. Det gør den fleksibel og meget nem at integrere med andre standarder der også bygger på XML, fx WML (Wireless Markup Language) eller SVG (Scalable Vector Graphics). Derfor er data og præsentation også adskilt i XForms. XML strukturen sørger for at opbevare data, mens det bliver op til den enkelte browser/enhed hvordan de enkelte elementer kommer til at se ud. Dette gør også at man kan benytte den samme formular på forskellige enheder. Det er enhedens opgave at tilpasse præsentationen til fx forskellige skærmstørrelser, mens datahåndteringen vil være den samme.

Med normale formularer er man afhængig af scripts (fx skrevet i JavaScript) til at validere indtastet data på klientsiden. XForms har en indbygget attribut man kan sætte til hvilken datatype feltet skal indeholde. Indtastes en datatype, fx et bogstav i et felt hvor typen er sat til `integer`, kan man via den indbyggede fejlhåndtering udskrive en dialogboks der gør brugeren opmærksom på dette. Dvs. at XForms i høj grad mindsker behovet for scripting og i flere tilfælde gør den fuldstændig overflødig.

XForms er blandt de meget nye teknologier. XForms 1.0 (Second Edition) blev først en W3C Recommendation 14. marts 2006, hvilket også forklare den meget begrænsede udbredelse den endnu har.[5]

5.3 Document Object Model (DOM)

Document Object Modellen er et platform- og sprog-neutralt interface der gør det muligt for programmer og scripts at dynamisk tilgå og opdatere indholdet, strukturen og layoutet af et dokument. Dokumentet kan processeres og resultatet af den processering kan så tilbage reflekteres på det præsenterede dokument.

DOM er taget med her da den også gør sig gældende for både SVG, XML og almindelige HTML filer. Implementeringen af DO modellen i de forskellige browsere gør det muligt at tilgå alle elementer på en vist side og gennem scripts ændre de forskellige elementer. Det kan være alt lige fra at ændre farve på et element (fx hvis en ny alarm optræder kan man signalere dette ved at rammen omring et ikon skifter farve), til at indsætte nye elementer. Specielt XML dokumenter er meget fleksible mht. redigering via DO modellen, da XML dokumentet i forvejen er meget systematisk bygget op og alle tags skal lukkes (afgrænsning af data). Da SVG bygger på XML og også har en DO model til rådighed i browserne, kan man altså via DO modellen fx tilføje en ny router (et SVG dokument fragment) på sit alarmmanager oversigts billede (hoveddokument) uden at hele siden skal hentes ind fra en (web)server igen. Man kan altså ved at opdatere og redigere sit dokument gennem DO modellen give brugeren en mere smertefri præsentation og egentlig holde siden kørende mere som en applikation end en hjemmeside.
DO modellen indeholder de forskellige standarders elementer og de enkelte elementers attributer har man derfor også mulighed for at ændre. Fx kan man ændre `visibility` attributten og derved skiftevis skjule eller vise fx indtastningsfelter på en HTML side, således at der kun er de felter brugeren har brug for.

5.4 XMLHttpRequest

XMLHttpRequest objektet er et interface til rådighed for en script engine således at det er muligt at udføre HTTP klient funktioner, som fx sende formular data eller indhente data fra en server.

Navnet XMLHttpRequest er givet for at understrege kompabiliteten med Internettet. Objektet understøtter også overførsel af andre dataformater end XML, fx almindelige tekststrenge.

Man kan altså med dette objektet indhente data i baggrunden uden at skulle genindlæse hele dokumentet igen, og så gennem scripting indsætte den nye data. Et eksempel kunne være når en bruger havde indtastet sit postnummer i en bestillingsformular, så slog dokumentet op hvad den tilhørende by er og indsatte det ved siden af. Et andet eksempel i forhold til en alarmmanager kunne være et GUIDOmaps pollingsmekanisme. Man kunne her benytte XMLHttpRequest objektet til at undersøge om der skulle være tilføjet nyt udstyr (ikoner) til GUIDOmap'et af en anden bruger.

XMLHttpRequest objektet har gennem de seneste år været implementeret som en ActiveX kontrol i Micorsoft Internet Explorer og er nu også blevet en del af mange andre browsere (dog ikke som ActiveX komponent, men som et almindeligt JavaScript objekt). Det skal dog understreges at W3Cs standardisering af objektet stadig *kun er under udarbejdelse*, hvorfor de forskellige browseres implementation af objektet også kan afvige fra hinanden. [6]

Kapitel 6

Analyse

Analyse kapitlet dækker overvejelser og begrundelser for valg af teknologier m.m. Dette kapitel fylder en stor del af rapporten eftersom det er på baggrund af de konklusioner, der er kommet ud af den her foretagne research, at UHC kan basere deres valg mht. en webbaseret alarmmanager.

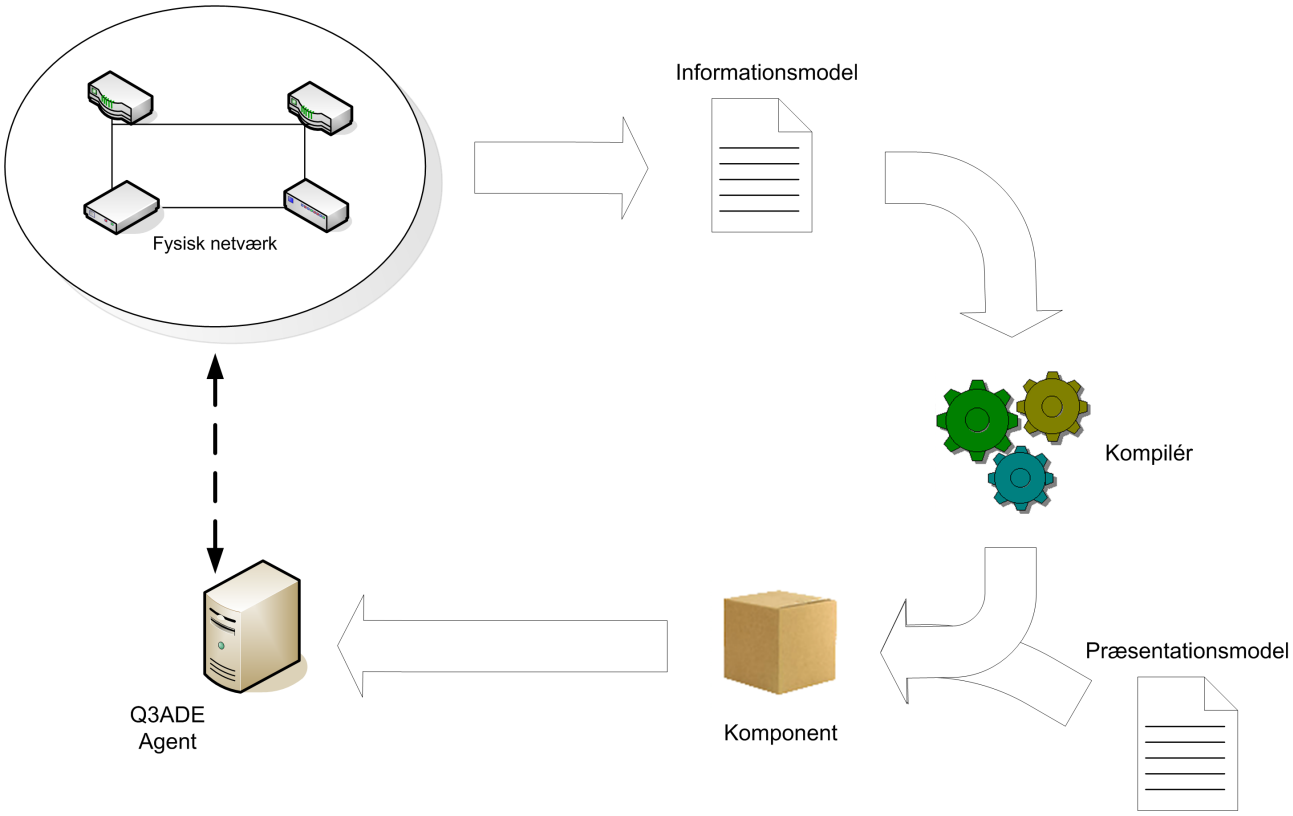
6.1 Baggrund

Som nævnt tidligere er UHC en virksomhed der leverer værktøjer til udvikling af Network Management Systemer. Deres primære produkt Q3ADE indeholder en række forskellige komponenter der kan oversætte/mediere mellem forskellige management protokoller, således at management systemer baseret på forskellige teknologier kan interoperere. For at Q3ADE agenten kan genkende et stykke tilkoblet udstyr, er der først en række operationer der skal gennemføres.

Først skal det fysiske netværk oversættes vha. et formelt sprog til en informationsmodel. Denne model indeholder al information om det specifikke stykke udstyr, altså den information man i den sidste ende kan lave management på. Informationsmodellen kompileres til en komponent pakke, der så kan lægges ind i Q3ADE agenten. Herefter kan man så via Q3ADE agenten kommunikere med det fysiske udstyr. Eller sagt med andre ord man kan lave management på udstyret! Figur 6.1 viser processen fra det fysiske netværk til en komponent pakke, man kan lægge ned i Q3ADE agenten.

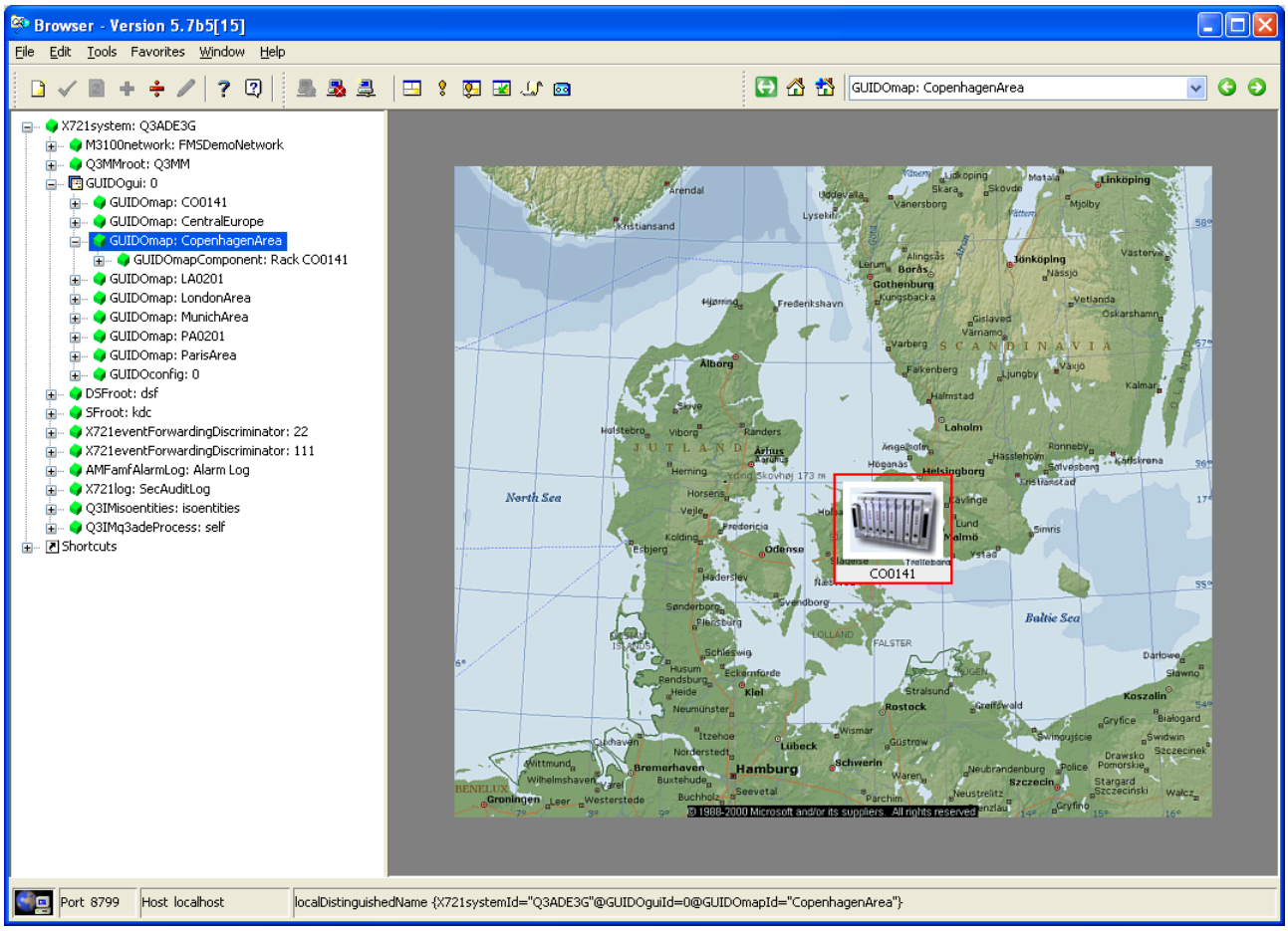
Et er at agenten kan genkende et stykke udstyr, noget andet er at skabe en grafisk præsentation i management systemet, således at en operatør kan danne sig et overblik over hvad der er tilkoblet systemet og hvordan de forskellige stykker udstyr er koblet sammen. I Q3ADE agenten har man mulighed for at oprette såkaldte *GUIDOmaps* til netop denne grafiske repræsentation af ens netværk. GUIDO er en grafisk præsentationspakke udviklet af UHC og som standard management værktøj har UHC udviklet Q3Browseren, som bygger på GUIDO java biblioteket. Figur 6.2 viser et eksempel på et GUIDOmap vist i Q3Browseren.

Informationsmodellen er en objektorienteret model hvor forskelllige klasser repræsenterer forskellige stykker udstyr. Disse klasser oprettes/instantieres i agenten således at de repræsenterer det fysiske netværk. Dette kan enten ske gennem autodiscovery eller gennem konfigrations-scripts der downloades og eksekveres i agenten. Den præsentation af det overvågede system man ønsker at vise for en alarm operatør er ikke nødvendigvis sammenfaldende med den hierarkiske struktur af disse instantierede klasser. Derfor har GUIDO nogle klasser/komponenter beregnet til præsentation af systemet, hvor et GUIDOmap viser relationer mellem forskellige komponenter (GUIDOmapComponents) og hvor feks. en GUIDOmapComponent referere til en instantieret klasse der endeligt repræsentere et stykke fysisk udstyr.



Figur 6.1: Oversættelses processen fra det fysiske netværk til brugbar komponent pakke

Når udstyret så er tilkoblet og genkendt af agenten og man har oprettet en grafisk repræsentation af sit netværk, kan man så overvåge om alt er som det skal være for det pågældende stykke udstyr. Hvis der forekommer en fejl i et stykke udstyr, udsender udstyret en alarm. Alt efter hvilken fejl der optræder, kan en alarmoperatør angive forskellige grader af alvorlighed for den optrådte alarm i management systemet. GUIDOmaps kan placeres hierarkisk i forhold til hinanden, hvorefter man så kan foretage drill-down fra det øverste lag til det nederste. Samtidig bobler alarmer også op gennem hierarkiet og vises på det øverste lag. Figur 6.3 viser et eksempel på en hierarkisk opbygning, hvor man kan klikke sig helt ned til et specifikt rack. Her starter man med et oversigtskort over Europa. På Europa kortet er der placeret fire ikoner, et for hvert del af Europa man kan "zoome" tættere på. Man kan se at der er optrådt en kritisk alarm (critical = rød) længere nede i hierarkiet. Klikker man på ikonet placeret ved Danmark hvor alarmen er optrådt, kommer man til et Danmarks kort. Dette kort indeholder et ikon for hvor i Danmark der står et rack placeret. Klikker man igen på rack ikonet kommer man helt ned til en grafisk præsentation af det fysiske rack og man kan nu se præcis hvilken komponent i rack'et der har udsendt den kritiske alarm. Hvis fx alarmen var udsendt fordi komponenten var ved at brænde sammen, kunne alarmoperatøren fx tilkalde en tekniker, der så kunne udskifte delen. Dette er den del af en alarmmanager som UHC ønsker udviklet som en webbaseret applikation i stedet, altså hvad dette projekt omhandler.



Figur 6.2: GUIDOmap vist i Q3Browseren

6.2 Standarder

På baggrund af UHCs krav om at en webbaseret alarmmanager udelukkende skal være baseret på de-facto standarder (se Problemformuleringens Krav 3 på side 4) er følgende valgt ud efter gennemgang af W3Cs liste over standarder: SVG som grafisk præsentationsmiddel og XForms til brug for indtastningsmuligheder. Udvælgelsen er foretaget på baggrund af hvilke standarder der er tilgængelige indenfor de to kategorier: Grafisk præsentation og bruger input. Begge bygger på XML og det er derfor meget nemt at benytte begge teknologier i samme fil. De følgende afsnit vil give en mere dybdegående analyse af de udvalgte standarder set i forhold til brugbarheden i dette projekt, samt fremtidige versioner.

6.2.1 SVG

Som tidligere nævnt er SVG et særdeles egnet grafisk præsentations middel. Vha. SVG er det muligt at lave flot 2D grafik med skygger, gradiente farveligninger, osv. samtidig med at man bevare en lille filstørrelse og et billede der kan vises i alle opløsninger.

Samtidig gør de indbyggede animationsmuligheder det også muligt at lave en flottere præsentation med fx animerede overgange mellem maps, zoom uden tab af kvalitet, m.m. Endvidere har man gennem

Figur 6.3: Drill down map eksempel

Document Object (DO) modellen mulighed for at opdatere de viste oplysninger uden at indlæse hele siden på ny. Dvs at man kan lave noget der mere minder om en applikation end om en hjemmeside.

6.2.2 XForms

XForms er almindelige HTML formularer langt overlegne. Mange firmaer har allerede investeret mange penge i XML formatet og XForms rødder i XML gør den nem at integrere med allerede udviklede løsninger. Samtidig gør opdelingen af data og præsentation også XForms langt mere fleksibel i forhold til hvilken enhed der skal vise formularen. Endvidere indeholder XForms mulighed for beregninger og klientside validering af brugerinput uden brug af scripts, hvilket kan mindske udviklingstiden meget.

XForms har altså potentialet til at blive den næste formular standard på Internettet. Den er også valgt som formular standard for XHTML 2.0, hvilket endnu en gang understreger at XForms er noget man forhåbentligt kommer til at se meget mere til i fremtiden. Som nævnt tidligere blev XForms først gjort til en W3C Recommendation, altså en vedtaget standard, i marts 2006, hvilket også fører til det *men* der skal findes omkring XForms. Pga. den meget korte "levetid" er der ingen browsere der direkte understøtter brug af XForms og der går nok et stykke tid før man ser browsere der direkte gør dette.

6.3 Browsere vs standarder

Dette afsnit omhandler forskellige browseres understøttelse af de forskellige industristandarder. Tabel 6.1 viser en oversigt over de i samarbejde med UHC udvalgte standarder, samt hvilke browsere der understøtter disse.

Browserne Firefox, Opera og Internet Explorer er valgt på baggrund af UHCs erfaringer med hvilke browsere deres kunder oftest benytter. UHCs hovedprodukt Q3ADE 3G kan afvikles på Solaris, Red Hat Linux og Windows. Derfor er det nødvendigt at se på flere forskellige browsere, således at folk med forskellige styresystemer alle kan benytte sig af den webbaserede alarmmanager.

Browser	Opera	Firefox	Internet Explorer	
Standard	9.0	1.5	6.0	7.0
XML	✓	✓	✓	✓
DOM	✓	✓	✓	✓
XMLHttpRequest (draft)	✓	✓	✓	✓
SVG 1.1 Tiny	✓	(✓)	-	-
SVG 1.1 Basic	✓	(✓)	-	-
SVG 1.1 Full	✓	(✓)	-	-
XFORMS 1.0	-	-	-	-

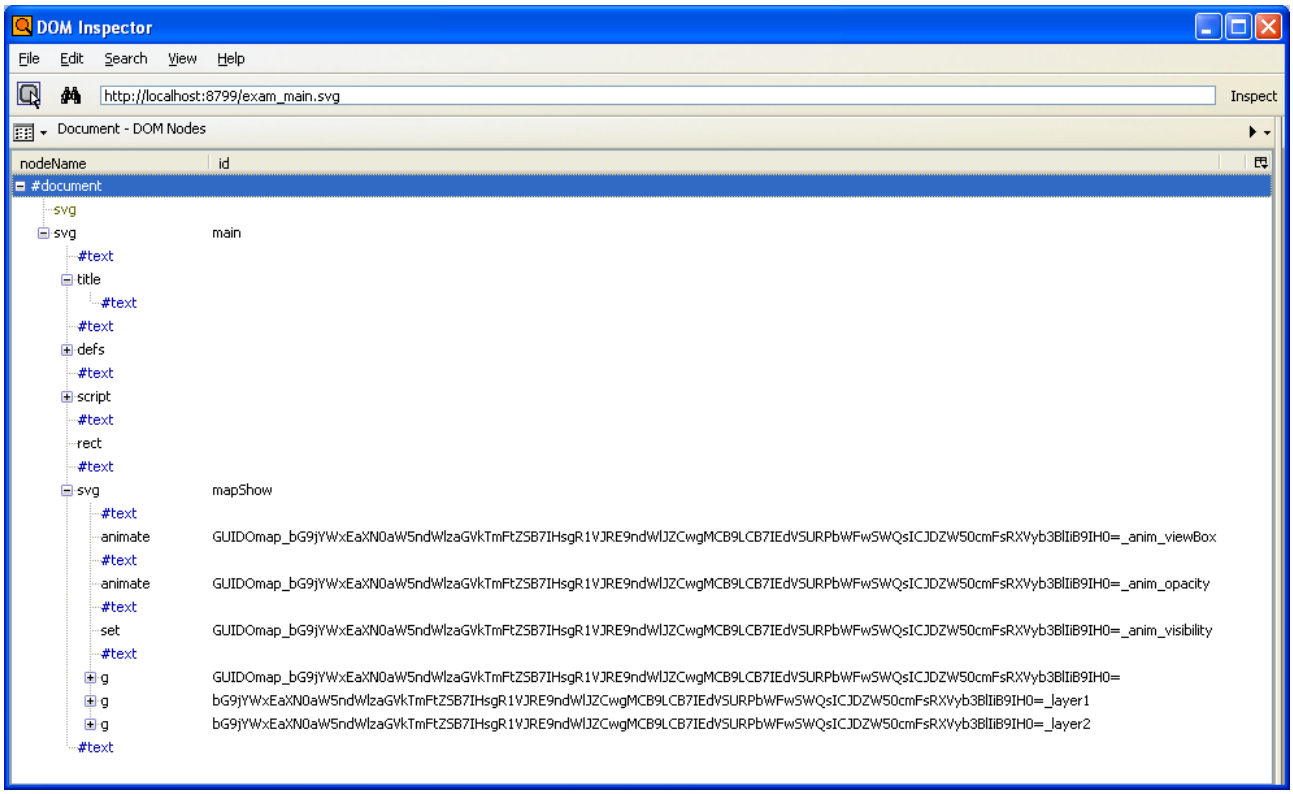
Tabel 6.1: Browserunderstøttelse (✓: fuld, ✓: delvis, - : ingen)

XML har efterhånden slået sig fast som en af de meste brugte standarder til data håndtering. Den er nem at bruge og ekstremt fleksibel. Derfor overrasker det heller ikke at alle browserne har en indbygget XML parser/viewer.

DO modellen har også været med i mange år, og udgør rygraden for at gøre hjemmesider mere dynamiske. Dynamisk HTML bygger fx på at man igennem DO modellen kan manipulere med forskellige elementer i et dokument. Den grundlæggende idé i DO modellen er at elementer eller indhold inkluderet i det modellen beskriver, skal kunne tilgås, ændres eller slettes gennem DO modellen. Samtidig gør DO modellens træstruktur at man kan se dokumentets struktur og opbygning. Figur 6.4 viser et simpelt dokument set ud fra DO modellen.

Pga. alle de udvidelser til fx HTML og nye standarder der langsomt bliver implementeret i de forskellige browsere findes der hos W3C efterhånden en hel del dokumenter om DO modellen. Derfor kan man ikke sige at nogen af browserne understøtter DO modellen fuldt ud. Samtidig er det næsten en umulig opgave at skulle sammenholde al dokumentationen for de forskellige browsere for bare at få et lille indblik i hvor meget de understøtter. Der findes fx både en DO model for almindelige (X)HTML sider, en for XML sider, en for SVG, osv. Grundlæggende har alle browserne mulighed for at gennem DO modellen at ændre dokumentets struktur og layout, og de indeholder også alle de samme primære hjælpefunktioner til at manipulere med DO modellen. Dvs. at grundidéen bag DO modellen altså findes i alle de udvalgte browsere, mens afvigelserne findes i mindre forskelligheder eller yderligere tilføjelser af fx script hjælpe funktioner.

Som det kan ses af tabel 6.1 er der ingen browsere der understøtter SVG 1.1 fuldt ud. Alle har mindre



Figur 6.4: Dokument struktur for prototypen af en webbaseret alarmmanager (DOM inspector i Mozilla Firefox)

afvigelser, eller elementer der ikke er implementerede. Med udgangspunkt i tabel 6.1 giver de følgende afsnit en dybere beskrivelse af hvad de enkelte browsere gør sig af mangler fra den fulde specifikation af SVG 1.1, samt status for understøttelse af de andre listede standarder.

For en komplet liste over hvilke SVG elementer de forskellige browsere understøtter, samt hvilke elementer der hører til de forskellige profiler (SVGT, SVGB og SVGF), se Bilag A på side 68.

6.3.1 Opera

Opera 9 understøtter XML fuldt ud og har en indbygget XML viewer med farvekodning. XML namespaces er også fuldt understøttet.

Opera 9 understøtter SVG 1.1 Basic og SVG 1.1 Tiny med enkelte undtagelser. Man kan med andre ord sige at den understøtter SVG 1.1 Full delvist.

Den understøtter event listening af alle events, men enkelte bliver ikke sendt til applikationen. Dette gælder fx `focusin`, `focusout` og `activate`. Eventsne `keyup` og `keydown` understøttes ikke da disse er specifikke for Adobes SVG viewer. Fonte er fuldt understøttet, også `font-family` parameteren, men hvis der mangler et tegn i den valgte font vælges et platform udvalgt tegn i stedet for at vælge det samme tegn fra den næste font i `font-family` parameteren.

Opera understøtter SVG dokumenter i `object`, `embed` og `iframe` i HTML, samt direkte åbning af et SVG dokument (fil med endelsen `.svg` eller `.svgz`). Opera understøtter dog ikke brug af SVG dokumenter som billede elementer (`<image>` element der peger på et SVG dokument), eller til brug i CSS parameter værdier (fx som baggrundsbillede). Dette er en generel ting for de browsere der

understøtter indlejret SVG, som opstår pga. at SVG dokumenter behandles som XML dokumenter og ikke som grafik. Et SVG `<image>` element kan dog indeholde alle understøttede raster grafik formater (GIF89a, JPEG, BMP, ICO, WBMP og PNG). Alle billeder (`<image>` elementer) i et SVG dokument, udskrives i Opera 9 som bitmap billeder.

Som tidligere nævnt bygger SVG på XML standarden og de samme regler for struktur og brug af namespaces gør sig derfor også gældende. Dvs. at alle tags skal åbnes og lukkes i den rigtige rækkefølge. Endvidere overholder Opera også SVG overensstemmelses reglerne der siger, at processeringen skal stoppes hvis parseren støder på en ukendt feature. Dette er selvfølgelig en nødvendighed der er taget med i SVG specifikationen, da SVG dokumenter ikke er prekompileret. Hvis man fx har glemt en skråstreg i et lukke tag forhindrer man på denne måde unødvendig brug af ressourcer på at forsøge at vise noget, der ikke kan vises.

XMLHttpRequest objektet er stadig ikke blevet en W3C Recommendation endnu, hvorfor Opera heller ikke kan siges at understøtte den fuldt. Opera understøtter dog brugen af XMLHttpRequest objektet baseret på de foreløbige specifikationer fra W3C.

Specifikationerne for Opera 9 indeholder ingen oplysninger om understøttelse af XForms.[9]

6.3.2 Firefox

Firefox 1.5 understøtter XML og har også en indbygget XML viewer. Den understøtter også brug af namespaces.

Firefox forsøger stadig at udvide deres understøttelse af SVG. Den næste version af Firefox, Firefox 2.0, skulle indeholde en meget bred implementering af SVG funktionalitet. Indtil videre er den eneste nye tilføjelse til den nuværende version, Firefox 1.5, tag'et `<textPath>`, mens en del mindre fejl i den nuværende understøttelse er blevet rettet.

Firefox's understøtter delvist SVG 1.1, men det tilnærmer sig ingen af de officielle profiler (SVG Tiny, SVG Basic eller SVG Full). Derfor er den delvise understøttelse listet i tabel 6.1 i parentes. Firefox understøtter i modsætning til Opera og Adobe ikke nogen form for animation. Heller ikke `<set>` elementet er understøttet, til brug ved fx `mouse-over` effekter. Ellers understøtter Firefox almindelig tegninger (`<rect>`, `<circle>`, osv.) lavet i SVG og også gradiente udfyldninger, m.m.

Som nævnt er XMLHttpRequest objektet stadig ikke blevet en W3C Recommendation endnu, hvorfor Firefox heller ikke kan siges at understøtte den fuldt. Firefox 1.5 understøtter dog brugen af XMLHttpRequest objektet baseret på de foreløbige specifikationer fra W3C.

Firefox 1.5 understøtter ikke XForms direkte, men Mozilla som står bag Firefox, har udviklet et plug-in til deres browsere. Plug-in'et hedder *Mozilla XForms Extension* og gør det muligt at benytte XForms i alle deres browsere.

6.3.3 Internet Explorer

Internet Explorer understøtter brug af XML. Den har en indbygget XML viewer med farvekodning og man kan folde elementer i XML dokumentet ind og ud. Den understøtter også brug af namespaces.

På trods af at Internet Explorer til dato er den mest udbredte browser, indeholder den nuværende version 6 ingen understøttelse af SVG. Den næste generation af Internet Explorer indeholder i beta

versionerne heller ikke nogen form for SVG understøttelse og Microsoft har heller ikke annonceret at Internet Explorer 7 kommer til at understøtte SVG.

Internet Explorer understøtter til gengæld muligheden for at benytte et plug-in til afvikling af SVG. Den mest benyttede SVG viewer er udviklet af Adobe. Bilag A.1 viser også at Adobe SVG Viewer 3.03 er den der understøtter mest af SVG specifikationen. Den indeholder også en script engine med DOM understøttelse, der gør det muligt at ændre direkte i SVG dokumentet via fx JavaScript. Endvidere har Adobe tilføjet nogle ekstra features, bl.a. funktionen `getURL()` der kan ses som et modstykke til `XMLHttpRequest`. Da SVG bygger på XML og DOM strukturerne, kan man ved hjælp af de to funktioner indhente et SVG fragment og "klistre" det på ens hoveddokument uden at skulle genindlæse hele siden. I UHCs tilfælde kunne dette fx være et vist oversigtskort hvor der tilføjes et ikon for en ny server.

Internet Explorer understøtter brugen af `XMLHttpRequest` objektet, dog som en activeX komponent i stedet for direkte som et JavaScript objekt. Microsoft har dog annonceret at version 7 af Internet Explorer skulle indeholde `XMLHttpRequest` som et JavaScript objekt på lige linie med Firefox og Opera.

Dokumentationen for Internet Explorer indeholder ligesom ved SVG ingen oplysninger om (fremtidig) understøttelse af XForms. I det hele taget er Microsofts information ret sparsom i forhold til Opera og Firefox. Det har fx ikke været muligt at finde en komplet liste over hvilke elementer indenfor de forskellige teknologier, Internet Explorer helt præcist understøtter. [7] & [8]

6.4 Use cases

En use case er en mere praktisk beskrivelse af en handling udført af en aktør. I en alarmmanager applikation findes følgende aktører:

- Alarmoperatør (bruger)
- Administrator
- Q3 ADE agent (server)

De følgende use cases beskriver forskellige handlinger der kan udføres af en alarmoperatør eller en administrator i en alarmmanager. Use casene bruges også i forbindelse med test af det komplette system, da det ofte vil være de her beskrevne handlinger man ønsker at teste. De er dog ikke benyttet til test formål i dette projekt, da prototypen ikke omhandler disse mere avancerede funktioner.

Use case oversigt:

Alle use case'ne er baseret på eksemplet i bogen *Applying UML and Patterns* [2].

6.4.1 Use case 1: Identificer og autoriser bruger

Characteristic information

- Goal in Context: At indentificere og godkende en bruger således at systemet kan benyttes
- Primary Actor: Bruger (Alarmoperatør, administrator)
- Scope: Bruger (Alarmoperatør, administrator), Q3ADE agent (server)
- Level: Primær funktion
- Preconditions: Q3ADE Agenten (Server delen) er startet. Der forefindes netværksforbindelse
- Success End Condition: Der blev oprettet forbindelse mellem klient og server, og brugeren blev logget på systemet (identificeret og godkendt)
- Failed End Condition: Der blev ikke oprettet forbindelse mellem klient og server. Brugeren blev ikke identificeret af systemet. Brugeren blev ikke godkendt af systemet. Brugeren blev ikke logget på systemet
- Trigger: Brugeren har behov for at tilgå information fra systemet

Main success scenario (or Basic Flow)

1 Der etableres netværksforbindelse til Q3ADE agenten gennem en browser

2 Den primære aktør præsenteres for identifikations process

3 Den primære aktør sender identifikation til serveren

4 De sendte oplysninger valideres af serveren

5 Hovedsiden vises

6 Systemet er klar til brug

Sub-variations (or Alternative Flows)

1a Der kan ikke etableres netværksforbindelse til serveren

1 Den primære aktør må tage kontakt til den netværksansvarlige

4a Validering fejler

1 Systemet afgiver en fejlmeddelelse

2 Den primære aktør præsenteres for identifikations processen igen

Related information

- Performance Target: $<$ 1 sek for systemet.
- Frequency: 1 gang dagligt eller efter behov (fx efter inactivity timeout)

6.4.2 Use case 2: Bekræft alarm

Characteristic information

- Goal in Context: At alarmoperatøren registrer og forholder sig til en indkommen alarm
- Primary Actor: Alarmoperatør
- Scope: Alarmoperatør
- Level: Primær funktion.
- Preconditions: Primær aktør er identificeret og godkendt af systemet (logget på). Der er etableret forbindelse til Q3ADE agenten
- Success End Condition: Alarmen blev registreret
- Failed End Condition: Alarmen blev ikke registreret
- Trigger: En alarm optræder i systemet (fx udsendt af et tilkoblet stykke udstyr)

Main success scenario (or Basic Flow)

1 Alarmen optræder i systemet

2 Alarmen sendes til klienten

3 Klienten præsenterer alarmen for den primære aktør

4 Den primære aktør klikker på knappen `Acknowledge` for at bekræfte alarmen

5 Den primære aktør forholder sig til hvordan alarmen skal behandles

Sub-variations (or Alternative Flows)

*a Alarmen ændrer status (severity)

1 Alarmen vises fortsat

2 Farven for alarmen ændres i forhold til alarmens status (severity)

4a Den primære aktør bekræfter ikke alarmen

1 Alarmen vises fortsat

Related information

- Subordinate Use Cases: Use case 1: Identificer og godkend bruger
- Performance Target: $<$ 1 sek. for systemet. Reel tid afhænger af primær aktørs reaktionstid
- Frequency: 0+ gange dagligt

6.4.3 Use case 3: Opret GUIDOmap/GUIDOmapComponent/GUIDOmapLink

Characteristic information

- Goal in Context: Administratoren opretter et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Primary Actor: Administrator
- Scope: Administrator, Q3ADE agent
- Level: Sekundær funktion.
- Preconditions: Primær aktør er identificeret og godkendt af systemet (logget på). Der er etableret forbindelse til Q3ADE agenten
- Success End Condition: Primær aktør får oprettet et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Failed End Condition: Primær aktør får ikke oprettet et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Trigger: Der opstår behov for at oprette et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink

Main success scenario (or Basic Flow)

1 Der etableres netværksforbindelse til Q3ADE agenten gennem en browser

2 Primær aktør vælger funktionen `Create map`

3 Programmet viser en parameter liste med tilhørende indtastningsfelter

4 Primær aktør indtaster parametre for det nye GUIDOmap/GUIDOmapComponent/GUIDOmapLink

5 Primær aktør vælger `Create`

6 Q3ADE agenten opretter et nyt GUIDOmap/GUIDOmapComponent/GUIDOmapLink på baggrund af de specificerede parametre

Sub-variations (or Alternative Flows)

*a Primær aktør har været inaktiv for længe og logges af systemet

1 Primær aktør logger på systemet og starter forfra

4a Primær aktør indtaster ikke valid data

1 Programmet udskriver en fejlmeddelelse
2 Primær aktør retter fejlen

Related information

- Subordinate Use Cases: Use case 1: Identificer og godkend bruger
- Performance Target: $<$ 1 sek. for systemet. Reel tid afhænger af primær aktørs indtastningstid
- Frequency: 0+ gange dagligt

6.4.4 Use case 4: Rediger GUIDOmap/GUIDOmapComponent/GUIDOmapLink

Characteristic information

- Goal in Context: Administratoren redigerer et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Primary Actor: Administrator
- Scope: Administrator, Q3ADE agent
- Level: Sekundær funktion.
- Preconditions: Primær aktør er identificeret og godkendt af systemet (logget på). Der er etableret forbindelse til Q3ADE agenten
- Success End Condition: Primær aktør får redigeret et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Failed End Condition: Primær aktør får ikke redigeret et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Trigger: Der opstår behov for at redigere et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink

Main success scenario (or Basic Flow)

1 Der etableres netværksforbindelse til Q3ADE agenten gennem en browser

2 Primær aktør vælger Editeringstilstand

3 Primær aktør vælger det GUIDOmap, den GUIDOmapComponent, eller det GUIDOmapLink der skal redigeres

4 Primær aktør foretager de fornødne ændringer

5 Primær aktør vælger funktionen `Apply changes`

6 Q3ADE agenten gemmer ændringerne

Sub-variations (or Alternative Flows)

2a Der skiftes ikke til Editeringstilstand

1 Programmet udskriver en fejlmeddelelse

2 Primær aktør vælger Editeringstilstand

Related information

- Subordinate Use Cases:
  Use case 1: Identificer og godkend bruger
  Use case 3: Opret GUIDOmap/GUIDOmapComponent/GUIDOmapLink
- Performance Target: $<$ 1 sek. for systemet. Reel tid afhænger af primær aktørs indtastningstid
- Frequency: 0 $<$ gange dagligt

6.4.5 Use case 5: Slet GUIDOmap/GUIDOmapComponent/GUIDOmapLink

Characteristic information

- Goal in Context: Administratoren sletter et et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Primary Actor: Administrator
- Scope: Administrator, Q3ADE agent
- Level: Sekundær funktion.
- Preconditions: Primær aktør er identificeret og godkendt af systemet (logget på). Der er etableret forbindelse til Q3ADE agenten
- Success End Condition: Primær aktør får slettet et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Failed End Condition: Primær aktør får ikke slettet et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink
- Trigger: Der opstår behov for at slette et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink

Main success scenario (or Basic Flow)

1 Der etableres netværksforbindelse til Q3ADE agenten gennem en browser

2 Primær aktør markerer et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink

3 Primær aktør vælger funktionen `Delete`

4 Q3ADE agenten sletter det pågældende GUIDOmap, den pågældende GUIDOmapComponent, eller det pågældende GUIDOmapLink

Sub-variations (or Alternative Flows)

2a Primær aktør markerer ikke et GUIDOmap, en GUIDOmapComponent, eller et GUIDOmapLink

1 Systemet foretager ingen ændring

Related information

- Subordinate Use Cases:
  Use case 1: Identificer og godkend bruger
  Use case 3: Opret GUIDOmap/GUIDOmapComponent/GUIDOmapLink
- Performance Target: $<$ 1 sek. for systemet. Reel tid afhænger af primær aktørs indtastningstid
- Frequency: 0+ gange dagligt

6.5 Kravspecifikation

UHCs kundekreds består af internationale kunder og alt deres materiale er derfor fremstillet på engelsk. Da UHC havde et ønske om evt. at bruge denne kravspecifikation senere hen, er denne også skrevet på engelsk.

Alarmmanager

1 General requirements for the Alarmmanager

1.1 The Alarmmanager must contain an Event channel where it can retrieve events from the server

1.2 The Alarmmanager must be able to act on events recieved from the server

1.3 All alarms must be colored alike according to the severity of the alarm i.e. there must be color consistency in all parts of the Alarmmanager

1.4 A user only has to log in once to access the Alarmmanager (Single Point Authentication)

1.5 A user shall have the posibillity to activate an inactivity timeout (the system logs the user off after a specified time with no activity)

2 Hierarchical viewer

2.1 The Hierarchical Viewer shall present a hierarchical (tree) view of all maps in the system

2.2 The user shall be able to select either visible or hidden for all underlying MapComponents and MapLinks under each Map

2.3 If a Map is selected in the Hierarchical Viewer the corresponding Map shall be presented in the Map Viewer

3 Topological viewer (MapViewer)

3.1 The MapViewer shall contain interactive drill-down map views

3.2 A Map can consist of MapComponents and MapLinks

3.2.1 MapComponents must be able to reflect an alarmstate given by the equiptment it is tied to

3.2.2 MapLinks must be able to reflect an alarmstate given by the equipment it is tied to

3.3 MapComponents and MapLinks that are clickable shall appear clickable

3.4 The MapViewer shall mark alarmstate changes so that they appear noticeable (i.e. flash 5 times)

3.5 The MapViewer shall be able to present Maps in layers

3.6 A layer shall consist of MapComponents and MapLinks

3.7 The MapViewer shall contain user controls for hiding / showing the different layers

3.8 MapComponents / MapLinks that have an alarm must always be visible, even if the layer that contains the component is hidden

3.9 The MapViewer shall contain a poll function in order to:

3.9.1 Retrieve information from the server about a MapComponents/MapLinks position

3.9.2 Retrieve information from the server about a MapComponents/MapLinks alarm state

3.9.3 Retrieve information from the server about a MapComponents/MapLinks size

3.9.4 Retrieve information from the server about a MapComponents/MapLinks image (icon)

4 Alarm log

4.1 The user shall be able to acknowledge alarms

4.1.1 When a user acknowledges an alarm the user information must be registered with the acknowledgement

4.2 Timestamps shall be shown in both local (client) time and reference time (e.g UTC)

4.3 All alarms must be presented in the alarmlog within no more than 1 second from occurrence (disregarding network transmission time)

4.4 The Alarm log shall contain an overall alarm summary

4.5 The Alarm log shall be able to present a detailed view of the individual alarm

5 Maintenance

5.1 The Alarmmanager shall contain an Edit mode

5.1.1 In Edit mode the user can create new MapComponents / MapLinks on any Map

5.1.2 In Edit mode the user can delete existing MapComponents / MapLinks on any Map

5.1.3 In Edit mode the user can edit existing MapComponents / MapLinks on any Map

5.1.4 In Edit mode the user can move existing MapComponents / MapLinks on any Map

6 Requirements related to a Web Edition

6.1 The user shall be given the option that all transfers between the server and the client is done over a secure line using SSL (Secure Socket Layer)

6.2 The Alarmmanager Web Edition shall be able to load in a given browser without any additional plug-in installation

6.3 The Alarmmanager Web Edition must contain cross platform browser support

6.4 The Alarmmanager Web Edition shall be able to load under the browsers Internet Explorer, Opera and Firefox

6.6 Indtastningsmuligheder

Pga. XForms ringe udbredelse browserne imellem, er der ikke lavet nogle eksempler baseret på XForms. Mozilla er indtil videre de eneste der har lavet et plug-in der understøtter en lille del af XForms. Men eftersom størstedelen i dag benytter Microsofts Internet Explorer, og denne ikke indeholder noget information om hvorvidt Microsoft i fremtiden ønsker at understøtte XForms, må XForms indtil videre siges at indeholde potentialet til at blive noget man ser meget mere til i fremtiden, men i nuværende stund ikke kan benyttes til noget brugbart i dette projekt.

Hvornår og om XForms bliver en kendt og udbredt standard er svært at sige noget om. Kigger man på SVG til sammenligning, kan man sige at SVG har eksisteret som W3C Recommendation siden januar 2003. Dvs. at det har taget mere end 3 år nå til hvor man er i dag. XHTML 2.0 som det pt er meningen XForms skal være formular standard for, har stadig kun status af en "working draft", dvs. at *forslaget* til en endelig specifikation stadig er under udarbejdelse. Så heller ikke her kan man finde en lille indikation af hvornår det kunne være en mulighed at se en XForms i brugbar stand.

Mit bedste bud vil være at hvis markedet ser mulighederne i XForms, med gode muligheder for tilpasning af formularer på alle enheder, nem udveksling af data, indbygget fejlhåndtering, osv. går der stadig mindst et par år før vi ser det taget i brug.

Det fjerner i midlertidigt ikke behovet for at en alarmoperatør kan indtaste data i en webbaseret alarmmanager. Det mest lige til alternativ er at benytte almindelige HTML formularer. Når alarmoperatøren skal have mulighed for at indtaste data, må man åbne et nyt midlertidigt vindue med en standard HTML formular, hvor alarmoperatøren så kan udfylde en række indtastningsfelter og sende data til serveren.

Alle browsere har i dag et eller andet form for pop-up blokeringsværktøj. Så allerede her kan man komme i problemer når et indtastningsvindue skal åbnes for at alarmoperatøren kan komme til at indtaste data. Problemet er ikke større end at man kan give tilladelse til pop-up vinduer, men hvis udgangspunktet er at man skal kunne koble sig på og se oplysninger om systemet fra en hvilken som helst computer der har forbindelse til netværket, ligger det lige på grænsen af at gå fra et irritationsmoment til et problem.

En formular standard der arbejder bedre sammen med SVG vil i høj grad forbedre selve bruger oplevelsen. Ønsket om at lave noget der kommer så tæt på en applikation, frem for en hjemmeside, eller sagt på en anden måde, at lave en hjemmeside der ligner og opfører sig som et computerprogram. Almindelig program elementer som fx et tree-view, en tabel hvor man kan sortere kolonner stigende/faldende og ændre bredden på kolonner, værktøjslininer, osv. er ting man vil være nødt til at lave vha. JavaScript og HTML til en web applikation. Det er muligt at lave men det er besværligt, og man vil i sidste ende være nødt til at tilpasse mange af disse kontroller til specifikke formål. Samtidig vil man også kunne lave flottere elementer i SVG og XForms, end de standard elementer der findes i almindelige formularer i dag. Samtidig vil XForms håndtering af data separat fra præsentationen også gøre at man vil kunne benytte samme data håndtering, men lave individuelt tilpassede brugergrænseflader til forskellige kunder nemmere end ellers.

Som en sidste mulighed på en løsning til indtastning af data, kunne man lave sine egne SVG formular elementer. Derved kunne man lave en form for dialogbokse i hoved SVG dokumentet, så man undgår pop-up vinduer og bruger oplevelsen igen bliver mere lig med en applikation. For at kunne lave sådanne SVG formular elementer kræver det at man kan reagere på keyboard input. Indtil videre er events som `keyup` og `keydown` specifikke for Adobes SVG Viewer, så denne løsning kan altså indtil videre ikke leve op til kravet om en web applikation der kan køre på flere platforme / i forskellige browsere.

6.7 Eksempler

Da en fuld funktionsdygtig webbaseret alarmmanager er en større kompliceret opgave, og når den samtidig forsøges bygget af fortrinsvis nye teknologier, må analyse delen nødvendigvis også indeholde nogle relevante eksempler på formålet med de enkelte teknologier i forhold til en webbaseret alarmmanager.

De følgende afsnit giver eksempler på brugen af de udvalgte teknologier og beskriver brugbarheden af de enkelte teknologier.

6.7.1 SVG eksempler

Ikoner og mouse-over effekt

For at bevæge sig virtuelt rundt mellem de forskellige GUIDOmaps i MapVieweren, kan man definere hvad der skal ske når man klikker på forskellige elementer. Udgangspunktet vil ofte være en form for drill-down kort, hvor man fx starter med et oversigts kort og herefter kan klikke sig videre ned til de enkelte enheder. Fra Internettet er man vant til at links bliver markeret på den ene eller den anden måde. Samme effekt ønskes her således at man kan indikere hvad man kan klikke på for at komme længere ned (eller op) i hierarkiet. Figur 6.5 viser et SVG billede (vist for alle 3 browsere) der gør brug af SVGs indbyggede animationseffekter (her tag'et `<set>`), der her er sat til at ændre farven på ikonet når musen flyttes henover det.

En anden vigtig ting illustreret i dette eksempel er brugen af symboler (eller ikoner om man vil, i SVG specifikationen kaldes det et *symbol*). SVG har indbyggede muligheder for at definere et symbol (dvs. en hvilken som helst tegning), som man så kan genbruge et uendeligt antal gange. I eksemplet er ikonet defineret som et symbol og herefter genbrugt. Symboler defineres i `<defs>` sektionen og man indsætter et symbol vha. `<use>` elementet. Tanken bag dette er at man fx kunne lave et ikon bibliotek så en kunde kan genbruge sine ikoner, eller købe sig til færdiglavede ikoner.

Som det kan ses af figur 6.5 understøtter Firefox ikke brug af `<set>` elementet (se tabel A.1 på Bilag A), hvorfor ikonet ikke ændres.

Figur 6.5 viser et simpelt eksempel på genbrug af et symbol. En kunde kunne fx have et bestemt symbol for en router (fx som her den grønne cirkel) og så genbruge dette på et oversigtskort over alle deres routere. I praksis ønsker man i de fleste tilfælde at udnytte alle SVGs muligheder og oprette symboler/ikoner med skyggeeffekter, gradiente farvelag, osv. Figur 6.6 viser et eksempel på brug at et mere detaljeret symbol. Computeren er tegnet med brug af gradiente farvelag, hvilket er med til at gøre symbolet mere 3-dimensionelt. Ligesom før virker hover effekten (computeren i højre side af figur 6.6) kun ved visning med Opera og Adobe SVG viewer.

Figur 6.7 viser et eksempel på "genbrug" af det detaljerede ikon (computeren, se figur 6.6). I alt er der vha. et lille server script tegnet 180 computere. Det tager lidt tid at rendere og vise de mange symboler. Dog er alle tre browsere nogenlunde samme tid om at vise siden med de 180 computer symboler.

En af de vigtigste grunde til at benytte SVG er skalerbarheden. Tegninger i SVG formatet skulle altså kunne vises i alle størrelser uden tab af kvalitet. Som det ses på figur 6.8 er der intet tab af kvalitet uanset zoom niveau. En anden vigtig egenskab ved SVG er muligheden for at lave animationer. Ikke bare som i dette eksempel, hvor et billede ændres en lille smule, men animerede alarmer, flytning af en SVG tegnet bil ved opdatering af GPS koordinater, zoom og overblænding ved skift i mellem GUIDOmaps, osv.

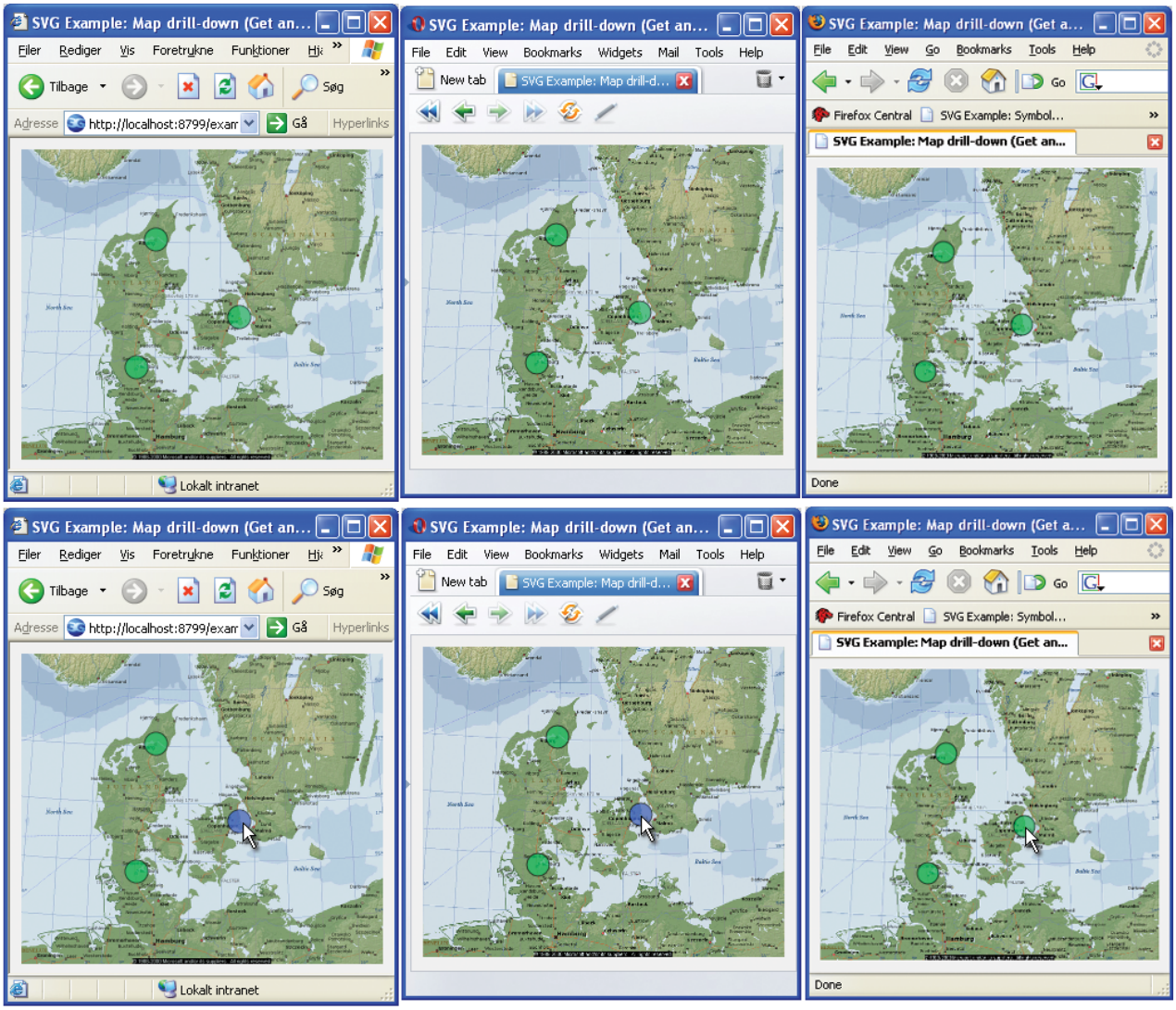
For kildekode til alle SVG eksempler, se Bilag B.

6.7.2 XMLHttpRequest eksempler

XMLHttpRequest'et kan bruges til at indhente data i baggrunden uden at man behøver at opdatere en hel side.

Indlæs og tilføj SVG data fra server

Det følgende eksempel viser et oversigtskort over Europa. Klikker man på ikonet (den grønne cirkel) ved Danmark indlæses SVG filen `exam_svg_server_response.svg` vha. et XMLHttpRequest objekt.

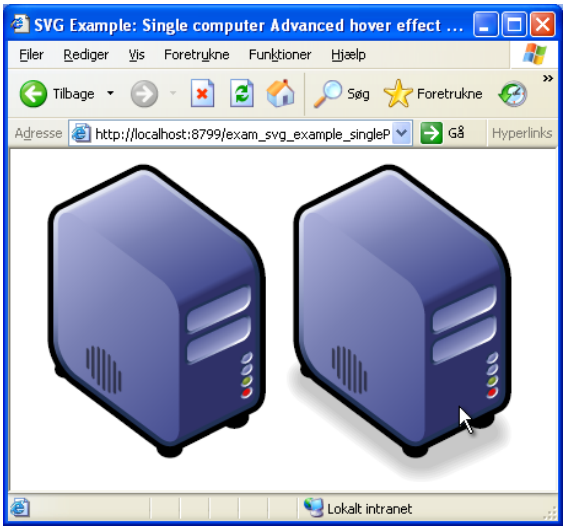


Figur 6.5: SVG eksempel - Ikon med hover effekt (Internet Explorer(ø), Opera(m) og Firefox(n))

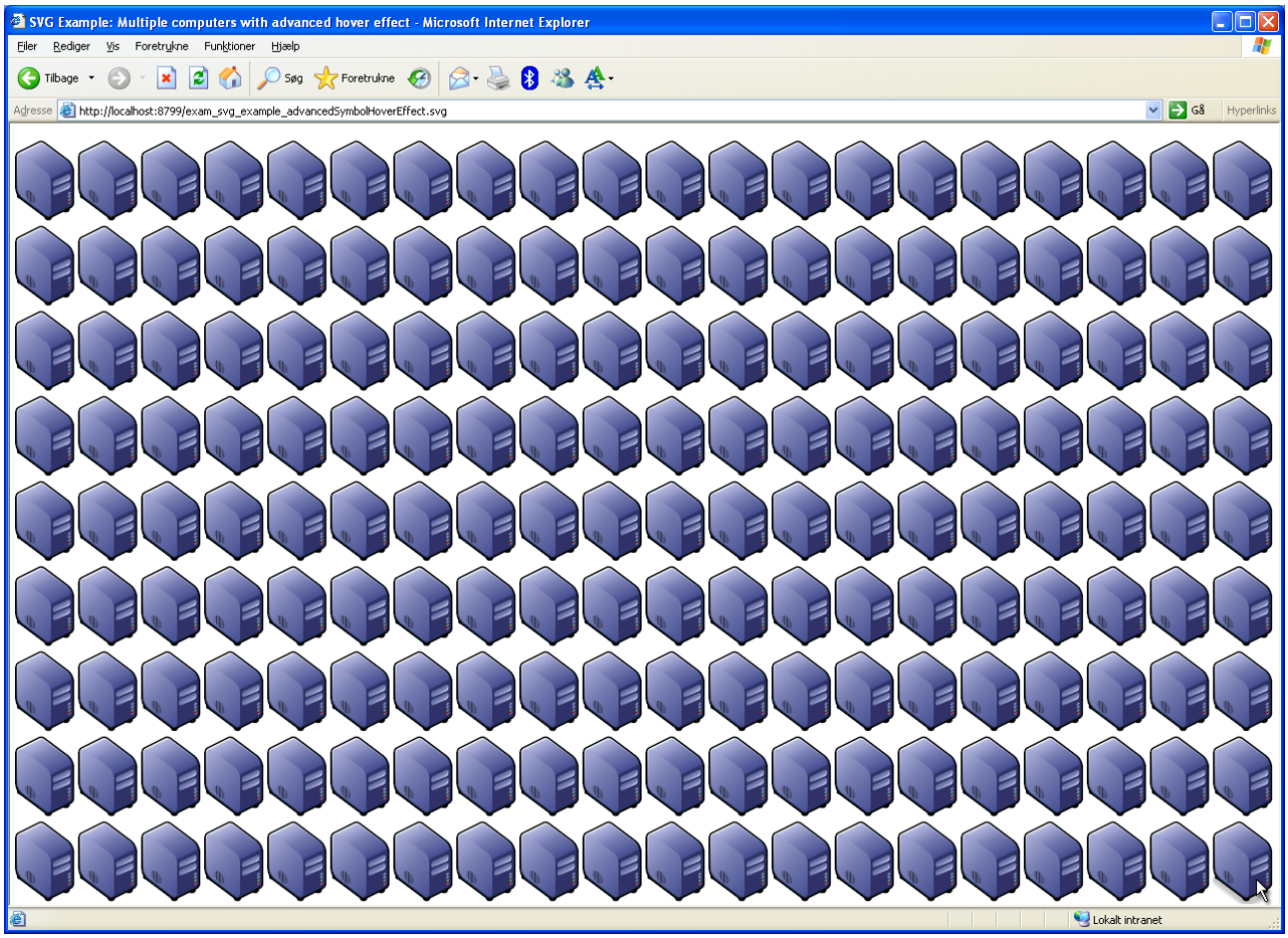
Filen `exam_svg_server_response.svg` indeholder et baggrundsbillede (oversigtskort over Danmark) og 3 ikoner. Dette tilføjes (dvs. det bliver det forreste lag og dermed skjuler alt andet) til dokumentet og det nye billede vises nu.

Microsoft har indtil videre implementeret XMLHttpRequest objektet som en ActiveX komponent, mens Opera og Firefox har det implementeret som et JavaScript objekt. Derfor er man nødt til at skelne mellem hvilken browser siden afvikles i. Funktionen `GetXmlHttpObject()` tester hvilken browser man bruger og returnerer et XMLHttpRequest objekt der passer til den pågældende browser på baggrund af det.

For at kunne sætte det originale dokument sammen med den indhentede data, skal den nye data først omdannes til brugbar XML. Igen har Microsoft valgt at implementere deres XML parser som en ActiveX komponent, men Adobes SVG Viewer indeholder en XML parser, `parseXML`, som er meget nemmere at benytte, så den er brugt i stedet. For at Adobes funktion parser SVG dataen til noget



Figur 6.6: SVG eksempel - Detaljeret ikon. Det højre ikon vist med mouse-over effekt



Figur 6.7: SVG Eksempel - Genbrug af detaljeret ikon

Figur 6.8: SVG Eksempel - Zoom på SVG tegnede ikoner sker uden tab af detaljer

brugbart, indeholder den en ekstra parameter. Denne parameter angiver et reference dokument, som den så benytter under parsningen. Funktionen `str2XML(str)` tager en streng som parameter, parser strengen til XML ud fra hvilken browser der benyttes og returnere et XML dokument fragment. Dette kan så tilføjes til hoveddokumentet. Figur 6.9 viser resultatet i de forskellige browsere. Billederne øverst viser SVG dokumentet før indlæsning af ikonerne og baggrundsbilledet, mens de nederste viser SVG dokumentet efter indlæsning af data.

Map Pollning

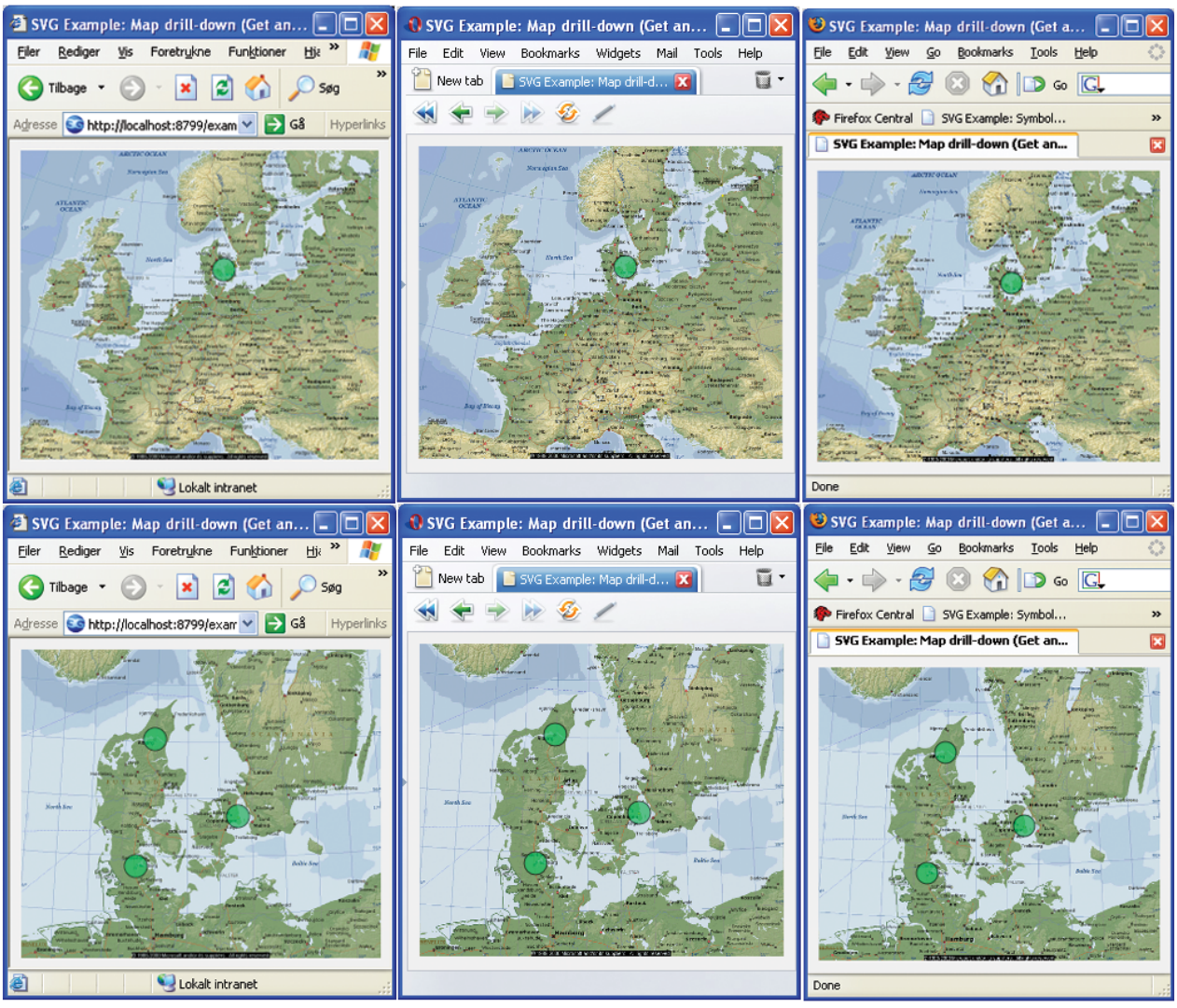
I kravspecifikationens krav 3.9 gør det sig gældende, at der skal være en funktion, der fra klientens side, går ned og henter information fra serveren. Poll funktionen er styret af en timer der gemmes i variablen `g_pollTimeOut`. Denne variabel sættes lig et timeout hver gang funktionen kaldes. Ønsker man at stoppe med at polle kan man cleare denne timer og sætte variablen `g_pollState` til værdien `false`. Ellers vil funktionen kører rekursivt i ring og udføre pollningen, så længe `g_pollState` er sat til `true` og der kommer et timer event.

Et eksempel på hvorfor man ønsker at kunne stoppe pollningen igen, er når der fx indlæses et nyt GUIDOmap. Her skal man kunne stoppe pollningen indtil det nye GUIDOmap er hentet ind. Herefter starter pollningen igen, men spørger nu efter data fra det nye GUIDOmap.

Den rekursive funktion der sørger for at data bliver indhentet med det specificerede tidsinterval (`g_pollTime`) vises her:

```
1  var g_pollState = false;
2  var g_pollTimeOut = null;
3  var g_pollTime = 10000; // miliseconds
4
5  function activatePolling(){
6    if(g_pollState == true){
7      pollMap();
8      g_pollTimeOut = setTimeout("activatePolling()", g_pollTime );
9    }
10 }
```

Funktionen `pollMap()` sørger for den egentlige pollning. Den opretter et `XMLHttpObjekt` på samme måde som i eksemplet med indlæsning af data og går ned og henter den egentlige data fra serveren.



Figur 6.9: Før/efter indlæsning og tilføjelse af SVG data (IE, Opera og Firefox)

Kapitel 7

Design

7.1 Overordnet design

7.1.1 Connection diagram

Figur 7.1 viser et Connection diagram over hvordan de forskellige dele kommunikerer med hinanden. Q3ADE agenten er knudepunktet i netværket. Den indsamler information om alt hvad der er tilkoblet netværket. Man kan så som alarmoperatør koble sig på agenten og udtrække information om netværkets forskellige dele. I dag sker dette via UHCs Q3Browser (vist her som en Java klient).

Design afsnittet beskriver opbygningen af en browser applikation til at udføre samme funktionalitet som i den nuværende Java klient. Dvs. at man altså kan koble sig på systemet og få præsenteret data vha. en webbrowser i stedet for gennem et Java program.

Q3ADE agenten sørger for al kommunikation mellem udstyr på netværket og en tilkoblet klient. Dvs. at hvis man som alarmoperatør ønsker at se information om netværket, vil det være agenten der præsenterer netværket på baggrund af den information agenten har indhentet. En klient kan altså aldrig kommunikere direkte med et stykke udstyr, eftersom det er agenten der indeholder alle informationsmodellerne til at udføre kommunikation med tilkoblet udstyr (se figur 6.1 på side 18 for processen fra fysisk netværk til komponenter benyttet til kommunikation i Q3ADE agenten).

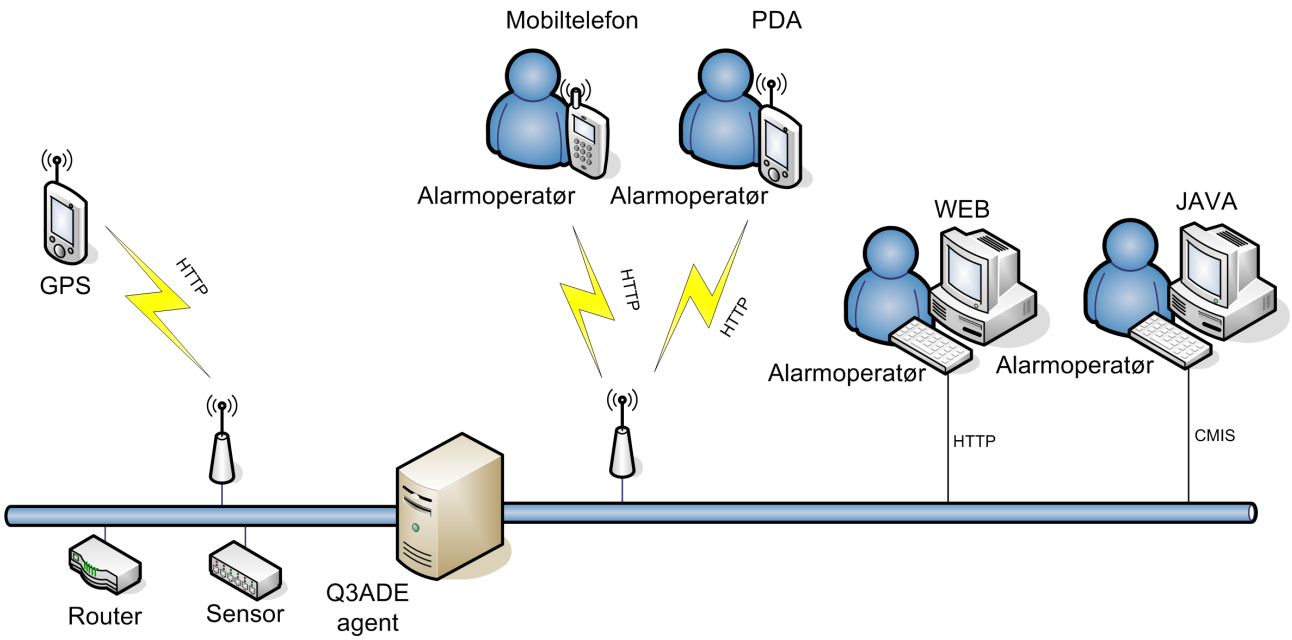
Selve netværksopbygningen kan variere meget fra system til system. Figur 7.1 viser blot et eksempel på et netværkssystem. Q3ADE agenten kan kommunikere via en række forskellige kommunikationsprotokoller og kommer med en række indbyggede komponenter, der gør den i stand til at overvåge og udføre management på de fleste stykker udstyr.

7.2 Brugergrænseflade

På baggrund af kravspecifikationen kan tre forskellige dele af brugergrænsefladen identificeres:

- Hierarkisk visning af tilgængelige GUIDOmaps
- Grafisk visning af et valgt GUIDOmap
- Alarm log til visning af opståede alarm events

Figur 7.2 viser hvordan brugergrænsefladen til en fuld funktionsdygtig alarmmanager tænkes udfærdiget. Da designet dækker en web applikation, vil præsentationsværktøjet være en browser. Browserens



Figur 7.1: Connection diagram

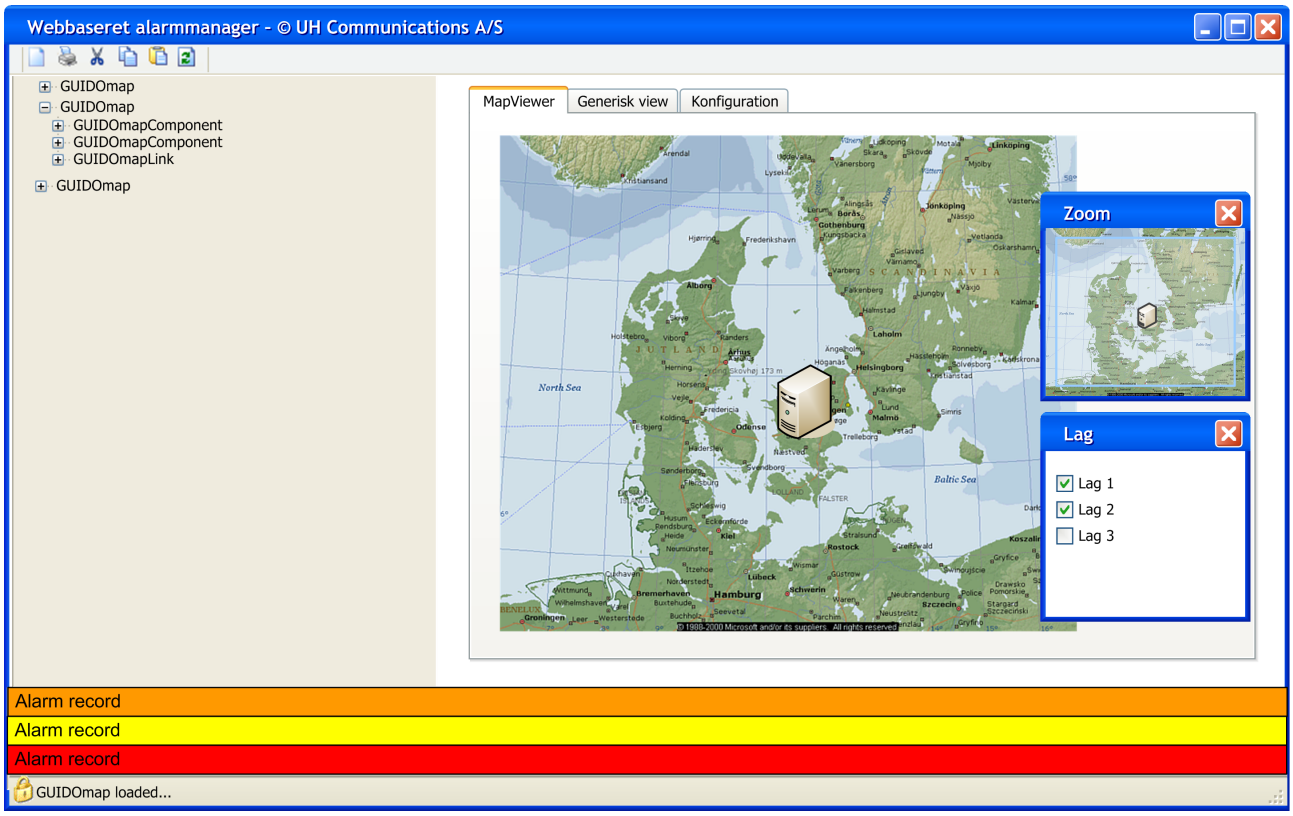
egne værktøjslinier skal skjules og erstattes af en til den webbaserede alarmmanager tilhørende værktøjslinie.

Den hierarkiske oversigt skal fungere som en slags menu, da det også er her man skal vælge, hvilket GUIDOmap der skal vises. Den hierarkiske oversigt skal opbygges og vises som en træstruktur, hvor man har mulighed for at folde underliggende komponenter ud eller ind ved de viste GUIDOmaps.

Som hovedvindue benyttes det grafiske præsentations vindue. Her skal selve det valgte GUIDOmap vises. Al interaktivitet med et valgt GUIDOmap sker her, dvs. hvis et GUIDOmap indeholder en reference der fører videre til et andet GUIDOmap, skal animationer m.m. ved skift imellem de to GUIDOmaps, vises i dette vindue.

Nederst i det samlede vindue skal der være en alarm log. Alarmlog'en skal udfærdiges som en almindelig applikations tabel, dvs. man skal have mulighed for at sortere de enkelte kolonner stigende eller faldende, man skal kunne tilpasse størrelsen af en kolonne, man skal kunne markere en eller flere rækker i tabellen, samt udføre en given funktion på en eller flere valgte rækker (fx se detaljer, eller i tilfældet med alarmlog'en bekræfte en alarm i hht. Use case 2).

Brugergrænsefladen skal indeholde mulighed for valg af tilstand. Dvs. der skal kunne skiftes til en editeringstilstand, således at en bruger har mulighed for at redigere et GUIDOmap i hht. Use case 3, 4, og 5. Brugeren skal herefter have mulighed for at vende tilbage til normal tilstand, og derefter igen kunne navigere sig rundt imellem de forskellige GUIDOmaps som før. Værktøjslinien skal indeholde knapper til skift mellem tilstandene.



Figur 7.2: Brugergrænseflade for webbaseret alarmmanager

Brugergrænsefladen skal indeholde kontroller til at skjule/vise de forskellige lag i et GUIDOmap. Disse kontroller skal samles i dialogboks, dvs. man skal kunne flytte dem rundt som en samlet enhed på brugergrænsefladen. På samme måde skal der være et zoom vindue, hvorfra man hele tiden har et komplet oversigtsbillede over det samlede GUIDOmap, samt en indikator af hvad der vises i hovedvinduet (fx hvis man er zoomet tæt ind GUIDOmap'et).

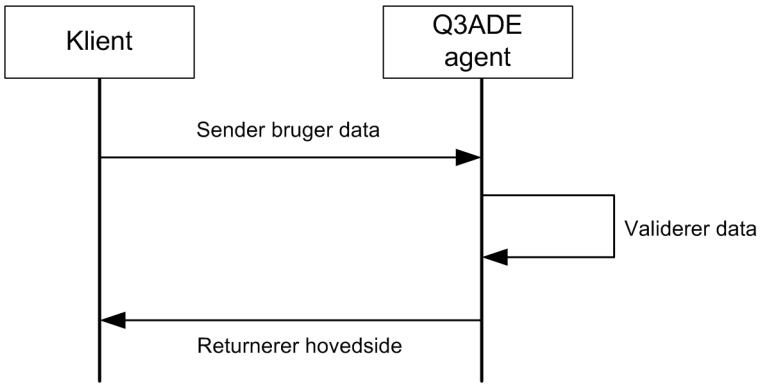
7.3 Sikkerhed

I henhold til kravspecifikationens krav 6.1 skal brugeren have mulighed for at vælge at al kommunikation foregår over en sikker netværksforbindelse sikret vha. SSL (Secure Socket Layer). Denne funktion skal kunne slås til eller fra via en samlet konfigurationsdialog til den webbaserede alarmmanager.

Før information fra Q3ADE agenten kan hentes og vises, skal en bruger foretage log-ind. Dvs. han med et unikt brugernavn og en tilhørende adgangskode, skal tilkendegive at han har rettigheder til at udføre forskellige funktioner i den webbaserede alarmmanager.

Figur 7.3 er et flowdiagram der viser data strømmen ved et bruger log-ind. Flowdiagrammerne viser hvordan informationerne i netværket sendes rundt. De udgør ikke protokoller, men viser kun de vigtigste informationer for at give en bedre forståelse af de forskellige funktioner.

Brugeren indtaster sine login data til klienten, der sender dem til agenten. Agenten står så for valideringen af de sendte oplysninger og returnere hovedsiden, hvis data valideres gyldig.

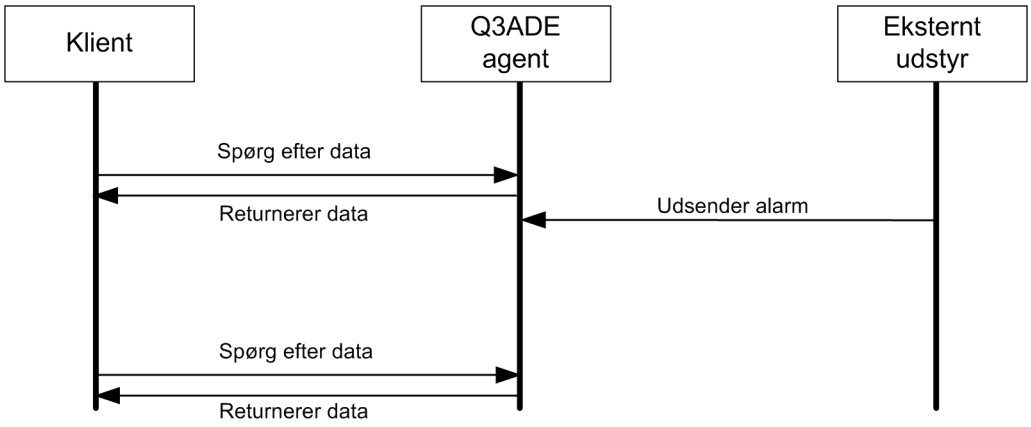


Figur 7.3: Flowdiagram for login

7.4 Poll funktion

Figur 7.4 viser flowdiagrammet for poll funktionen. I kravspecifikationens krav 3.9 gør det sig gældende, at der skal være en funktion, der fra klientens side, går ned og henter information fra agenten. Her er vist kommunikationsgangen ved to pollninger. Klienten udsender en forespørgsel efter data fra agenten og agenten returnerer den forespurgte data.

Der skal være mulighed for at starte og stoppe pollningen, fx ved skift mellem GUIDOmaps. Endvidere skal der være en indikator der viser den nuværende poll status. Brugeren skal også kunne angive tidsintervallet hvormed pollningen sker for det enkelte GUIDOmap.



Figur 7.4: Flowdiagram for poll funktionen

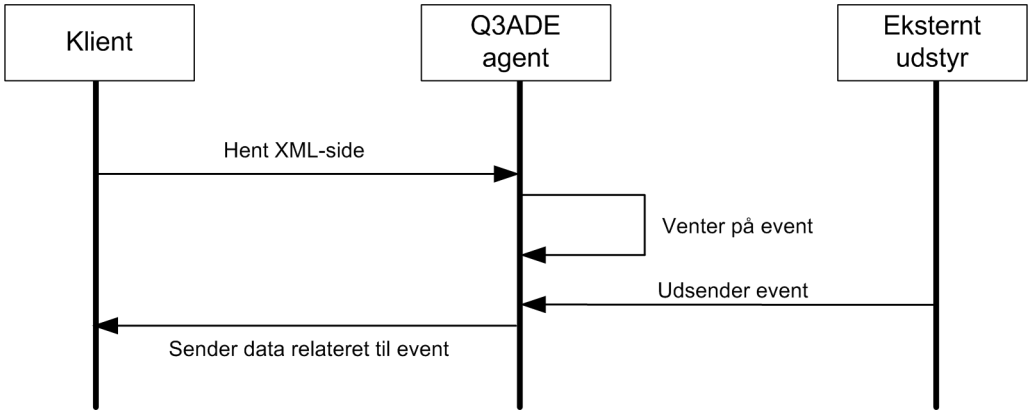
7.5 Event kanal

Når agenten modtager et event fra et stykke udstyr, reagerer agenten på eventet og sender et CMIS event videre til klienten. Klienten kan så opdatere sine data ud fra det indkomne event. I kravspecifikationens krav 1.1 og krav 1.2 gør det sig gældende at der skal være en Event kanal, således at man kan modtage events fra Q3ADE agenten (serveren).

Eventkanalen adskiller sig fra poll funktionen. Ved poll funktionen er det klienten der går ned og beder om data med et specificeret tidsinterval, mens eventkanalen skal stå og lytte efter events fra agenten og kunne reagere omgående på indkomne events. Sammenligner man figur 7.4 med figur 7.5

kommer denne forskel til udtryk i afstanden i mellem at et event optræder og at det sendes til klienten. Ved pollning kan eventet optræde lige efter at data er blevet returneret til klienten. Reaktionen på eventet kan altså først ske ved den næste pollning, hvor event data så vil blive returneret med til klienten. I alarmmanagement systemer er det ikke ualmindeligt at en alarmevent udsendt fra et stykke udstyr, skal præsenteres for en bruger inden for 1 sekund. Derfor er eventkanalen vigtig, da man her har mulighed for sende eventet, der når agenten, videre til klienten med det samme. Samtidig vil en meget lav poll frekvens resultere i al for meget netværkstrafik og belaste både klienten og agenten unødvendigt.

En udokumenteret lille test har vist at det er muligt vha. et oprettet `XMLHttpRequest` objekt at forsøge at hente en side fra agenten, der først returnes efter et godt stykke tid. Siden indeholdt en `sleep` funktion sat til at vente i ca. 20 min, hvorefter informationen på siden blev genereret og sendt tilbage til klienten. Man kan altså få klienten til at stå og vente i baggrunden. Når et event så optræder, genererer agenten den side som klient forespurgte, med information om eventet. Klienten kan så reagere på eventet og starte eventkanalen igen, dvs. sætte sig til at vente på at et nyt event optræder i systemet.



Figur 7.5: Flowdiagram for event kanal

7.6 Ikon bibliotek

Da netværk ofte består af mange af de samme enheder, skal man som bruger have mulighed for at genbruge sine ikoner. Her menes specifikt SVG tegnede ikoner og ikke raster grafik ikoner. Disse kan man genbruge ved blot at angive stien til billedet ens på sine GUIDOmapComponents. SVG ikoner kan være en lang række tegnedefinitioner der til sammen udgør et skalerbart, detaljeret ikon (se fx computeren på figur 6.6 og figur 6.8).

Det skal være muligt at tilføje ikoner til et samlet bibliotek. SVG dokumentationen indeholder muligheder for eksterne referencer. Dvs. man kan tegne et SVG billede, gemme det som en fil og så i fx et hoved SVG dokument blot indsætte stien til den fil der indeholder ens SVG ikon tegning (altså på samme måde som med almindelig rastergrafik indsat via et `<image>` element).

Desværre indeholder ingen af browserne mulighed for eksterne referencer i forbindelse med indsætning af SVG grafik i et eksisterende SVG dokument.

Designet af et ikon bibliotek skal altså i første omgang baseres på, at agenten skal indeholde en funktion der finder frem til alle SVG baserede ikoner på det pågældende GUIDOmap og sørger for at kildekoden til de fundne ikoner inkluderes (én gang for hvert benyttet ikon!) i hoveddokumentets (det første der genereres når man indlæser et nyt GUIDOmap) `<defs>` afsnit. Herefter kan man så

benytte `<use>` elementet til at placere ikonerne (eller GUIDOmapComponents som det jo egentlig er) på GUIDOmap'et.

Definitionen på en GUIDOmapComponent skal altså udvides så den kan indeholde information om hvorvidt det er et billede eller et SVG ikon, der skal vises som den grafiske repræsentation på GUIDOmap'et.

7.7 XML struktur for GUIDOmap data

Det nemmeste format at indhente og udtrække information fra set med SVG øjne er et XML dokument. Agenten indeholder allerede indbyggede funktioner til at generere XML data, så det er en nem måde at generere data, der skal returneres til klienten på.

For alle komponenter (GUIDOmapComponents eller GUIDOmapLinks) benyttet på GUIDOmap'et skal et XML element der afspejler en komponent inkludere følgende data:

- `id` - *Unikt ID for komponenten*
- `category` - *Kategori for elementet*
- `alarmStatus` - *Alarmstatus (severity) for den pågældende komponent*

Derudover skal et GUIDOmapLink også indeholde `lineColor`, der skal indeholde den originale farve på GUIDOmapLinket.

Man skal kunne konfigurere agenten til hvilke af komponenternes attributer der yderligere skal sendes med for det enkelte GUIDOmap, så det er muligt at lave kunde specifikke løsninger.

Der skal være en funktion der kan parse XML data filen og opdatere de forskellige komponenter på GUIDOmap'et med de nye data. Ved opdatering af alarmstatus skal denne funktion yderligere kunne starte alarmanimationen på den specifikke komponent.

Følgende viser et eksempel på returneret XML standard data for et GUIDOmap med tre GUIDOmapLinks og fire GUIDOmapComponents.

```
1 <polldata>
2   <GUIDOelement id = "Link Cop-to-Lon" category = "GUIDOmapLink"
        alarmStatus = "0" lineColor = "#000000" />
3   <GUIDOelement id = "Link Cop-to-Mun" category = "GUIDOmapLink"
        alarmStatus = "0" lineColor = "#000000" />
4   <GUIDOelement id = "Link Lon-to-Par" category = "GUIDOmapLink"
        alarmStatus = "0" lineColor = "#000000" />
5   <GUIDOelement id = "Site Copenhagen" category = "GUIDOmapComponent"
        alarmStatus = "3" />
6   <GUIDOelement id = "Site London" category = "GUIDOmapComponent"
        alarmStatus = "5" />
7   <GUIDOelement id = "Site Munich" category = "GUIDOmapComponent"
        alarmStatus = "0" />
8   <GUIDOelement id = "Site Paris" category = "GUIDOmapComponent"
        alarmStatus = "3" />
9 </polldata>
```

Kapitel 8

Implementering

Det følgende afsnit beskriver implementationen af en prototype på en webbaseret alarmmanager. Prototypen er baseret på eksemplerne fra analyse delen og er grundlæggende en samling af de forskellige eksempler. Designet definere tre dele, men implementationen gælder kun det grafiske præsentations vindue (i kravspecifikationen kaldet *MapViewer*).

Når hoveddokumentet er indlæst kaldes funktionen `init(evt)`. Denne bestemmer hvilken SVG viewer der benyttes og sætter den globale variabel `g_userAgent` lig med den fundne SVG Viewer. Dette er nødvendigt for at kunne tilpasse de forskellige funktioner til den enkelte browser/SVG Viewers implementering af en script engine. Fx understøtter Firefox 1.5 ikke brug af animation og man har derfor brug for at kunne oprette en betinget sætning, der ud fra hvilken SVG Viewer der benyttes, enten kan starte animationen, eller springe den over.

8.1 Sikkerhed

I første omgang er der fokuseret på at kunne vise den grafiske repræsentation af et GUIDOmap hierarki, så sikkerheden er foreløbigt gemt til en senere version. Agenten er dog konfigureret til eksplicit log-ind, således at man får en standard log-ind dialog boks når man forsøger at åbne hovedsiden. Dvs. at man ikke kan se hovedsiden, med mindre man indtaster gyldige log ind oplysninger.

Brug af SSL og mulighed for konfiguration af dette er gemt til den fulde version, der indeholder den hierarkiske visning, samt alarmlog'en.

8.2 Indlæsning af nyt GUIDOmap

Den grundlæggende funktion i prototypen er at kunne indlæse et GUIDOmap fra agenten og præsentere det grafisk. UHC har deres eget script sprog kaldet LMIF. Vha. LMIF kan man lave server scripts der fx kan udtrække data fra agenten. En side der indeholder LMIF scripts skal have endelsen `.lhp`. Det første problem der opstod ved at generere en side med SVG kode i agenten og returnere den til en browser, var at browserne ikke genkendte endelsen `.lhp` og slet ikke som SVG! Derfor er der lavet et script i agenten, der gør at man kan bede om en SVG fil (altså en fil med endelsen `.svg`) og få returneret en `.lhp` side med MIME typen sat til `image/svg+xml`. Dvs. at agenten kan indeholde en fil med navnet `main.lhp` der genererer et SVG dokument. Hvis man i en browser indtaster adressen og filnavnet `main.svg`, vil agenten først lede efter en fil der hedder `main.svg`. Hvis den ikke findes, leder den efter en fil med navnet `main.lhp`. Finder den nu en fil, vil al LMIF script kode blive udført som var det en almindelig `.lhp` fil og herefter vil den genererede fil blive returneret med MIME typen sat

til `image/svg+xml` og den originale endelse `.svg`. På den måde behandler browserne automatisk siden som et SVG dokument, selvom agenten egentlig behandler det som en lhp-side.

Eftersom al information om hvordan et GUIDOmap skal præsenteres ligger i agenten, er det også vha. LMIF scripts et GUIDOmap genereres som et SVG dokument. Vha. JavaScript styres indlæsningen af data fra agenten og hvordan dataen skal tilføjes til dokumentet. Grundelementet i et SVG dokument er `<svg>` elementet. Alle browserne understøtter brug af `<svg>` elementer i `<svg>` elementer. På et `<svg>` element kan man sætte attributten `viewBox`. Vha. `viewBox` attributten kan man definere i pixels hvor meget ens SVG dokument skal kunne vise. Denne størrelse kan sagtens afvige fra den fysiske størrelse af ens SVG dokument i pixels. Hvis man fx har en browser der er 1024 pixels bred og man ønsker at vise et billede der er 2400 pixels bredt, kan man sætte `viewBox` elementet til "0 0 2400 1200". Forudsat at billedet vises fra punkt (0,0) vil hele billedets brede nu blive vist på SVG dokumentet.

GUIDOmap'sne der er brugt sammen med denne prototype indeholder alle et baggrundsbillede af forskellig størrelse. For at et GUIDOmap kan vises i naturlig størrelse er man nødt til at vise GUIDOmap'et i en SVG container i selve hoved SVG dokumentet. Denne SVG container har ID'et `mapShow`. For hver gang der indlæses et nyt GUIDOmap, sættes `viewBox` elementet til størrelsen af baggrundsbilledet således at hele GUIDOmap'et altid vises. Dette gør også at man kan placere fx boksen med knapper til visning af de forskellige lag uden for GUIDOmap'et i selve hoveddokumentet.

Hovedfilen `exam_main.lhp` indeholder en JavaScript funktion, der starter indlæsningen af et GUIDOmap. Variablen `g_firstMap` indeholder en base64 kodning af de meget lange unikke id's som man benytter for at kunne identificere et GUIDOmap i agenten. Parameteren `map` i funktionen `loadMap(evt, map)` er netop sådan et ID. Funktionen laver en URL streng og sender ID'et med i adresse strengen. Herefter kaldes funktionen `requestData(url)` med URL'en som parameter. Denne funktion opretter et `XMLHttpRequest` objekt og indlæser den angivne URL.
Linien `xmlHttp.onreadystatechange = stateChanged;` gør at funktionen `stateChanged()` kaldes når objektet ændrer tilstand.

Filen `exam_getMap.lhp` genererer i første omgang SVG grundstrukturen for GUIDOmap'et, d-vs. SVG containeren. Samtidig sættes størrelsen på SVG containeren og `viewBox` attributten og `preserveAspectRatio` attributten, der angiver hvordan indholdet i SVG containeren (billeder, m.m.) skal skaleres, angives.

`XMLHttpRequest` objektet indeholder en række flag, der sættes når fx al data er hentet fra agenten. Linien `if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")` sikrer at data behandling af svaret fra agenten først påbegyndes når hele filen er hentet. Svaret fra agenten hentes ud af `XMLHttpRequest` objektet som en tekst streng og parses med funktionen `str2SVG` til XML data der kan tilføjes til SVG containeren.

Af ukendte årsager blev `<image>` elementer indlæst vha. `XMLHttpRequest` objektet og parset til XML data, ikke vist i Firefox når disse blev tilføjet til dokumentet. Vha. Mozillas eget hjælpe værktøj *DOMinspector* kunne man se at data blev tilføjet rigtigt nok, men det virker som om Firefox ikke opdatere `<image>` elementerne og sørger for at det billede de peger på bliver vist. Derfor skal indlæst data igennem en hjælpe funktion, `updateImagesInFirefox(str)` der ud fra SVG og XLINK namespacene opretter et `<image>` direkte ud fra DOM metoden `createElementNS()` og erstatter de originale `<image>` elementer med disse i stedet. Herefter har Firefox ingen problemer med at vise billederne.

Herefter indsættes den indlæste SVG container (med ID sat til `mapHidden`) i hoveddokumentets DOM struktur. En reference til den nye SVG container gemmes i variablen `g_appendPos`. Derefter skal "lag 0" hentes. Som defineret i kravspecifikationen skal man kunne præsentere GUIDOmaps i lag. Samtidig

indeholder alle komponenter en parameter således at man kan angive hvilket lag den enkelte komponent skal tilhøre. Denne værdi skal angives til en postiv værdi større end nul, da lag nul bruges til at indlæse baggrundsbilledet (hvis der er defineret et), samt hvor mange lag det pågældende GUIDOmap indeholder. For at hente lag nul, eller baggrundslaget om man vil, benyttes funktionen `loadLayer(map, isVisible)` der på samme måde som ved indlæsningen af SVG containeren kalder den samme fil. Forskellen er blot at lag nummeret nu sendes med (i dette tilfælde 0 for at indlæse baggrunden). Filen tester på om der er et lag nummer med eller ej, og på baggrund af det genereres enten SVG grundstrukturen, eller det angivne lag på GUIDOmap'et. Når lag nul er hentet og indsat i den indlæste SVG container, startes animationen af overblænding i mellem de to GUIDOmaps (det der allerede vises og det nye). Dette er grunden til at den nye SVG containeren indsættes bag den originale SVG container (`mapShow`). Når animationen slutter og det er indholdet af den nye SVG container der vises, mens den gamle skjules (overblændings animationen sætter gennemsigtigheden for GUIDOmap'et fra 1 til 0). Samtidig køres funktionen `renderNewMap(map)`. Denne funktion sørger for at slette det gamle GUIDOmap og ændre ID'et på den nye SVG container til `mapShow`. Har man valgt ikke at benytte animationer (eller understøtter browseren ikke animation), køres denne naturligvis ikke og funktionen `renderNewMap(map)` køres med det samme.

Lige så snart at baggrundslaget er indlæst, begynder indlæsningen af de resterende lag. Vha. referencen i variablen `g_appendPos` tilføjes de til SVG containeren. Selv om animationen ikke er færdig, kan indlæsningen af de resterende lag altså sagtens fortsætte.

Når alle lag er indlæst, køres endnu en lille renderings hjælpefunktion `setWidthFromTxtLength(tag)`. Alle GUIDOmapComponents og GUIDOmapLinks kan have en label. Bredden af denne label kan man ikke regne ud på serveren, da tekst ikke vises helt ens i de forskellige browsere. Dvs. man altså først på klient siden kan beregne præcis hvor mange pixels teksten på labellen fylder og herefter tilpasse bredden til teksten.

Som det sidste i indlæsningen af et nyt GUIDOmap, startes poll funktionen.

8.3 Poll GUIDOmap

Den grundlæggende indlæsning af poll data er taget fra eksemplet i analyse afsnittet og data strukturen der hentes er beskrevet i designafsnittet. Når denne XML data er hentet fra agenten, skal de forskellige GUIDOmapComponents og GUIDOmapLinks opdateres. Dette gøres ved hjælp funktionen `pollMapDataLoaded()`. Den løber alle elementerne i XML filen igennem og opdaterer alarmstatus på de forskellige komponenter. Alle komponenter på GUIDOmap'et indeholder deres nuværende alarmstatus (severity). Hvis den nuværende værdi er forskellig fra den nye værdi, startes "alarm animationen" for den pågældende komponent. For GUIDOmapComponents skiftes farven på den omkransende firkant til den pågældende alarmfarve og firkanten blinker fem gange. For GUIDOmapLinks ændres grundfarven til alarmfarven og hele GUIDOmapLinket blinker herefter fem gange.

Endvidere er der implementeret en status indikator for om pollningen er startet eller stoppet. Klikker man på denne indikator skifter man også status for pollningen, eller sagt med andre ord man kan tænde eller slukke for pollningen.

8.4 Skjul/vis lag på GUIDOmap

Når indlæsningen af et GUIDOmap påbegyndes, startes indlæsningen af kontrollerne til at skjule og vise lag på GUIDOmap'et også. Indlæsning og parsning af data foregår på samme måde som ved indlæsning af et GUIDOmap, altså vha. et `XMLHttpRequest` objekt og funktionen `str2SVG(str)`.

Filen `getLayerControl.lhp` genererer selve kontrollen ud fra GUIDOmap ID'et der sendes med i URL'en. Et LMIF script undersøger hvor mange lag det pågældende GUIDOmap indeholder og tilpasser antallet af knapper til antallet af lag.

Herefter kan man ved at klikke på knappen ud for det enkelte lag nummer, skjule eller vise det pågældende lag. Når man klikker på en af knapperne, kaldes funktionen `layer_switch(evt,f)`, hvor `f` er lag nummeret. SVG specifikationen indeholder et gruppe element, `<g>`, som man kan gruppere elementer med. Hvert lag indlæses som en gruppe med et id sammensat af det unikke GUIDOmap id, teksten "layer_" og lag nummeret. Synligheden af et lag kan så sættes på det gruppe element der omkranser alle elementer i det pågældende lag, ved at ændre attributten `visibility` til hhv. `visible` eller `hidden`. Alle elementer i gruppe elementet nedarver synligheden, så alle elementer i gruppen skjules eller vises samtidig.

8.5 Animeret overgang i mellem GUIDOmaps

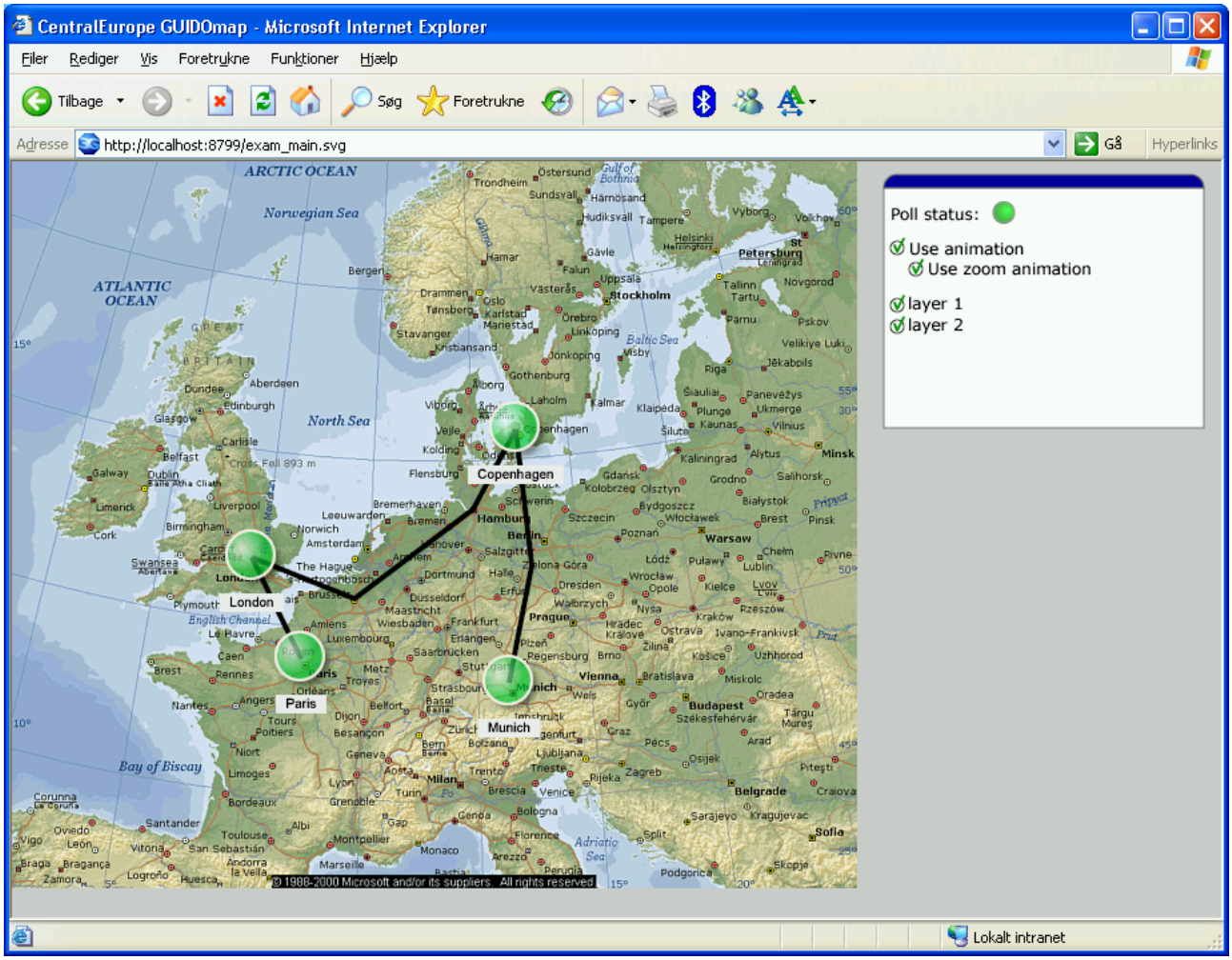
På grundstrukturen af den SVG container der indeholder alle elementer for det enkelte GUIDOmap, er der defineret tre SVG animationer.

Den første animerer `viewBox` attributten fra at vise hele billedet, til kun at vise et lille udsnit omkring der hvor brugeren klikkede med musen. Dette vil virke som om man zoomer ind på der hvor man klikkede (dvs. den valgte GUIDOmapComponent, eller det valgte GUIDOmapLink). En SVG animation kan startes på to måder. Enten ved at attributten `begin` er sat til noget deklarativt, dvs. at man benytter sig af SVG kode til at starte animationen, fx ved at sætte attributten til `mouseover` eller `click`, eller ved hjælp af javascript. For at kunne kalde den indbyggede funktion `beginElement()` på animationselementet for at starte animationen via JavaScript, skal attributten `begin` være sat til `indefinite`. Varigheden for animationen er her sat til ét sekund.

Animation nummer to laver overblændings effekten. Den animere gennemsigtigheden af hele GUIDOmap'et fra ingen gennmsigtighed (værdien 1) til transparent (værdien 0). Attributten `fill` sættes til `freeze` så resultatet af animationen bliver bibeholdt når animationen er færdig. Det vil i dette tilfælde sige at GUIDOmap'et forbliver transparent når animationen er slut.

Den sidste animation opnås ved brug af `<set>` elementet. Den sætter attributten `visibility` på GUIDOmap'et til `hidden` for at være helt sikker på at hele GUIDOmap'et skjules.

Alle animationerne startes fra funktionen `animateChange()`. Den kalder den indbyggede funktion `beginElement()` på alle animations elementerne, hvilket starter de enkelte animationer.



Figur 8.1: Den implementerede prototype

Kapitel 9

Test

Testscenariet omfatter lokal afvikling af den implementerede prototype på en lokal Q3ADE agent. Testplanerne er udfærdiget for at teste alle implementerede funktioner i prototypen.

9.1 Testplan

Oversigt over udførte test cases:

9.1.1 Test case: Indlæs GUIDOmap

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	22-9-2006	KD	1	1.0

Test-formål:

At undersøge om de i kravspecifikationen anførte krav til indlæsning af et GUIDOmap i en browser med SVG understøttelse overholdes

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 2 GUIDOmaps. Disse kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Det indlæste GUIDOmap vises

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.1, 3.2, 3.5, 3.6

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
GUIDOmap'et indlæses og vises. Al ekstern raster grafik indlæses også og vises.
Opera 9
GUIDOmap'et indlæses og vises. Al ekstern raster grafik indlæses også og vises.
Firefox 1.5
GUIDOmap'et indlæses og vises. Al ekstern raster grafik indlæses også og vises. Enkelte gange vises tekst elementer ikke ordentligt. Der kan fx mangle halvdelen af det sidste bogstav.

Test-konklusion:

Alle tre browsere kan indlæse og vise et GUIDOmap. De en gang i mellem manglende bogstaver må tillægges firefox redering af SVG dokumenter.

9.1.2 Test case: Vis klikbarhed på ikon

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	26-9-2006	KD	2	1.0

Test-formål:

At undersøge om klikbarheden af et ikon vises ved brugen af ikoner med specificeret effekt når musen holdes over ikonet på et GUIDOmap

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 1 GUIDOmap med tilhørende MapComponent med attributen `roll-over icon` defineret. Dette kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Standard GUIDOmap'et vises
4. Flyt musen skiftevis henover et / alle ikoner (MapComponents med `roll-over` ikon attributen sat), skiftevis væk fra ikonet igen
5. Ikonet musen holdes henover skal skifte mellem det under `GUIDOview.rollOverImage.normal` og det under `GUIDOview.rollOverImage.rollOver` angivne billede

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.3, 6.4, 3.2

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Den viste MapComponent skifter mellem de to angivne billeder når musen køres henover
Opera 9
Den viste MapComponent skifter mellem de to angivne billeder når musen køres henover
Firefox 1.5
Den viste MapComponent skifter ikke mellem de to angivne billeder når musen køres henover

Test-konklusion:

Internet Explorer/Adobe og Opera kunne udføre testen fejlfrit. Firefox 1.5 understøtter ikke animation og brug af `<set>` elementet og kan derfor ikke vise effekten

9.1.3 Test case: Skjul/vis lag på GUIDOmap

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	3	1.0

Test-formål:

At undersøge om de på et GUIDOmap indlæste lag kan vises/skjules ved hjælp af brugerkontrollerne

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 1 GUIDOmap. Dette GUIDOmap indeholder som minimum et lag. Dette kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Standard GUIDOmap'et vises
4. Klik på kontrollen (det grønne V) ud for et lag
5. Komponenterne tilhørende laget skjules nu
6. Klik igen på kontrollen (den hvide cirkel) ud for det samme lag
7. Komponenterne tilhørende laget vises nu igen

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.1, 3.2, 3.5, 3.6, 3.7

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3
Ved klik på kontrolknappen vises/skjules laget skiftevis
Opera 9
Ved klik på kontrolknappen vises/skjules laget skiftevis
Firefox 1.5
Ved klik på kontrolknappen vises/skjules laget skiftevis

Test-konklusion:

Alle tre browsere kan vise og skjule et lag vha. kontrollen.
NB: Ved første klik på kontrollen sker der ikke noget! Alle browserne er pga. et amerikansk patent vedr. automatisk aktivering af indlejrede objekter nødsaget til at man skal aktivere disse elementer vha. et klik.

9.1.4 Test case: Poll GUIDOmap

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	4	1.0

Test-formål:

At undersøge om et indlæst GUIDOmap kan indhente information fra Q3ADE agenten vha. pollning og præsentere den information på GUIDOmap'et

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Trace sætninger level 7 er slået til på Q3ADE agenten (kommando: `systest e 7`)
3. Q3ADE agenten indeholder minimum 1 GUIDOmap. Dette kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
4. Der er etableret netværksforbindelse til Q3ADE agenten
5. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Det indlæste GUIDOmap vises
4. Poll status indikatoren skifter til grøn
5. Q3ADE agent viser meddelelsen: "Map polled and data returned" for hver gang GUIDOmap'et polles

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Følgende krav fra kravspecifikationen er en del af denne test: 3.9

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Poll status indikatoren skifter til grøn og viser at pollningen er startet. Q3ADE agenten udskriver succes meddelelsen
Opera 9
Poll status indikatoren skifter til grøn og viser at pollningen er startet. Q3ADE agenten udskriver succes meddelelsen
Firefox 1.5
Poll status indikatoren skifter til grøn og viser at pollningen er startet. Q3ADE agenten udskriver succes meddelelsen

Test-konklusion:

Alle tre browsere kan udføre pollning af et GUIDOmap

9.1.5 Test case: Opdater alarmstatus

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	5	1.0

Test-formål:

At undersøge om MapComponents og MapLinks får opdateret alarmstatus

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 1 GUIDOmap. Dette kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Q3Browser er installeret
5. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl
6. Test 4: Poll GUIDOmap skal være gennemført uden fejl

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Det valgte standard GUIDOmap'et vises
4. Poll status indikatoren skifter til grøn
5. Start programmet Q3Browser
6. Fold `GUIDOgui` ud i tree-viewet
7. Fold det før indlæste GUIDOmap ud i tree-viewet
8. Vælg en GUIDOMapComponent
9. Ændrer alarmtilstanden under feltet Alarm til `activeReportable-Critical`
10. Alarmen vises nu i browseren på den valgte MapComponent (rød (=critical) firkant der blinker)

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Testen udføres med valg af både GUIDOmapComponent og GUIDOmapLink under test beskrivelsens punkt 8

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3(.2).1, 3(.2).2, 3.4, 3.9, 3(.9).2

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Alarmtilstanden opdateres for den valgte MapComponent/det valgte MapLink
Opera 9
Alarmtilstanden opdateres for den valgte MapComponent/det valgte MapLink
Firefox 1.5
Alarmtilstanden opdateres for den valgte MapComponent/det valgte MapLink

Test-konklusion:

Alle tre browsere kan opdatere alarmtilstanden for en MapComponent eller et MapLink

9.1.6 Test case: Drill-down mellem GUIDOmaps

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	6	1.0

Test-formål:

At undersøge skift i mellem to GUIDOmaps uden animation

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 2 GUIDOmaps. Disse kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Det ene GUIDOmap skal indeholde et ikon der peger på det andet GUIDOmap (`linkTo` attributen skal være sat)
5. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl
6. Markeringen i feltet `Use animation` er slået fra i prototypen

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Standard GUIDOmap'et vises
4. Klik på et ikon
5. Der skiftes til det næste GUIDOmap

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.1, 3.2

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Der skiftes i mellem de to GUIDOmaps
Opera 9
Der skiftes i mellem de to GUIDOmaps
Firefox 1.5
Der skiftes i mellem de to GUIDOmaps

Test-konklusion:

Alle tre browsere skifter til og viser det nye GUIDOmap

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	7	1.0

Test-formål:

At undersøge om standard animeringen (overblænding) ved skift i mellem to GUIDOmaps vises

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 2 GUIDOmaps. Disse kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Det ene GUIDOmap skal indeholde et ikon der peger på det andet GUIDOmap (`linkTo` attributen skal være sat)
5. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl
6. Markeringen i feltet `Use animation` er slået til i prototypen
7. Markeringen i feltet `Use zoom animation` er slået fra i prototypen

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Standard GUIDOmap'et vises
4. Klik på et ikon
5. Animationen (overblænding) vises, mens der skiftes til det næste GUIDOmap

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.1, 3.2

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Animationen vises og der skiftes i mellem de to GUIDOmaps
Opera 9
Animationen vises og der skiftes i mellem de to GUIDOmaps. Enkelte gange vises det "nye" GUIDOmaps baggrundsbillede først når animationen er færdig
Firefox 1.5
Animationen vises ikke men der skiftes i mellem de to GUIDOmaps

Test-konklusion:

Internet Explorer/Adobe fuldførte testen fejlfrit. Opera havde en tendens til ikke at nå at rendere baggrundsbilledet for det nye GUIDOmap før animationen sluttede. Dette tilegnes Operas egen rendering af SVG `<image>` elementer i et SVG dokument. Firefox 1.5 understøtter ikke brug af animation, hvorfor den ikke vises her. Alle tre browsere viser dog det nye GUIDOmap

Test case

Testtype:	Dato:	Udført af:	Testnr.:	Version:
Accept test	27-9-2006	KD	8	1.0

Test-formål:

At undersøge om animeret skift (zoom + overblænding) i mellem to GUIDOmaps vises

Test-forudsætninger:

1. Q3ADE agenten er startet
2. Q3ADE agenten indeholder minimum 2 GUIDOmaps. Disse kan genereres vha. LMIF scriptet $/Projects/KD Examproject/sw/exam_generate_GUIDOmaps.lmi
3. Der er etableret netværksforbindelse til Q3ADE agenten
4. Det ene GUIDOmap skal indeholde et ikon der peger på det andet GUIDOmap (`linkTo` attributen skal være sat)
5. Test 1: Indlæs GUIDOmap skal være gennemført uden fejl
6. Markeringen i feltet `Use animation` er slået til i prototypen
7. Markeringen i feltet `Use zoom animation` er slået til i prototypen

Testbeskrivelse:

1. Start en browser der understøtter visning af SVG billeder
2. Indlæs prototypen på den webbaserede alarmmanager ved at indtaste adressen på hovedsiden: `exam_main.svg` (Q3ADE agent kan lokalt nås vha. `http://localhost:8799/exam_main.svg`)
3. Standard GUIDOmap'et vises
4. Klik på et ikon
5. Animationen (zoom + overblænding) vises, mens der skiftes til det næste GUIDOmap

Testen udføres ved brug af browserne Firefox 1.5, Opera 9 og Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in i hht. krav 6.4 i kravspecifikationen

Specifikke test-cases:

Referencer til krav under test:

Testen omfatter følgende punkter i kravspecifikationen: 3.1, 3.2

Tilladte afvigelser / fejl:

Ingen

Korrigerende handlinger ved fundne fejl:

Fejlen rettes og testen udføres igen

Test-resultater:

Internet Explorer 6 med Adobe SVG Viewer 3.03 plug-in
Animationen vises og der skiftes i mellem de to GUIDOmaps
Opera 9
Animationen vises og der skiftes i mellem de to GUIDOmaps. Enkelte gange vises det "nye" GUIDOmaps baggrundsbillede først når animationen er færdig
Firefox 1.5
Animationen vises ikke men der skiftes i mellem de to GUIDOmaps

Test-konklusion:

Internet Explorer/Adobe fuldførte testen fejlfrit. Opera havde en tendens til ikke at nå at rendere baggrundsbilledet for det nye GUIDOmap før animationen sluttede. Dette tilegnes Operas egen rendering af SVG `<image>` elementer i et SVG dokument. Firefox 1.5 understøtter ikke brug af animation, hvorfor den ikke vises her

9.2 Performance test

Følgende er en performance test af ressource forbruget målt på CPU belastning ved pollning og ved de animerede overgange ved skift i mellem to GUIDOmaps.

	Måling										
	1	2	3	4	5	6	7	8	9	10	gnm.
IE	1%	2%	11%	7%	7%	7%	7%	7%	15%	7%	7.1%
Opera	4%	7%	8%	4%	4%	4%	4%	5%	10%	4%	5.4%
Firefox	5%	5%	4%	6%	5%	6%	5%	5%	5%	7%	5.3%

Tabel 9.1: Performance data - CPU belastning ved pollning

	Fra GUIDOmap 1 til 2				Fra GUIDOmap 2 til 3			
	Test 1	Test 2	Test 3	gnm.	Test 1	Test 2	Test 3	gnm.
IE	47%	36%	58%	47%	49%	54%	46%	49.7%
Opera	54%	53%	56%	54.3%	60%	62%	53%	58.3%
Firefox	25%	29%	29%	27.7%	43%	56%	32%	43.7%

Tabel 9.2: Performance data - CPU belastning ved zoom + overblændingsanimation

	Fra GUIDOmap 1 til 2				Fra GUIDOmap 2 til 3			
	Test 1	Test 2	Test 3	gnm.	Test 1	Test 2	Test 3	gnm.
IE	51%	50%	40%	47.0%	58%	47%	54%	53.0%
Opera	58%	54%	56%	56.0%	53%	55%	55%	54.3%
Firefox	25%	25%	23%	24.3%	49%	42%	36%	42.3%

Tabel 9.3: Performance data - CPU belastning målt ved overblændingsanimation

Testen er foretaget ved brug af Windows indbyggede CPU måler. Test maskinen indeholder en Intel Pentium 4 3GHz processor med Hyper Threading og 1GB RAM, altså en fortrinsvis kraftig maskine.

Kigger man på den gennemsnitlige CPU belastning ved pollning, bruger Internet Explorer / Adobe SVG Viewer en lille smule mere CPU kraft på at udføre en enkelt pollning. I alle tilfælde med alle tre browsere, returnere CPU belastningen til 0 i mellem pollningerne, så timer funktionen bruger altså ikke noget på at stå og vente (i hvert fald ikke noget synligt). Men uanset om det er ca. 5 eller 7 % CPU belastning, er det på ingen måde en skræmmende mængde regnekraft der skal benyttes. Ønsker man derimod noget højere poll frekvens, fx to gange i sekundet, kan det begynde at blive en betydelig belastning, hvis computeren skal udføre andre opgaver end at stå og udføre pollning.

De to andre test indeholder to væsentlige pointer der er værd at nævne. Først så er det værd at bemærke at der ikke er den store forskel på om man kun benytter den ene slags animation, eller om man benytter begge. Afvigelserne på gennemsnittet af målingerne er kun nogle få procent point. For det andet er det værd at bemærke Firefox browserens CPU belastning. Firefox understøtter ikke brug af animation, så her er værdierne udelukkende for indlæsning af et GUIDOmap. Ved skift fra GUIDOmap 1 til 2 er CPU belastningen naturligt meget mindre for Firefox end de to andre browsere, men ved skift fra GUIDOmap 2 til 3 stiger belastningen ved brug af Firefox med næsten 20 %. En del af forklaringen på dette hænger sammen med den mængde data der skal tilføjes og altså også den mængde data der

skal hentes fra agenten. GUIDOmap 2 indeholder kun en komponent, mens GUIDOmap 3 indeholder 10 komponenter.

Til sidst skal det bemærkes at skift i mellem to GUIDOmaps er en handling udført af en alarmoperatør, og altså ikke noget der skal udføres flere gange i sekundet.

Kapitel 10

Værktøjer

Til web programmering er programmet (x)HTML Kit benyttet. Det har en god farve kodning og der findes mange plug-ins til det. Mest har jeg benyttet et plug-in til der kan indsætte SVG elementer (fx et animations element, `<animate>`). Samtidig er dette plug-in en god reference da man kan vælge at indsætte en attribut med alle de værdier den pågældende attribut kan sættes til. Herefter kan man så vælge den man ønsker og slette resten.

Det mest uundværlige værktøj er uden tvivl Mozillas DOM Inspector. Her kan man få et tree view over DOM strukturen for den indlæste dokument. I forbindelse med at skulle indlæse SVG data fra agenten og tilføje dette til hoveddokumentet, har dette værktøj været en stor hjælp når man skal finde ud af om data bliver indsat det rigtige sted. Samtidig kan man vælge at se et knudepunkt i dokumentets DOM struktur som et JavaScript objekt. Det betyder at man kan se alle attributter og indbyggede funktioner knyttet til det pågældende objekt. Dette kan også være en stor hjælp i debug processen.

Kapitel 11

Fremtidige udvidelser

Web applikationen er udviklet til at præsentere en alarmoperatør for alarmer der optræder i systemet. Det må derfor aldrig være muligt at skjule en komponent der har alarmstatus (severity) af andet end "cleared". Prototypen skal udvides således at en komponent der ændrer status fra "cleared" automatisk bliver synlig på GUIDOmap'et uanset om alarmoperatøren har valgt at skjule den pågældende komponent eller ej.

Sikkerheden er en af de vigtigste ting i moderne software udvikling, hvorfor det også må ses som en ting der skal implementeres i en endelig udgave. Brugeren skal have mulighed for at angive i et konfigurationspanel at al netværkstrafik skal foregår ved brug af SSL.

Eventkanalen er agentens hurtigste vej til at vise den nuværende alarmsituation. Derfor er denne også en udvidelse der skal være med i en endelig version.

Ikon bibliotek skal implementeres så en bruger har mulighed for at benytte SVG ikoner. Det vil kræve rettelser både i prototypen, men også som tidligere nævnt i agenten. Agenten skal udvides således at man på en GUIDOmapComponent kan specificere at ikonet er af typen SVG.

Ud fra designet og kravspecifikationen, skal der også implementeres en hierarkisk visning af de GUIDOmaps agenten indeholder. Endvidere er der også krav om en alarmlog. Alarmlog'en vil være den der primært skal modtage alarm events fra agenten. Hvis alle GUIDOmaps også skulle modtage events, ville netværks belastningen blive forhøjet unødvendigt.

Kapitel 12

Konklusion

Det overordnede mål med projektet var at foretage en mængde research vedr. mulighederne for at udvikle en webbaseret alarmmanager frem for UH Communications (UHC) nuværende Java klient. Der er i projektet foretaget en omfangsrig og grundig analyse af anvendeligheden af en række teknologier udvalgt i samarbejde med UHC til netop dette. Analyse afnittet afslører at standardisering er tidskrævende process. På trods af at det er mere end tre år siden at specifikationen for SVG officielt fik status som standard, må understøttelsen stadig i et vist omfang ses som sparsomt. Dette skyldes primært at Microsoft som den største udbyder af webbrowsere, ikke understøtter direkte afvikling af SVG. Samtidig har Adobe beklageligvis i slutningsfasen af dette projekt offentliggjort at de stopper udviklingen af deres SVG Viewer. Begrundelsen bunder i at flere og flere browsere understøtter direkte afvikling af SVG og det derfor overflødiggør deres produkt. Dette vil jeg sige i høj grad afhænger af Microsofts vilje til at understøtte SVG. Microsoft har i enkelte af deres almindelige programmer, fx Visio, mulighed for at gemme et diagram som et SVG dokument, så SVG formatet er ikke fremmed for Microsoft. Størstedelen af UHCs kunder benytter Windows platformen og dermed også Internet Explorer, så fremtiden for en webbaseret alarmmanager der bygger på SVG afhænger i høj grad af hvad Microsoft vælger at satse på. Specielt når udviklingen af den mest udbredte SVG Viewer stopper.

I forbindelse med indtastning af data, blev XForms udvalgt som teknologi. Baseret på teori og analyse afsnittene vedrørende XForms, må konklusionen være at XForms er i sin spæde begyndelse. Specifikationen for XForms har også kun været officielt frigivet som standard af W3C siden starten af året.

Ingen af browserne havde direkte understøttelse af XForms og kun Mozilla havde en beta udgave at et plug-in, der derigennem kunne tilføje understøttelse af XForms. Her skal UHCs kundeandel der benytter Internet Explorer igen sammenholdes med disse muligheder. XForms indeholder potentialet til at man kan udvikle indtastningsmuligheder der er integrerbare med SVG, men den manglende understøttelse i browserne gør konklusionen klar, nemlig at der skal større udbredelse til, før det er en teknologi der skal satses på.

Det overordnede mål vedr. de udvalgte teknologier er altså nået. Der er redegjort for de forskellige teknologiers nuværende stadie, samt deres fremtidsperspektiver.

Som en sekundær del af projektet skulle mulighederne med de udvalgte teknologier afprøves vha. nogle implementerede eksempler. Disse eksempler er blevet udviklet og samlet til en mindre prototype, der dog i forhold til den fulde kravspecifikation og det fulde design kun indeholder en begrænset funktionalitet.

Prototypen er i stand til at udtrække information om et GUIDOmap fra Q3ADE agenten og præsentere dette i en webbrowser. Kravet var at denne grafiske præsentation skulle kunne ske i browserne Internet Explorer, Opera og Firefox. Dette krav er opfyldt og man kan indlæse et GUIDOmap i alle

de tre udvalgte browsere. Dermed er første skridt i retning af at tilgode se alle UHCs kunder, uanset hvilken platform de har valgt at køre med, taget. En fuld test på de enkelte platforme skal dog gennemføres. Selvom det i princippet er browser udviklernes opgave at sørge for at fx script enginen er ens i den enkelte browser uanset hvilken platform browseren afvikles på (forudsat selvfølgelig at browseren understøtter platformen!), er det kun en fuld test der kan afsløre om virkeligheden forholder sig sådan.

Som en forbedring af brugergrænsefladen for alarmoperatøren i forhold til den nuværende Java version, blev der stillet som krav, at man skulle indlæse et GUIDOmap i flere lag, for at gøre en form for gruppering af elementer på et GUIDOmap mulig. Dette krav er også opfyldt. Man kan specificere et lag nummer for en GUIDOmapComponent eller et GUIDOmapLink og via kontrollerne "tænde" og "slukke" for de enkelte lag. Dette vil være en stor hjælp til alarmoperatøren, hvis et oversigtskort fx indeholder mange komponenter og man ønsker at sortere eller minimere antallet af viste komponenter på skærmen. Kort sagt en alarmoperatør har nu mulighed for at tilpasse visningen efter ønske. I den forbindelse blev det også stillet som krav at en alarm aldrig må kunne skjules. Dvs. at hvis en alarm optræder i systemet på en komponent som alarmoperatøren har valgt at skjule, skal denne blive synlig og vise alarmtilstanden. Dette krav er ikke opfyldt. Hvis man skjuler et lag i prototypen og dette lag indeholder en komponent der herefter ændrer alarmstatus, vil denne komponent forblive skjult.

Der blev stillet krav om at overgangen i mellem to GUIDOmaps skulle forbedres rent grafisk. Dette krav er opfyldt. I prototypen sker denne overgang vha. animationer defineret i SVG. Der er lavet to forskellige animationer, en zoom effekt og en overblænding, og der er mulighed for at vælge animerede overgange helt eller delvist fra. I en endelig version af en webbaseret alarmmanager, vil det være mere hensigtsmæssigt at flytte konfigurationen af dette til et samlet konfigurationspanel. Samtidig skal operatørens valg gemmes i agenten, således at de valgte indstillinger automatisk sættes når alarmoperatøren logger på systemet.

Der blev stillet krav om at et GUIDOmap skulle kunne foretage pollning af agenten og herefter opdatere de enkelte komponenter på baggrund af den returnerede data. Dette krav er også opfyldt. Poll mekanismen aktiveres automatisk når et GUIDOmap er færdig indlæst, og opdatere alle komponenters alarmstatus. Hvis en alarmstatus har ændret sig siden sidste opdatering, startes en alarmanimation, der skal fremkalde opmærksomhed fra alarmoperatøren og aktivt vise at der er sket en ændring / fejl i systemet.

På det menneskelige plan er der ikke nogen anledning til bemærkninger. Risikostyringen har fungeret og UHC er blevet gjort opmærksom på evt. sygdom eller fravær. Der har ikke været noget data tab undervejs, så også backup proceduren har fungeret upåklageligt. Alt i alt er jeg tilfreds med projektet og glad for den erfaring og viden jeg har tilegnet mig undervejs.

Litteratur

Bøger

[1] Vladimir Geroimenko & Chaomei Chen, *Visualizing Information Using SVG and X3D*, Springer, 2005, ISBN 1852337907

[2] Craig Larman, *Applying UML and Paterns*, Prentice Hall PTR, 2002, ISBN 0-13-092569

Hjemmesider

[3] W3 Consortium - About SVG, http://www.w3.org/Graphics/SVG/About

[4] W3 Consortium SVG 1.1 specifications, http://www.w3.org/TR/SVG11/

[5] XForms 1.0 (Second Edition) - http://www.w3.org/TR/xforms/

[6] The XMLHTTPRequest Object, working draft, June 19 2006 - http://www.w3.org/TR/XMLHttpRequest/

[7] Internet Explorer 6 - Funktioner i IE 6 - http://www.microsoft.com/danmark/windows/ie/features.mspx

[8] Internet Explorer 7 - Funktioner i IE 7 - http://www.microsoft.com/danmark/windows/ie/ie7/about/features/default.mspx

[9] Opera 9 Browser - Supported W3C web standards - http://www.opera.com/docs/specs/

[10] Opera 9 Browser - Supported SVG - http://www.opera.com/docs/specs/svg/

[11] SVG understøttelse i Firefox 1.5 - http://developer.mozilla.org/en/docs/SVG_in_Firefox_1.5

[12] OpenClipart.org http://openclipart.org/cgi-bin/navigate

Underskriftsside

Projekt:	**Webbaseret alarmmanager**
Afleveringsfrist:	*Fredag den 6. oktober 2006*
DTU vejleder:	Paul Fischer, IMM
Projekt udført af:	Kjartan Døj, s022338

_____

Denne rapport er modtaget af _____ den ____ / ____ - 2006 kl: ___:___

Bilag

Bilag A

Understøttelse af SVG

Følgende er en liste over alle elementer i den fulde SVG profil, samt hvilke elementer der hører til de forskellige SVG profiler. Samtidig viser tabellen hvilke elementer de forskellige browsere / SVG viewere understøtter og hvilke undtagelser de forskellige elementer evt. måtte have.

Signatur forklaring:

- **T**: Delvist understøttet i SVG Tiny
- **B**: Delvist understøttet i SVG Basic
- **T**: Fuldt understøttet i SVG Tiny
- **B**: Fuldt understøttet i SVG Basic

Element		FireFox 1.5		Opera 9		Adobe SVG Viewer 3.0	
Structure Module							
`svg`	**TB**	✓	`currentScale` and `currentTranslate` DOM attributes are implemented, but there is no pan and zoom user interface. SVGSVGElement Unimplemented attributes:kode contentScriptType, contentStyle-Type, viewport, useCurrentView, currentView Unimplemented bindings: `pauseAnimations`, `unpauseAnimations`, `animationsPaused`, `getCurrentTime`, `setCurrentTime`, `getIntersectionList`, `getEnclosureList`, `checkIntersection`, `checkEnclosure`, `deselectAll`, `createSVGAngle`, `getElementById`	✓	Unsupported attributes: `version`, `baseProfile`	✓	Unsupported attributes: `contentScriptType`, `contentStyleType`
`g`	**TB**	✓		✓		✓	
`defs`	**TB**	✓		✓		✓	

fortsættes...

Element		FireFox 1.5		Opera 9		Adobe SVG Viewer 3.0	
`desc`	TB	✓	Only stored in the DOM, no user interface.	✓		✓	
`title`	TB	✓		✓		✓	
`metadata`	TB	✓	Only stored in the DOM, no user interface.	✓		✓	
`symbol`	- B	✓		✓		✓	
`use`	TB	✓	Only works for internal document references (bug 269482). Doesn't completely follow <svg:use> cascading rules (bug 265894). Doesn't deliver events to a SVGElementInstance tree (bug 265895).	✓		✓	Only support links to elements within the same file
Conditional Processing Module							
`switch`	TB	✓		✓		✓	
Image Module							
`image`	TB	✓	Only works for raster images (bug 272288).	✓	Image types supported: All those supported in Opera except svg	✓	Unsupported attributes: `preserveAspectRatio`. Supports GIF (unanimated), JPEG, PNG and SVG (static only) images
Style Module							
`style`	- B	✓		✓	Unsupported atrributes: `type`, `media`, `title`	✓	Unsupported atrributes: `title`
Shape Module							
`path`	TB	✓	SVGPathElement Interface Unimplemented attributes: `pathLength`, `normalizedPathSegList`, `animatedPathSegList`, `animatedNormalizedPathSegList` Unimplemented bindings: `getTotalLength`, `getPointAtLength`, `getPathSegAtLength` SVGPathSegList Interface Unimplemented bindings: `replaceItem()`	✓		✓	Unsupported attributes: `pathLength`
`rect`	TB	✓		✓		✓	
`circle`	TB	✓		✓		✓	

fortsættes...

Element		FireFox 1.5		Opera 9		Adobe SVG Viewer 3.0	
line	TB	✓		✓		✓	
ellipse	TB	✓		✓		✓	
polyline	TB	✓		✓		✓	
polygon	TB	✓		✓		✓	
Text Module							
text	TB	✓	SVGTextElement Unimplemented attributes: `rotate`, `textLength`, `lengthAdjust` Unimplemented bindings: `getNumberOfChars`, `getSubStringLength`, `getStartPositionOfChar`, `getEndPositionOfChar`, `getRotationOfChar`, `getCharNumAtPosition`, `selectSubString` Bindings not functional at onload time: `getExtentOfChar`	✓		✓	Unsupported attributes: `textLength`. `lengthAdjust`
tspan	- B	✓	SVGTSpanElement Unimplemented attributes: `rotate`, `textLength`, `lengthAdjust` Unimplemented bindings: `getNumberOfChars`, `getComputedTextLength`, `getSubStringLength`, `getStartPositionOfChar`, `getEndPositionOfChar`, `getExtentOfChar`, `getRotationOfChar`, `getCharNumAtPosition`, `selectSubString`	✓		✓	Unsupported attributes: `textLength`. `lengthAdjust`
tref	- B	-		✓		✓	Unsupported attributes: `textLength`. `lengthAdjust`
textPath	- B	✓	Implemented in Firefox 2.Unimplemented attributes: method, spacing, textLength, lengthAdjust	✓	Unsupported attributes: `spacing`	✓	Unsupported attributes: `textLength`. `lengthAdjust`, `method`, `spacing`
altGlyph	- B	-		-		✓	
altGlyphDef	- B	-		-		✓	
altGlyphItem	- B	-		-		-	
glyphRef	- B	-		-		✓	
Marker Module							
marker	- -	✓		✓		✓	
Color Profile Module							
color-profile	- B	-		-		✓	

fortsættes...

Element		FireFox 1.5	Opera 9	Adobe SVG Viewer 3.0
Gradient Module				
`linearGradient`	- B	✓	✓	✓
`radialGradient`	- B	✓	✓	✓
`stop`	- B	✓	✓	✓
Pattern Module				
`pattern`	- B	-	✓	✓ Unsupported attributes: `patternContentUnits`
Clip Module				
`clipPath`	- B	✓ Won't handle clip paths which have elements with different clip-rule properties or that reference other clipPaths. (bug 267224).	✓	✓
Mask Module				
`mask`	- B	-	✓	✓ Unsupported attribute: `maskContentUnits`
Filter Module				
`filter`	- B	-	✓	✓ Unsupported attributes: `xlink:href`
`feBlend`	- B	-	✓	✓
`feColorMatrix`	- B	-	✓	✓
`feComponentTransfer`	- B	-	✓	✓
`feComposite`	- B	-	✓	✓
`feConvolveMatrix`	- -	-	✓	✓
`feDiffuseLighting`	- -	-	✓	✓
`feDisplacementMap`	- -	-	✓	✓
`feDistantLight`	- -	-	✓	✓
`feFlood`	- B	-	✓	✓
`feFuncA`	- B	-	✓	✓
`feFuncB`	- B	-	✓	✓
`feFuncG`	- B	-	✓	✓
`feFuncR`	- B	-	✓	✓

fortsættes...

Element		FireFox 1.5	Opera 9	Adobe SVG Viewer 3.0
feGaussianBlur	- B	-	✓	✓
feImage	- B	-	✓	✓
feMerge	- B	-	✓	✓
feMergeNode	- B	-	✓	✓
feMorphology	- -	-	✓	✓
feOffset	- B	-	✓	✓
fePointLight	- -	-	✓	✓
feSpecularLighting	- -	-	✓	✓
feSpotLight	- -	-	✓	✓
feTile	- B	-	✓	✓
feTurbulence	- -	-	✓	✓
Cursor Module				
cursor	- -	-	-	-
Hyperlinking Module				
a	TB	✓ Implemented as an XBL binding - object is not of type SVGAElement. Only xlink:href, xlink:show, and xlink:target (as of Firefox 2) attributes implemented.	✓ Unsupported attributes: `target`	✓
View Module				
view	- B	-	✓ Unsupported attributes: `zoomAndPan`, `viewTarget`	✓ Unsupported attribute: `viewTarget`
Scripting Module				
script	- B	✓	✓ Unsupported attributes: `type`. The only supported type is 'text/ecmascript'	✓
Animation Module				
animate	TB	-	✓	✓ Unsupported attributes: `min`, `max`
animateColor	TB	-	✓	✓ Unsupported attributes: `min`, `max`
animateMotion	TB	-	✓	✓ Unsupported attributes: `min`, `max`
animateTransform	TB	-	✓	✓ Unsupported attributes: `min`, `max`

fortsættes...

Element		FireFox 1.5	Opera 9	Adobe SVG Viewer 3.0
mpath	TB	-	✓	✓
set	TB	-	✓	✓ Unsupported attributes: min, max
Font Module				
font	? ?	-	✓ Unsupported attributes: horiz-origin-x, horiz-origin-y, vert-origin-x, vert-origin-y, vert-adv-y	✓
font-face	TB	-	✓ Unsupported attributes: accent-height, bbox, cap-height, font-stretch, hanging, ideographic, mathematical, overline-position, overline-thickness panose-1, slope, stemh, stemv, strikethrough-position, strikethrough-thickness, underline-position, underline-thickness, v-alphabetic, v-hanging, v-ideographic, v-mathematical, widths, x-height	✓ Unsupported attributes: units-per-em, ascent, decent
glyph	TB	-	✓ Unsupported attributes: orientation, lang, vert-origin-x, vert-origin-y, vert-adv-y. Currently the only way to define the glyph is to use the 'd' attribute. Arbitrary child elements defining the glyph is unsupported.	✓ Unsupported attributes: glyph-name, han
missing-glyph	TB	-	✓ Unsupported attributes: vert-origin-x, vert-origin-y, vert-adv-y. Currently the only way to define the glyph is to use the 'd' attribute. Arbitrary child elements defining the glyph is unsupported.	✓ Unsupported attributes: horiz-adv-x, vert-adv-y, d
hkern	TB	-	✓	✓
vkern	- B	-	✓	✓
definition-src	- B	-	-	-
font-face-format	- B	-	-	-
font-face-name	TB	-	-	-
font-face-src	TB	-	-	-
font-face-uri	- B	-	✓	-
Extensibility Module				

fortsættes...

Element		FireFox 1.5		Opera 9		Adobe SVG Viewer 3.0
foreignObject	TB	✓	Implemented, but not built.	✓	Child elements of the foreignObject elements are not rendered, the only way to display something in the foreignObject is to use xlink:href to point to the content.	-

Bilag B

Kildekode

© UH Communications - Pga. rettigheder til kildekoden er denne ikke offentlig tilgængelig

Underskriftsside

Projekt:	**Webbaseret alarmmanager**
Afleveringsfrist:	*Fredag den 6. oktober 2006*
DTU vejleder:	Paul Fischer, IMM
Projekt udført af:	Kjartan Døj, s022338

_____

Denne rapport er modtaget af _____ den ____ / ____ - 2006 kl: ___:___