POHJOIS-KARJALAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Henri Lämsä

ALFRESCO SÄHKÖISEN ASIAKIRJAMUODOSTUSSUUNITELMAN TIETOJÄRJESTELMÄNÄ

Opinnäytetyö
2011 joulukuu

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Syyskuu 2011
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. 358-13-260-6800

Tekijä

Henri Lämsä

Alfresco sähköisen asiakirjamuodostussuunnitelman tietojärjestelmänä

Toimeksiantaja Arcusys Oy

Tiivistelmä

Työssä perehdyttiin sähköiseen arkistonmuodostussuunnitelmaan (eAMS), jota kunnat käyttävät siirryttäessä kokonaan sähköiseen asiakirjojen säilyttämiseen sekä hallintaan. EAMS:n tietojärjestelmän, joka vastaa sähköisen tiedon hallinnasta sekä käsittelystä, täytyy vastata SÄHKE2-standardin asettamia vaatimuksia. Näitä vaatimuksia lukuun ottamatta tietojärjestelmälle ei ole asetettu muita vaatimuksia. Tietojärjestelmä voi koostua yhdestä suuresta kokonaisuudesta tai useammasta pienestä. Työssä perehdyttiin siihen miten vapaan lähdekoodin sisällönhallintaohjelmisto Alfresco sopisi kyseessä olevaan tarkoitukseen osana tietojärjestelmää.

Työssä kartoitettiin SÄHKE2:n vaatimukset ja tutkittiin miten Alfresco voisi vastata näitä vaatimuksia. Alfrescon vahva puoli on siinä, että se on vapaan lähdekoodin järjestelmä, joka mahdollistaa sen muokkaamisen sekä laajentamisen. Tutkinnan pohjalta laadittiin demoympäristö, jossa Alfrescoa oli muokattu sekä laajennettu vastaamaan olennaisia
SÄHKE2:n vaatimuksia.

Kieli
suomi

Sivuja 57

Asiasanat
Sisällönhallinta, eAMS, Alfresco

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

THESIS
September 2011
Degree Programme in
Information Technology
Karjalankatu 3
FIN 80200 JOENSUU
FINLAND
Tel. 358-13-260-6800

Author

Henri Lämsä

Alfresco as an Information System of Electric Document Formation Project

Commissioned by Arcusys Oy

Abstract

The purpose of this thesis was to get familiarized with electric document formation project (eAMS) which is used by Finnish provinces when changing over to electric document sustainability and management. The information system of eAMS which handles management of electric information must correspond to requirements of SÄHKE2 standard. Apart from the requirements of SÄHKE2 there are no any additional requirements for the information system. The Information system can consist of just one entity or several smaller entities. In the thesis it was studied how an open source content management system Alfresco could suit as a component of the information system.

The requirements of SÄHKE2 were examined and how Alfresco could correspond to those requirements. One of the strong features of Alfresco is that it is an open source which makes it possible to modify and extend it. Demo environment in which Alfresco had been modified and extended to correspond some of the most relevant SÄHKE2 requirements was implemented based on that examination.

Language
Finnish

Pages 57

Keywords
content management, eAMS, Alfresco

Lyhenteet

AIIM	Association for Information and Image Management
DAM	Digital Asset Management
DM	Document Management
eAMS	Sähköinen arkistonmuodostussuunnitelma
ECM	Enterprise Content Management
HTML	Hypertext Markup Language
LDAP	Lightweight Directory Access Protocol
RM	Records Management
WCM	Web Content Management
XML	Extensible Markup Language

1. Johdanto

Organisaatioiden sisäisen informaation määrän jatkuvasti lisääntyessä yhä useampi organisaatio on siirtymässä perinteisistä sisällönhallinnasta fyysisessä muodossa kokonaan sähköiseen muotoon. Jotta informaation käsittely olisi tehokasta ja organisoitua, tarvitaan menetelmiä, joilla sitä voidaan hallita. Tästä syystä organisaatioiden sekä yritysten tarve hyville sisällönhallintajärjestelmille on yhä etenevässä kasvussa.

Tässä opinnäytetyössä perehdyttiin sisällönhallinnan syövereihin, arkistolaitoksen sähköisen arkistonmuodostussuunnitelmaan (eAMS) sekä erääseen avoimen lähdekoodin teknologioita käyttävään Enterprise Content Management alustaan, Alfrescoon. Lisäksi työssä toteutettiin testiympäristö sisällönhallintajärjestelmästä, joka noudattaa sähköistä arkistonmuodostussuunnitelmaa ja jossa alustana käytetään Alfrescoa.

Toimeksiantajana toimi Arcusys Oy, joka on vuonna 2003 perustettu tietotekniikan innovatiivinen palveluyritys. Yhtiön pääpaikka sijaitsee Joensuussa. Arcusys on erikoistunut teollisuuden, terveydenhuollon ja julkishallinnon IT-asiantuntijapalveluihin sekä tietojärjestelmäratkaisuihin.

Opinnäytetyön tavoitteena oli tutkia, miten Alfresco sopii käytettäväksi tietojärjestelmänä osana sähköistä arkistonmuodostussuunnitelmaa. Työssä selvitettiin voiko Alfrescoa laajentaa niin, että se vastaa eAMS:n asettamia vaatimuksia. Työssä perehdyttiin Alfrescon laajennettavuuteen ja rakennettiin testiympäristö Alfrescoon, jossa näitä vaatimuksia on toteutettu.

2. Sisällönhallinta

Sisältö voidaan helposti sekoittaa raakaan informaatioon tai silloin kun ollaan tekemisissä tietokoneen käyttämää informaation kanssa, voidaan puhua datasta. Sisältö ei kuitenkaan ole pelkkää dataa. Sisältö on informaatiota, joka on muutettu käytännölliseen muotoon ja jolle on annettu tietty tarkoitus. Sisältö on informaatiota, johon on sisällytetty tieto sen kontekstista. Näistä tiedoista käyte-

tään yleisesti nimitystä metadata. Sisällönhallintajärjestelmät käyttävät metadataa organisoimaan sekä hallitsemaan informaatiota. (Boiko 2005, 4.)

Hyvin lyhyesti voidaan todeta, että sisällönhallinta on julkaisujen organisointiprosessi. Sisällönhallintaa tarvitaan silloin, kun tiedon julkaisemisesta tulee liian suurta ja liian tärkeää, jotta se voitaisiin tehdä ilman rakennetta. Riippuen näkökulmasta sisällönhallinta voi tarkoittaa seuraavia asioita:

- liiketoiminnan näkökulmasta: sisällönhallinta tarjoaa liikearvoa
- analyyttisestä näkökulmasta: sisällönhallinta vakauttaa organisaatiollisia voimavaroja
- ammatillisesta näkökulmasta: sisällönhallinta yhdistää sisältöön liittyviä lainalaisuuksia
- prosessin näkökulmasta: sisällönhallinta kerää, hallinnoi sekä julkaisee tietoa
- teknisestä näkökulmasta: sisällönhallinta on tekninen infrastruktuuri

(Boiko 2005, 4.)

Yleisestä näkökulmasta voidaan todeta, että sisällönhallinta on organisaation julkaisujen hallintaa, jonka tarkoitus on parantaa ja tehostaa organisaation toimintaa. Tänä päivänä organisaatiossa liikkuvan informaation määrä on kasvanut suuresti ja se ei aina ole tehokkaasti organisoitua. Informaatio on silloin arvokkaimmillaan, kun se on nopeasti saatavissa ja se on esitetty oikeassa muodossa. Muuttamalla informaation sisällöksi ja liittämällä se sisällönhallintaan tiedosta tulee organisoitua sekä tehokkaampaa. Kun informaation käsittelystä ja organisoimisesta tulee tehokkaampaa, voidaan todeta sisällönhallinnan myös tehostavan organisaation toimintaa.

3. Yrityksen sisällönhallinta

Yrityksen sisällönhallinta, englanniksi Enterprise Content Management (ECM), on strategioita, tapoja sekä työkaluja, joita käytetään keräämään, hallinnoimaan, säilyttämään, suojaamaan sekä julkaisemaan sisältöä liittyen organisaation toimintaan. (AIIM ECM) Kun perinteiset arkistointi-, dokumenttien hallinta

sekä prosessikulkujen toiminnallisuudet on sulautettu uusiin tai käytetty laatimaan kokonaan uusia tuotteita, jotka yhdistävät web-pohjaisia komponentteja sekä perinteisiä tuotteita, voidaan katsoa sisällönhallinnasta tulleen ECM. ECM pyrkii siis määrittelemään itsensä niin, ettei se ole pelkästään yrityksen web-orientoitunut anti ulkomaailmaan, vaan sisältää kaiken rakenteellisen sekä rakenteettoman informaation yrityksen sisällä. (Kampffmeyer 2006, 4.)

Arkonyymi ECM:n määritelmä on muuttunut useaan kertaan viime vuosien aikana. Vuodesta 2003 lähtien AIIM määritteli ECM:n seuraavasti: "*The technologies used to capture, manage, store, deliver, and preserve information to support business processes*".

Vuonna 2005 tehtiin toinen määritelmä, jossa prosessimääritelmä jätettiin pois: "*Enterprise Content Management is the technologies, tools, and methods used to capture, manage, store, preserve, and deliver content across an enterprise." All the same BPM was accentuated by AIIM as essential component in white papers and posters*". (Kampffmeyer 2006, 4.)

ECM:n termi on melko hankala eikä sille ole olemassa yksiselitteistä selitystä. ECM myyjät käyttävät sitä terminä kuvailemaan sarjana sisältöön liittyvistä teknologioista, jotka sisältävät sellaisia teknologioita kuten DM, WCM, DAM, RM sekä Imaging. ECM:n harjoittajat kuvaavat ECM:n olevan vähemmän teknologiaa ja enemmän sitä kuinka tietoa kerätään, organisoidaan sekä jaetaan yrityksessä. (Pots 2008, 18).

3.1. Tiedonkeruu sekä hallinta

Tiedonkeruu sisältää toiminnallisuuksia sekä komponentteja analogisen sekä elektronisen informaation laatimiseen, keräämiseen, valmistelemiseen sekä prosessoimiseen. Tiedonkeruu jakautuu sekä manuaaliseen että automaattiseen tiedonkeruuseen. Manuaalinen tiedonkeruu voi sisältää informaatiota missä muodossa tahansa paperidokumenteista elektronisiin toimistodokumentteihin, sähköpostiin, lomakkeisiin, multimediatuotoksiin sekä digitaaliseen ääneen tai videoon. Automaattinen tiedonkeruu käyttää EDI- sekä XML-dokumentteja, liiketoiminta sekä ERP (Enterprise Resource Planning) – sovelluksia sekä ole-

massa olevia erityissovelluksia lähteenään tiedonkeruussa. (Kampffmeyer 2006, 30.)

Hallintakomponentit hallitsevat, käsittelevät sekä käyttävät informaatiota. Ne sisältävät

- tietokannan järjestelmänhallintaan sekä tiedonhakemiseen
- tiedonhakemisen valvontajärjestelmän takaamaan tietoturvaa.

Tavoite suljetussa ECM-järjestelmässä on tarjota nämä komponentit kaikille hallintasovelluksille, kuten dokumenttien hallinnalle, web-sisällönhallinnalle, asiahallinnalle sekä työnkululle. Jotta komponentit saataisiin yhdistettyä toisiinsa, niillä täytyy olla standardisoidut rajapinnat sekä tietoturva komponenttien väliselle keskustelulle. (Kampffmeyer 2006, 30.)

3.2. Tiedon säilyttäminen

Tiedon säilyttäminen jakautuu kahden pääryhmän komponentteihin: komponentteihin, joita käytetään väliaikaiseen säilyttämiseen sekä komponentteihin, joita käytetään pitkäaikaiseen säilyttämiseen. Väliaikaisen säilyttämisen komponentit jaetaan AIIM:n mukaan kolmeen kategoriaan: varastot, kirjastopalvelut sekä teknologiat. Näitä komponentteja pidetään toisinaan tiedostojärjestelmässä ja ne myös sisältävät turvallisuusteknologioita. Varastot sisältävät tiedostojärjestelmän, sisällönhallintajärjestelmän, tietokannan sekä datavarannot, jotka ovat tietokantoja monipuolisempia säilytysjärjestelmiä. Kirjastopalvelu huolehtii mm. informaation hakemisesta ja palauttamisesta, versionhallinnasta sekä sisällön integroinnista. Teknologiat pitävät sisällään tallennusmuodot, kuten kovalevyt, CD-levyt jne. (Kampffmeyer 2006, 54.)

Pitkäaikaisen säilyttämisen komponentit huolehtivat tiedon pitkäaikaisesta säilyttämisestä sekä varmuuskopioinnista. Tähän ryhmään kuuluu elektroninen sekä fyysinen informaatio, kuten paperilla oleva tieto. Elektroninen säilytys koostuu yhdistelmästä järjestelmänhallintakomponenteista kuten asianhallinnasta, dokumenttien hallinasta sekä kirjastopalveluista. (Kampffmeyer 2006, 59.)

3.3. Tiedon julkaiseminen sekä suojaaminen

Tiedon julkaisun komponentteja käytetään esittämään tietoa hallinta- sekä säilytyskomponenteista. Ne sisältävät myös toiminnallisuudet syöttämään tietoa järjestelmään. Julkaisun komponentit jakautuvat kolmeen ryhmään: tiedoston muodon muuttaminen, suojaus sekä jakelu. Tiedon muodon muuttamisen komponentit huolehtivat tiedoston muodon muuttamisesta toiseen, kuten doc-tiedostosta pdf-tiedostoksi. (Kampffmeyer 2006, 70.)

Tiedon suojaus pitää huolen dokumenttien alkuperäisyyden sekä tekijänoikeuksien varmistamisesta. Siihen kuuluu dokumentin sähköinen nimikirjoitus, jota käytetään dokumentin lähettäjän todentamiseen sekä tarkistamaan, ettei dokumentin sisältö ole muuttunut. Lisäksi tiedon suojaus hallinnoi dokumentin tekijänoikeuksia. Tekijänoikeudet todetaan elektronisella vesileimalla, jotka liitetään suoraan tiedostoon. (Kampffmeyer 2006, 70.)

Jakelun komponentteja ovat mm. sähköposti, datamedia, muistiot sekä julkaisut verkkosivuilla tai portaaleissa. Mahdollisia tuotoksia tai jakelumedioita ovat Internet, portaalit, sähköposti sekä faksi, datasiirrot EDI-, XML- tai muissa muodoissa, mobiililaitteet, datamediat (CD ja DVD), TV ja muu multimediapalvelu sekä paperi. Jakelun komponenttien tehtävä on tarjota informaatiota käyttäjälle parhaalla mahdollisella tavalla säilyttäen kuitenkin käyttötarkoituksensa niin pitkälle kuin mahdollista. (Kampffmeyer 2006, 71.)

3.4. Dokumenttien hallinta

Dokumenttien hallinta syntyi tarpeesta hallinnoida aina kasvavaa informaation määrää. Dokumenttien hallintajärjestelmien tarkoitus on säilyttää, tallentaa, hakea sekä poistaa informaatiota, joka säilytetään dokumentteina. Dokumenttien hallintajärjestelmä rakentuu varaston ympärille, jota käytetään hallinnoimaan kaikentyyppistä informaatiota, jolla voi olla arvoa organisaatiolle, sekä suojelemaan informaation menettämiseltä.

Tänä päivänä dokumenttien hallinnan on lähes jokaisen yrityksen ongelma, sillä jokainen yritys tuottaa tietoa, jonka hallinta hyötyy sellaisista ominaisuuksista, kuten versiointi, metatieto, tietoturva, haku sekä työkulku. Hyvä esimerkki klas-

sisesta dokumentin hallinnasta on teollisuusyrityksillä, joilla on laajat kehitys- sekä tuottolinjat. Niin kuin voi kuvitella, tällaiset yritykset voivat käsitellä jopa tuhansia dokumentteja päivässä. Näitä dokumentteja, jotka voivat olla usealla eri kielellä, laaditaan sekä käsitellään eri puolilla yritystä eri osakkaiden toimesta. (Pots 2008, 8.)

Ominaisuudet, joita toimiva dokumenttien hallinta tarvitsee toimiakseen, ovat

- helppo integrointi dokumenttien laatimistyökalujen kanssa
- turvallisuus
- kirjastopalvelut, kuten versiointi, metatiedot sekä haku
- työnkulku, joita käytetään prosessien automatisoimiseen
- skaalaus/luotettavuus
- kustomoitava käyttöliittymä. (Pots 2008, 9).

3.5. Web-sisällön hallinta

Web-sisällön hallinta on samankaltainen kuin dokumenttien hallinta. Ilmiselvin ero näiden välillä on se, että web-sisällön hallinnassa sisältö on tarkoitettu julkaistavaksi verkkosivuilla tai osana web-sovellusta. Muita eroavaisuuksia on

- sisällöntuottamistyökalut
- julkaisun erottaminen sisällöstä
- systemaattinen sisällön julkaiseminen.

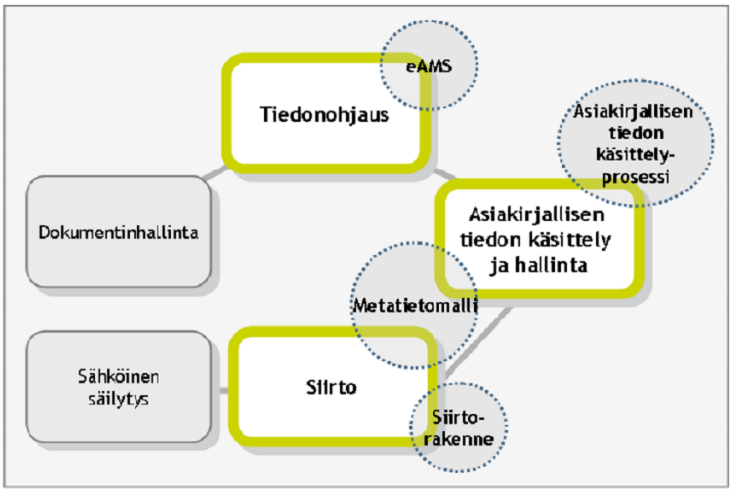
Suurin osa sisällönhallinnan ratkaisuista käsittelee tiedostoja, jotka on tehty toimistotyökaluilla. Web-sisällönhallinnassa käytetään erilaisia työkaluja tekstieditoreista grafiikantuottamisohjelmiin. Tämä tarkoittaa sitä, että Web-sisällön hallinnan ratkaisujen täytyy olla hyvin joustavia sen suhteen miten se integroituu sisällöntuottamistyökaluihin. (Pots 2008, 11.)

Web-sisällön hallinta ei vaadi erottelua sisällön esittämisen verkkosivuilla ja säilytyspaikan välillä. Kuitenkin moni ratkaisu hyötyy tästä periaatteesta, sillä se tekee sivuston uudelleen suunnittelusta helpompaa, lisäksi se helpottaa julkaisua useammista kanavista sekä mahdollistaa sisällön laatimisen tekniikkaan perehtymättömille ihmisille. Suurimmalta osin sisältö WCM:ssä sijaitsee varas-

tossa ja se haetaan järjestelmien ja sovellusten toimesta, jotka tarvitsevat sitä. Tämä voi tapahtua aikavälein automaattisesti, käyttäjän, työnkulun tai kaikkien näiden toimesta. (Pots 2008, 11-12.)

4. EAMS

EAMS on suunnitelma, jonka organisaatio käy läpi siirtyessään kokonaan sähköiseen dokumenttien hallintaan sekä säilyttämiseen. EAMS sisältää tehtäväluokitus- sekä metatietomallin, joita organisaatio käyttää pohjana tietojärjestelmää toteuttaessaan. Metatietomalli sisältää dokumenttien käsittelyyn sekä hallintaan liittyvät metatiedot. Näitä malleja organisaatio voi täydentää omien tarpeidensa mukaan. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008.) Kuvassa 1 esitellään asiakirjahallinnan perusrakenne.



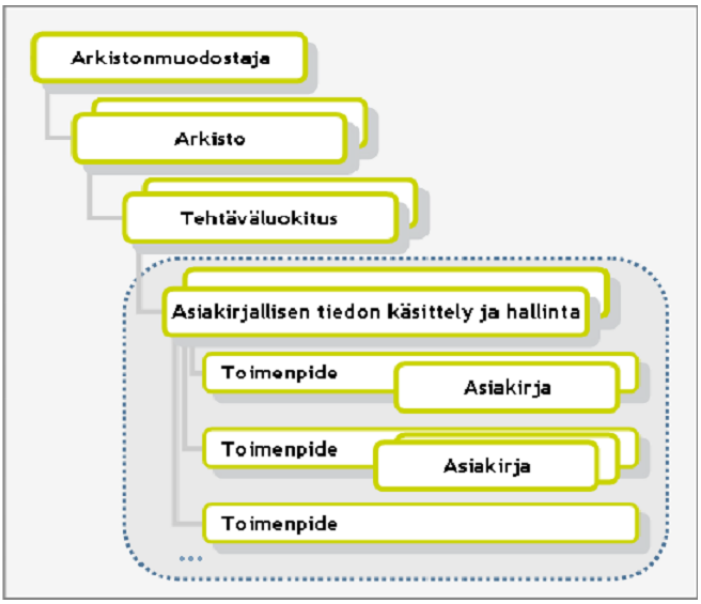
Kuva 1. Asiakirjahallinta (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008).

4.1. SÄHKE2

Vuoden 2006 alussa tuli voimaan arkistolaitoksen määräys asiankäsittelyjärjestelmään sisältyvien pysyvästi säilytettävien asiakirjallisten tietojen säilyttämisestä yksinomaan sähköisessä muodossa (KA 1486/40/2005). Määräyksessä esitetyt vaatimukset perustuivat arkistolaitoksessa 18.2.2005 hyväksyttyihin SÄHKE-määrityksiin. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008.)

SÄHKE2 asettaa vaatimukset asiakirjallisen tiedon säilyttämiselle sähköisessä muodossa. Sen tarkoitus on pääasiassa taata sujuva, luotettava sekä nykyaikainen tiedon säilyttäminen. Vuoden 2011 alusta alkaen voidaan hakea SÄHKE2-sertifikaattia mm. eAMS-tietojärjestelmälle. Myönnetty sertifikaatti on voimassa kolme vuotta.

SÄHKE2:n keskeiset tavoitteet ovat sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen. SÄHKE2 antaa reunaehdot käsittelyprosessien sähköistämiselle ja säilyttämiselle, jolloin päästään eroon paperiperusteisista käsittelyprosessista ja paperiarkistoinnista. Tavoitteena on saada tiedon käsittelystä sekä hallinnasta tietojärjestelmissä automaattista ja luotettavaa, tarpeeton tieto voidaan hävittää ohjatusti sekä sähköinen tieto on tallennettavissa pitkäaikaissäilytyksestä huolehtivaan järjestelmään. (Arkistolaitoksen SÄHKE2–normi sähköisen tiedonhallinnan edistäjänä, 2010) Kuvassa 2 on kuvattu SÄHKE2-hierarkia.



Kuva 2. SÄHKE2-hierarkia (Arkistolaitoksen SÄHKE2–normi sähköisen tiedonhallinnan edistäjänä, 2010).

4.2. Tehtäväluokitus

Asiakirjat liitetään tietojärjestelmässä organisaation tehtävien mukaiseen rakenteeseen eli tehtäväluokitukseen, joka toimii eAMS:n runkona. Tehtäväluokitukseen liitetään kuvaukset tehtävien käsittelyvaiheista ja niihin liittyvistä asiakirjallisista tiedoista ja asiakirjatyypeistä oletusmetatietoineen. Tehtäväluokitukseen sisältyvät käsittelyvaiheet mahdollistavat tiedonohjauksen toteutumisen, jolloin tiedonohjaus tuottaa tietojärjestelmään

- käsittelyvaiheen mukaisen toimenpiteen tyyppi-tiedon,
- tiedot käsittelyvaiheeseen sisältyvistä asiakirjatyypeistä ja
- käsittelyvaiheeseen sisältyvien asiakirjatyyppien oletusmetatiedot.

Tehtäväluokitus ja sen käsittelyvaiheet eivät korvaa työnkulku- ja prosessikuvauksia ja niiden sähköisiä toteutuksia, vaan tehtäväluokitus tarjoaa käsittelyvaihekohtaista ohjaustietoa. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 4-5.)

eAMS tarjoaa kaksi eri tapaa käsittelyprosessien kuvaukseen. Jos prosessit on kuvattu suoraan, tehtäväluokituksen mukaisiksi, voidaan prosessikuvaukset hyödyntää suoraan arkistonmuodostussuunnitelman käsittelyvaiheiksi. Mikäli organisaatio taas on kuvannut ns. perusprosesseja eli esimerkiksi lausuntoprosessin, päätösprosessin, muutoksenhakuprosessin, liitetään nämä perusprosessikuvaukset jokaisen tehtävän yhteyteen, joihin kyseinen perusprosessi sisältyy. (Arkistonmuodostussuunnitelma AMS 2011.)

Organisaation tehtävät jaetaan hierarkkisesti päätehtäviin, tehtäviin ja alatehtäviin. Käsittelyvaihekuvaukset liitetään hierarkian alimmalle tasolle. Päätehtävät jaetaan kahteentoista luokkaan seuraavasti:

- 00 Yleishallinto
- 01 Henkilöstöhallinto
- 02 Taloushallinto, verotus ja omaisuuden hallinta
- 03 Lainsäädäntö ja laillisuus
- 04 Kansainvälinen yhteistyö
- 05 Sosiaalihuolto ja sosiaaliturva

- 06 Terveydenhuolto
- 07 Järjestys ja turvallisuus
- 08 Maankäyttö, rakentaminen ja asuminen
- 09 Ympäristönsuojelu
- 10 Opetus-, tutkimus- ja sivistystoimi
- 11 Elinkeinopalvelut ja liiketoiminta

Organisaatiot voivat sitten syventää tehtäväluokitusta omien tarpeittensa mukaan. Luokitus numeroidaan juoksevasti siten, että jokainen taso kuvataan kahdella numerolla. (Sähköisen AMS:n laadintaprosessi 2007.)

Tehtävien käsittelyvaiheiden kuvaamisen yhteydessä kartoitetaan vaiheisiin liittyvät asiakirjalliset tiedot ja ne tietojärjestelmät, joissa tehtäviä ja asiakirjallisia tietoja käsitellään ja tallennetaan tai jonne ne rekisteröidään. Asiakirjalliset tiedot merkitään arkistonmuodostussuunnitelmaan sen käsittelyvaiheen yhteyteen, jossa se luodaan tai jonka seurauksena se organisaatioon saapuu. Tietojärjestelmistä muodostettavat näkymät ovat myös asiakirjallisia tietoja. Näkymien lisäksi arkistonmuodostussuunnitelmaan määritellään ne kentät (tietorakenteet), joista näkymät muodostetaan. (Sähköisen AMS:n laadintaprosessi 2007.)

Asiakirjallisen tiedon näkymät voivat olla lomakkeita, joissa käyttäjä tallentaa tietoa. Lomakkeiden käsitteiden olisi hyvä olla selkeitä ja yksinkertaisia niin, että käyttäjältä ei vaadita tarkkaa tietämystä asiakirjanhallinnon tuntemusta. Tällöin lomakkeen täyttäjän on helppo yksilöidä työtehtävissään syntyvät ja saapuvat asiakirjat, tietojärjestelmät ja mahdolliset uudistamistarpeet tai päivitystarpeet erityisesti säilytysaikojen, säilytysmuotojen ja julkisuutta koskevien tietojen kohdalla. (Sähköisen AMS:n laadintaprosessi 2007.)

4.3. Tiedonohjauksen periaatteet

EAMS ohjaa asiakirjallisen tiedon muodostumista, käsittelyä, hallintaa ja säilyttämistä. Sähköinen tiedonohjaus vaikuttaa kaikkiin tietojärjestelmiin, joissa asiakirjallista tietoa käsitellään. Tiedonohjaus toteutetaan tietojärjestelmiin niiden kehityshankkeiden yhteydessä.

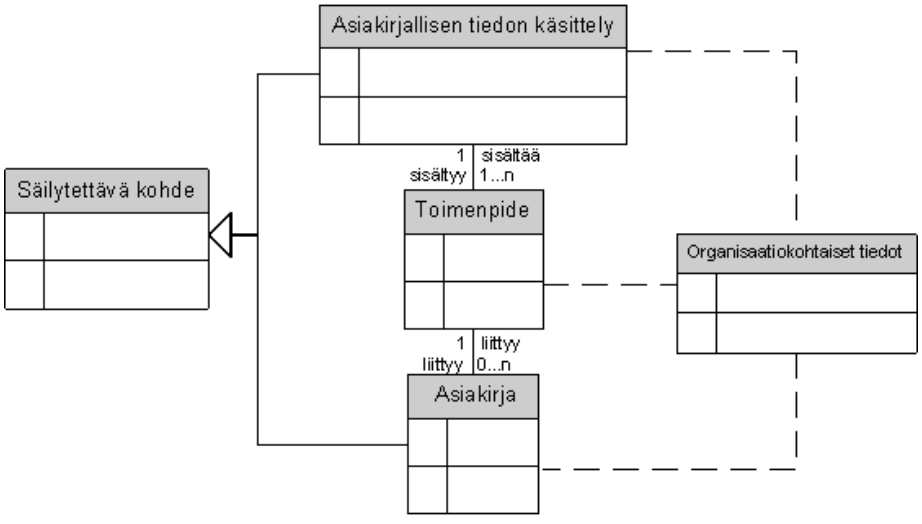
Tietojärjestelmä hyödyntää tiedonohjausta, joka on edellytys sähköisten asiakirjallisten tietojen hallinnalle. Tietojärjestelmän ja tiedonohjauksen kesken välitetään tehtäväluokitukseen liittyvien metatietoarvojen lisäksi myös ohjaustietoja käsittelyvaiheista, niihin liittyvistä asiakirjatyypeistä ja metatietoarvojen määräytymiseen liittyvistä käsittelysäännöistä. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 4-5.)

EAMS vaatii sen, että

- metatietoarvojen tallentaminen ja muuttaminen on sidottu käyttöoikeuksiin
- tietojärjestelmän metatietoarvot määräytyvät eAMS:n metatietoarvoista automaattisesti eAMS:n määriteltyjen käsittelysääntöjen mukaisesti. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 10.)

4.4. Metatiedot

SÄHKE2-vaatimusten mukainen metatietomalli sisältää asiakirjallisen tiedon käsittelyn ja hallinnan metatiedot. Kuvassa 3 on kuvattu SÄHKE2:n metatietomalli.



Kuva 3. Metatietomalli

Metatietomallin yhteiset metatiedot on kuvattu säilytettävä kohde nimisenä kokonaisuutena. Sillä tarkoitetaan yhteisiä metatietoja, joita käytetään alikirjallisen tiedon käsittelyn, toimenpiteen ja asiakirjan tasolla. Asiakirjan tiedon käsittelyyn eli käsittelyprosessiin kuuluu yleensä useita käsittelyvaiheita, joihin taas voi kuulua useampi asiakirja. Näitä kokonaisuuksia voidaan laajentaa organisaatiokohtaisilla tiedoilla. (Metatietomalli 2008)

EAMS vaatii

- Tehtävän
- Toimenpiteen ja asiakirjan tyypin
- Julkisuusluokan
- Tilan
- Henkilötietoja
- Säilytysajan pituuden sekä säilytysajan perusteen
- Salassapitoajan, salassapidon perusteen, suojaustason, turvallisuusluokan
- Käyttäjäryhmän

(Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 4-5)

4.5. Käsittelyprosessi

EAMS:ssa asiakirjat liitetään käsittelyprosessiin. Käsittelyprosessi koostuu organisaatiokohtaisista useista käsittelyvaiheista, joita voi olla, vaikka vireillepano, käsittely sekä päätös. Käsittelyvaiheiden edetessä asiakirjoihin tallennetaan metatietoja. Näin voidaan automatisoida sekä ohjata käsittelyn kulkua. Muun muassa säilytysaika voi olla sidottuna käsittelyvaiheeseen. Kun käsittelyvaihe, kuten päätös, on suoritettu, liitetään asiakirjaan säilytysaika, jonka umpeutuessa asiakirja automaattisesti poistetaan tietojärjestelmästä. Käsittelyprosessilla saadaan käyttäjäriippuvaisten tallennuksien määrää vähennettyä. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 12)

Käsittelyprosessin alkaessa siihen tallennetaan toimenpiteiden metatiedot. Niin ikään toimenpiteistä tallennetaan metatiedot tietojärjestelmään, kuten esimerkiksi toimija, päivämäärät jne. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 12)

EAMS:n määrittelemiä vaatimuksia käsittelyprosessille: Käsittelyprosessit kuvataan JHS 152 – suositukseen sisältyvällä työnkulkukuvaustasolla. Kuvauksessa huomioidaan myös prosessiin liittyvät tiedot ja käyttöoikeudet, Käsittelyprosessi määräytyy tehtäväluokan perusteella. Käsittelyprosessille tuotetaan yksilöivä tunnus tietojärjestelmästä. (Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen 2008, 12)

4.6. Elinkaari

Asiakirjoilla on elinkaari, jonka ne käyvät läpi sen laatimisesta aina hävittämiseen asti. Asiakirjan metatiedot ohjaavat elinkaarta. Näillä määrätään milloin asiakirjan tiedot salataan ja milloin asiakirja päätyy julkiseksi niin, että kellä tahansa on oikeus nähdä asiakirja. Asiakirjalle annetaan säilytysaika joko sen laatimisvaiheessa, käsittelyvaiheen suoriuduttua tai käyttäjän syötteestä. Kun säilytysaika on annettu, lasketaan asiakirjalle hävittämispäivämäärä, jonka umpeuduttua asiakirja metatietoineen hävitetään pysyvästi automaattisesti.

Asiakirjaan on myös tallennettu metatieto siitä kenellä on oikeus nähdä ja käsitellä asiakirjaa. Nämä oikeudet voivat muuttua asiakirjan elinkaaren aikana. Esimerkiksi asiakirja voi muuttua julkiseksi tietyn ajanjakson jälkeen tai sitten tietyn käsittelyvaiheen jälkeen. Kun asiakirja muuttuu julkiseksi, rajoitukset puretaan automaattisesti niin, että asiakirjat ovat kaikille nähtävissä. Sama pätee myös silloin, kun asiakirja merkitään salaiseksi, jolloin oikeuksia rajoitetaan sen mukaan, mikä salassapitoluokka asiakirjalle on asetettu. Asiakirjan metatietoihin on tallennettu tieto siitä käyttäjäryhmästä, jolla on oikeudet asiakirjan käsittelyyn.

5. Alfresco

Alfresco on johtava vapaan lähdekoodin alusta ECM:lle. Alfrescon ensimmäinen tuote julkaistiin kesäkuussa 2005 ja sen jälkeen sen kehitys on ollut huikeaa. Alusta on hyvässä vauhdissa toteuttamaan visionsa tulla varteenotettavaksi vaihtoehdoksi perinteisille vaihtoehdoille, jotka eivät pysy perässä sille innovaatiolle mitä vapaan lähdekoodin komponentit tarjoavat. (Pots 2008)

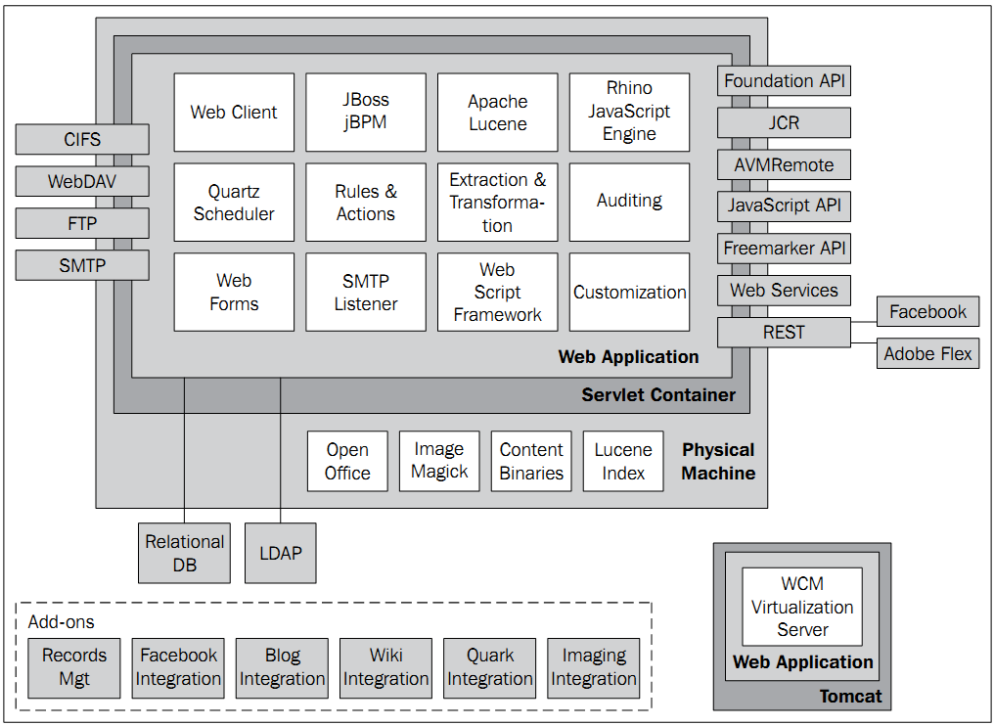
Alfresco tarjoaa ominaisuuksia, kuten DM, WC, RM, Activiti työnkulku, Lucene-haku.

5.1. Korkean tason arkkitehtuuri

Moni Alfrescon kilpailijoista käyttää suljetun lähdekoodin teknologioita, joista osa voi olla toistensa kilpailijoita. Näitä komponentteja on hankittu ja integroitu ajan kuluessa. Jotkin näistä ovat käyneet läpi massiivisen infrastruktuuriuudistukset vuosien kuluessa päätyen hiukan erikoiseen lopputulokseen. Alfresco ei kärsi näistä haitoista. Alfrescon arkkitehtuuri

- on suhteellisen yksinkertainen
- on rakennettu käyttäen viimeisimpiä runkoja sekä vapaan lähdekoodin komponentteja
- tukee useita tärkeitä sisällönhallintaan liittyviä standardeja.

Kuvassa 4 on kuvattu Alfrescon korkean tason arkkitehtuuri:



Kuva 4. Alfrescon korkean tason arkkitehtuuri. (Pots 2008, 18.)

Tärkeitä Alfrescon ominaisuuksia

- Sisällön viemiseen varastoon sekä tuomiseen ulos on monia eri tapoja käyttäen protokollia (kuvattu diagrammissa vasemmalla) tai API:a (kuvattu diagrammissa oikealla).
- Alfresco ajetaan web-sovelluksena servlet containerissa.
- Muokkaukset sekä laajennukset ajetaan osana Alfrsecon verkkosovellusta. Laajennusmekanismi pitää muokkaukset erillään ydintuotteesta pitääkseen tiedostot ehjinä päivityksessä.
- Metadata sijaitsee relaatiotietokannassa, kun taas sisältötiedostot sijaitsevat tiedostojärjestelmässä.
- VCM virtuaalipalvelin on Tomcatin instanssi. (Pots 2008, 18–19)

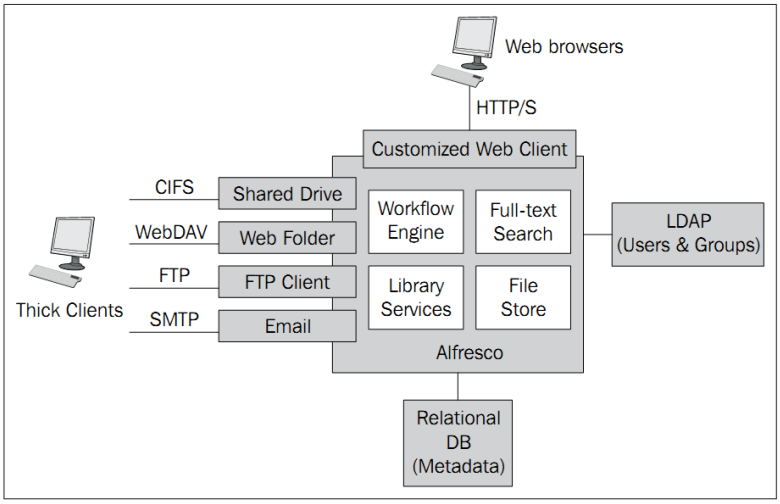
5.2. Vapaan lähdekoodin komponentit sekä protokollat

Alfresco koostuu monista vapaan lähdekoodin komponenteista. Hyvä puoli näiden käyttämisessä on se, että ei tarvitse keksiä pyörää uudelleen. Yleisesti myös vapaan lähdekoodin komponentit on hyvin standardisoitu, sillä ne on tehty

varta vasten integroitumaan muiden vapaan lähdekoodin komponenttien kanssa. Joitain tärkeimpiä vapaan lähdekoodin komponentteja Alfrescossa on: Lucene, Hibernate, Mozilla Rhino, Spring, sekä Jboss jBPM. Alfresco tukee protokollia, kuten FTP, WebDAV, CIFS, JCR API, Portlet API, SOAP, Xforms, XML Schema, XSLT, XSL:FO sekä LDAP. (Pots 2008, 19–20.)

5.3. Dokumenttien hallinta Alfrescossa

Seuraava diagrammi näyttää esimerkin korkean tason arkkitehtuurista, jolla DM voitaisiin toteuttaa Alfrescossa:



Kuva 5. Alfresco DM-arkkitehtuuri (Pots 2008, 10.)

Alfresco tukee eri työkalujen, kuten Windows Officen sekä Windwos Explorerin integroinnin protokollien, kuten CIFS ja LDAP kautta. Alfresco tallentaa metadatan relaatiotietokantaan, kun taas varsinaiset sisältötiedostot tallennetaan tiedostojärjestelmään. (Pots 2008, 10.)

6. Alfrescon laajentaminen

Alfresco on tehty helposti laajennettavaksi. Ohjelmasta löytyy erillinen tiedostorakenne, joka on tehty varta vasten laajennuksille sekä muokkauksille. Näin kaikki laajennuksen tiedostot ovat erillään Alfrescon omasta rakenteesta ja siksi

laajentaminen on helposti hallittavissa ja laajennukset eivät rikkoonnu mm. silloin, kun Alfrescoa päivitetään. Alfrescossa on myös mahdollista laajentaa sen omia moduuleja perimällä ne. Näin myös nämä laajennukset pysyvät itsenäisinä erillisinä kokonaisuuksina.

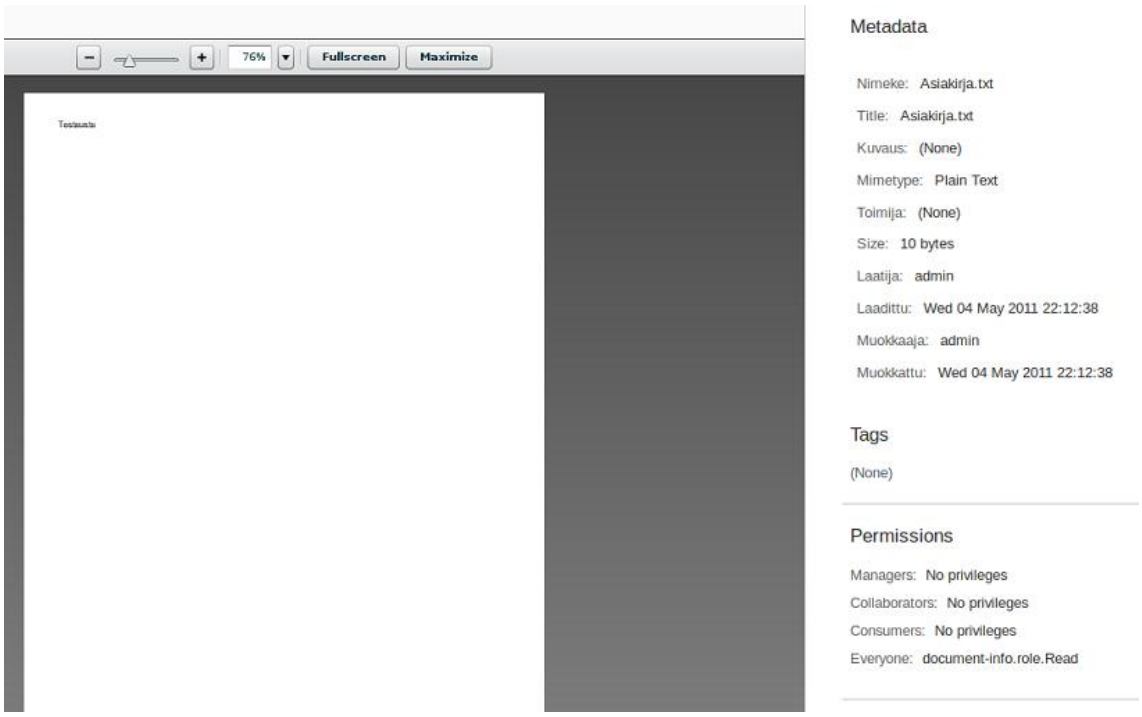
6.1. Datatiedostorakenne

Tässä luvussa selvitetään Alfrescon rakenne niin, että ymmärrettäisiin miten Alfrescon tietorakenne toimii ja niin ymmärrettäisiin miten laajentaa Alfrescoa.

6.2. Sisältömalli

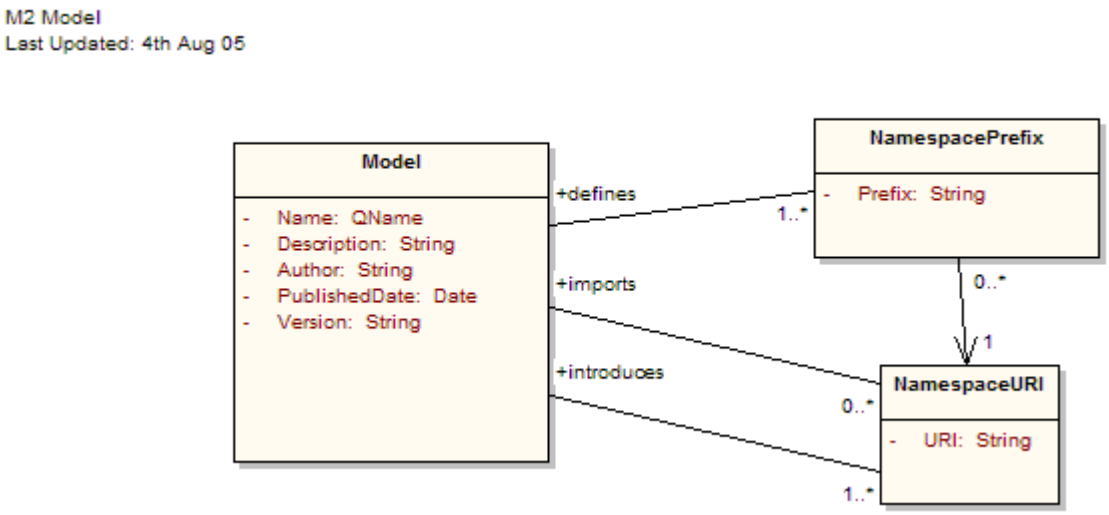
Hyvin pelkistetysti Sisältömalli on kokoelma siihen liittyviä sisältötyyppejä sekä – piirteitä ja niiden ominaisuuksia. Sisältötyypit ja sisältöpiirteet liitetään Alfrescossa tiedostoihin ja niiden tarkoitus on kuvata tiedostoa ominaisuuksien eli metatiedoin. Sisältötyyppi voisi olla, vaikka asiakirja ja sen ominaisuuksia olisivat asiakirjaa kuvaavat metatiedot, kuten nimi, tekijä, laatimispäivämäärä jne. Sisältötyyppejä on erilaisia riippuen tiedostosta, jota ne kuvaavat. Esimerkiksi MP3-tiedostolle voisi olla olemassa omanlaisensa sisältötyyppi. Tämä sisältötyyppi pitäisi sisällään sellaisia ominaisuuksia, jotka kuvaisivat MP3-tiedostolle ominaisia metatietoja, joita ei välttämättä tarvittaisi kuvaamaan sellaista tiedostoa kuten asiakirja. Näitä metatietoja voisi olla mm. albumin nimi, laulun nimi, säveltäjä jne.

Myös asiakirjoille voidaan määritellä useampia erilaisia sisältötyyppejä, kuten esimerkiksi hakemus tai pöytäkirja. Näille ominaisuuksille voidaan antaa oletusarvot, mutta ne eivät välttämättä sisällä alkuunsa arvoja lainkaan, vaan niille tuodaan arvot mm. käyttäjän syötteestä tai jonkin tapahtuman yhteydessä, kuten asiakirjan laatimisesta. Näitä arvoja voidaan muokata ja niitä voidaan tulostaa Webb-selaimessa erilaisissa näkymissä. Sisältömallit pitävät sisällään näitä sisältötyyppejä sekä sisältöpiirteitä ja ne on toteutettu XML-tiedostoina. Kuvassa 6 on näkymä asiakirjan metatiedoista.



Kuva 6. Asiakirjan metatiedot

Sisältöpiirteet, eli aspectit eroavat sisältötyypeistä niin, että niiden ei ole tarkoitus kuvata mitään tietynlaista tiedostotyyppiä, vaan ne ovat yleismaallisempia. Sisältöpiirteitä voidaan liittää sisältötyyppeihin. Kuvassa 7 on kuvattu Alfrescon sisältömalli.

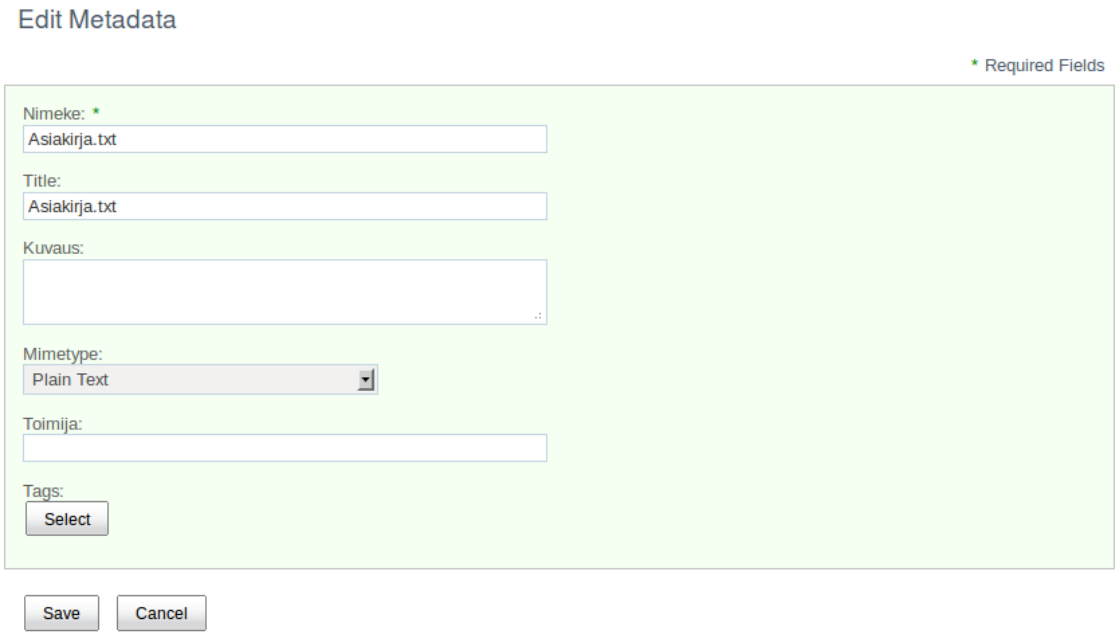


Kuva 7. Sisältömalli

Kaikki sisältömallit noudattavat tiettyä kaavaa. Jokainen malli aloitetaan esittelemällä se:

```
<model                                                name="cm:contentmodel"
xmlns="http://www.alfresco.org/model/dictionary/1.0">
   <description>Alfresco Content Model</description>
   <author>Alfresco</author>
   <published>2005-06-03</published>
   <version>1.0</version>
   ...
```

Ylhäällä oleva esimerkki on Alfrsecon omasta mallista *contentmodel* (ja myös seuraavat esimerkit tässä luvussa). Malli avataan model-tagillä, jossa mallille annetaan nimi. Kuvassa 8 on asiakirjan metatietojen muokkausnäkymä.



Kuva 8. Metadatan editointinäkymä

6.3. Nimitilat

Sisältömallissa voidaan viitata määritelmiin, kuten ominaisuuksiin ja piirteisiin. Sisältömalli yksilöidään nimitiloilla. Nimiavaruus-URI (NamespaceURI) on nimitilan nimi ja Nimitilan etuliite (NamspacePrefix) on määritelmä, jota käytetään viittamaan sisältömalliin mm. silloin, kun viitataan sen sisältämiin arvoihin. Näin voidaan käyttää eri sisältömallien arvoissa samoja nimimääritelmiä ja kuitenkin erotella ne toisistaan.

Nimitiloja voidaan määritellä kuinka paljon tahansa, mutta yleensä on järkevintä määritellä yksi nimitila sisältömallia kohden, sillä jos jostain syystä pitäisi määritellä toinen nimitila, olisi syytä harkita toisen mallin luomista. Nimitiloja voidaan myös tuoda toisista malleista, jolloin voidaan viitata sen sisältämiin tietorakenteisiin.

Alhaalla olevassa esimerkissä tuodaan sisältömalli dictionary sekä annetaan mallille content nimitila.

```
<imports>
   <import           uri="http://www.alfresco.org/model/dictionary/1.0"
prefix="d"/>
</imports>
<namespaces>
   <namespace           uri="http://www.alfresco.org/model/content/1.0"
prefix="cm"/>
</namespaces>
```

6.4. Sisältötyypit

Sisältötyyppien on tarkoitus kuvata tietynlaista tiedostoa ominaisuuksin. Sisältötyypit voivat periä toisen sisältötyypin, mutta yksi tiedosto ei voi koskaan omistaa enempää kuin yhtä sisältötyyppiä. Periessään sisältötyyppi saa kaikki ne ominaisuudet mitä perittävällä sisältötyypillä on. Periminen tekee tyyppien määrittelemisestä joustavampaa. Esimerkki periytymisestä voisi olla:

Sisältö -> Asiakirja -> Pöytäkirja

Jokainen sisältötyyppi pitää sisällään joukon ominaisuuksia. Sisältötyyppi voi omistaa sarjan omia ominaisuuksia tai sisältöpiirteitä tai sekä että.

```
<types>
   <type name="cm:cmobject">
      <title>Object</title>
      <parent>sys:base</parent>
      <properties>
         <property name="cm:name">
            <type>d:text</type>
         </property>
      </properties>
      <mandatory-aspects>
         <aspect>cm:auditable</aspect>
      </mandatory-aspects>
   </type>
   ...
```

- **types**

- **type name**
  - **title** - Tyypin titteli
  - **parent** - Tyyppi, jonka tämä tyyppi perii
  - **properties**
    - **property name** - Ominaisuus ja sen nimi, jota käytetään siihen viittaamiseen
      - **type** - Ominaisuuden tyyppi, joka voi olla mm.: d:text, d:int, d:long, d:float, d:double, d:date, d:datetime, d:boolean
    - **<mandatory-aspects>** - piirteet, jotka liitetään sisältötyyppiin
      - **<aspect>**

6.5. Piirteet

Piirteet ovat sarja ominaisuuksia, joita voidaan käyttää laajentamaan sisältötyypin ominaisuuksia. Piirteitä voidaan sisällyttää sisältötyyppeihin alusta asti, mutta niitä voidaan myös sisällyttää tiettyjen tapahtumien yhteydessä. Tähän käytetään yleensä JavaScriptiä tai Javaa.

Esimerkki piirteiden käyttämisestä voisi olla, vaikka tiedoston lähettäminen. Jos olisi kaksi samankaltaista asiakirjaa, joilla olisi samanlaiset metatiedot ja niistä toinen lähetettäisiin jonnekin toiseen osastoon, kuten tilauskäsittelyyn, toimenpiteestä olisi hyvä liittää metatiedot, kuten lähetyspäivämäärä, kohde sekä alkuperä. Mutta näitä metatietoja olisi turha sisällyttää siihen asiakirjaan, jonka on tarkoitus jäädä säilytettäväksi. Niinpä olisi järkevämpää tehdä näistä metatiedoista piirre, jonka nimi voisi olla, vaikka lähetetty. Näin tämä piirre voitaisiin liittää asiakirjaan siinä yhteydessä, kun se lähetetään.

Piirteet määritellään seuraavasti:

```
<aspects>
      <aspect name="cm:auditable">
         <title>Auditable</title>
         <properties>
            <property name="cm:created">
               <type>d:datetime</type>
            </property>
            <property name="cm:creator">
```

```
            <type>d:text</type>
         </property>
         <property name="cm:modified">
            <type>d:datetime</type>
         </property>
         <property name="cm:modifier">
            <type>d:text</type>
         </property>
         <property name="cm:accessed">
            <type>d:datetime</type>
         </property>
      </properties>
   </aspect>
   ...
```

- **aspects** – aloitetaan tila piirteille
    - **aspect name** – aloitetaan piirre ja nimetään se
        - **title** – piirteen titteli eli se, joka nähdään tulostettaessa
        - **properties** – aloitetaan piirteen ominaisuudet

6.6. Assosiaatiot

Kun luodaan assosiaatio, operaatiot, jotka kohdistuvat yhteen, heijastuvat myös niihin, joiden välillä on assosiaatioyhteys operaation kohdistuvan kanssa. Näitä operaatioita ovat mm. poistaminen.

6.6.1. Lapsiassosiaatiot

Lapsiassosiaatiossa assosiaatiot on luotu vanhemman ja lapsien välille niin, että operaatiot kulkeutuvat vanhemmasta lapsiin, mutta ei toisinpäin tai lapselta lapselle. Näin siis, jos vanhempi poistetaan poistuvat myös lapset. On mahdollista määritellä onko lapsen nimi uniikki vanhemmalle vai ei eli voiko vanhempi omistaa useampia samannimisiä lapsia. Voidaan myös määrittää kulkeutuvatko muokkauksen aikaleimat lapselta vanhemmalle poikkeuksena vallitsevaan sääntöön, jonka mukaan operaatiot eivät periydy lapselta vanhemmalle.

Tätä assosiaatiomenetelmää käytetään yleisesti mm. kansioiden välillä. Kun ylempänä polkua oleva kansio poistetaan, poistuu myös sen sisältämät kansiot. Kuitenkin, assosiaatio voidaan luoda minkä tyyppisten noodien välille tahansa.

```
<type name="cm:folder">
      <title>Folder</title>
      <parent>cm:cmobject</parent>
      <associations>
         <child-association name="cm:contains">
            <source>
               <mandatory>false</mandatory>
               <many>false</many>
            </source>
            <target>
               <class>sys:base</class>
               <mandatory>false</mandatory>
               <many>true</many>
            </target>
            <duplicate>false</duplicate>
            <propagateTimestamps>true</propagateTimestamps>
         </child-association>
      </associations>
   </type>
```

- **associations**
  - **child-association**
    - **source**
      - **mandatory** – true tai false, onko lapsella pakko olla vanhempi
      - **many** – true tai false, voiko lapsella olla useampi vanhempi
    - **target**
      - **class** – tyypin nimi, mihin sisältötyyppiin assossiaatio vaikuttaa
      - **mandatory** – true tai false, onko vanhemmalla pakko olla lapsia
      - **many** – true tai false, voiko olla useampi lapsi
    - **duplicate** – true tai false, voiko olla samannimisiä lapsia
    - **propagateTimestamps** – true tai false, kulkeutuvatko muokkauksien aikaleimat lapselta vanhemmalle

6.6.2. Vertaisassosiaatiot

On myös mahdollista määritellä vertaisassosiaatioita, mutta tässä tapauksessa poistaminen ei kulkeudu noodien välillä eikä Alfrescon hakukieli vielä tue liitäntöjä assosiaatioiden välillä eli hakukoneen löytäessä noodin ei se löydä samalla haulla assosioituneita noodeja.

6.7. Mallin rekisteröiminen

Kun varanto (reposoitory) on käynnistynyt, se lukee kaikki rekisteröidyt mallit ja kääntää ne. Tässä kohtaa repository voi alkaa hallinnoida sisältöä malleissa. Mallit rekisteröidään xml-tiedostoissa, jotka päättyvät -context liitteeseen. Esim. myModel-context.xml. Alfrescon esilatausohjelma lukee tiedostot tietyistä ennalta määrätyistä hakemistopoluista. Esimerkiksi Alfrescon omat mallit luetaan hakemistosta: projects\repository\config\alfresco ja ne on rekisteröity tiedostoon nimeltä: core-services-context.xml. Malli rekisteröidään beaneissä xml-tiedostossa. Laajentavat mallit luetaan niille omasta varatusta tiedostopolusta projects\repository\config\alfresco\extension.

```
<bean                                   id="extension.dictionaryBootstrap"
parent="dictionaryModelBootstrap" depends-on="dictionaryBootstrap">
        <property name="models">
            <list>
                <value>alfresco/extension/exampleModel.xml</value>
            </list>
        </property>
    </bean>
```

- **bean** – aloitetaan bean ja annetaan sille nimi. Etuilitte extension kertoo kyseessä olevan laajentavan mallin.
- **parent** – vanhempi metamalli ja sen riippuvuudet
- **property** – määritellään mitä ollaan rekisteröimässä, tässä tapauksessa mallia
- **list** – lista malleista, voidaan määritellä useampi samaan papuun
    - **value** – tiedostopolku, josta bean löytyy

Jos malleja ei rekisteröidä, ne jäävät kokonaan lukematta ja niitä ei voida silloin käyttää ollenkaan.

6.8. Konfiguraatio

Konfiguraatiotiedostoilla muokataan pääasiassa käyttöliittymän näkymiä. Konfiguraatiotiedostot ovat xml-tiedostoja, jotka päättyvät – config päätteeseen.

Konfiguraatiotiedostoilla voidaan mm. lisätä kieliä sisäänkirjautumissivulle, jolloin käyttäjä voi vaihtaa kielen, lisätä omia menuikoneita käyttöliittymään sekä näyttää kustomoituja metatietokenttiä.

Konfiguraatiotiedostoissa voidaan joko laajentaa valmiita konfiguraatioita, korvata entuudestaan olemassa olevia konfiguraatioita tai luoda omia konfiguraatioita. Jos halutaan korvata Alfrescon jokin oma konfiguraatio-elementti, lisätään config-tagiin arvo replace = true, joka kertoo Alfrescolle, että kyseinen elementti korvaa kaikki edelliset elementit.

```
<config evaluator="xx" condition="yy" replace="true">
```

Esimerkiksi seuraava konfiguraatioelementti korvaa kielilistan sisäänkirjautumissivulla:

```
<config        evaluator="string-compare"        condition="Languages"
replace="true">
   <languages>
      <language locale="fr_FR">French</language>
      <language locale="de_DE">German</language>
   </languages>
</config>
```

Toisaalta, jos tahdotaan vain laajentaa jo entuudestaan valmista konfiguraatiota, onnistuu se seuraavasti:

```
   <config evaluator="string-compare" condition="Languages">
      <languages>
         <language locale="fr_FR">French</language>
      </languages>
   </config>
```

Tämä määritelmä lisäisi ranskan kielen listaan eikä vaikuttaisi ollenkaan siinä entuudestaan oleviin kieliin.

Jos tahdotaan korvata koko konfiguraatiotiedosto, voidaan se tehdä ylikirjoittamalla se spring bean, joka määrittelee konfiguraatiolähteet, – context-tiedostoissa. Jokainen – context tiedosto luetaan automaattisesti extension-tiedostopolun alta ja siellä olevat beanit rekisteröidään aivan kuten mallien rekisteröimisessä.

```
<bean                                          id="webClientConfigSource"
class="org.alfresco.config.source.UrlConfigSource">
   <constructor-arg>
      <list>
         <value>classpath:alfresco/web-client-config.xml</value>
         <value>classpath:alfresco/web-client-config-
dialogs.xml</value>
         <value>classpath:alfresco/web-client-config-
wizards.xml</value>
         <value>classpath:alfresco/web-client-config-
properties.xml</value>
```

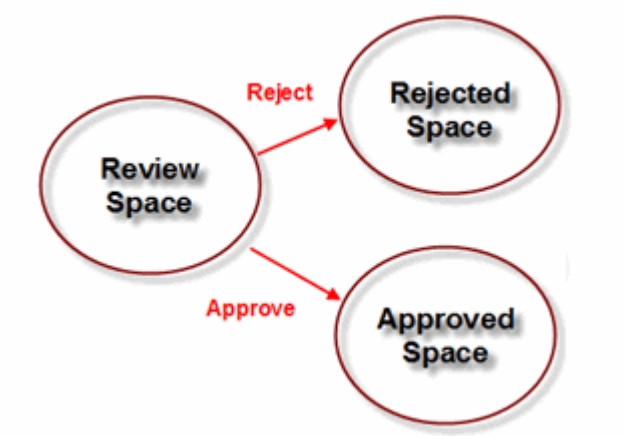
```
         <value>classpath:alfresco/web-client-config-
navigation.xml</value>
         <value>classpath:alfresco/web-client-config-wcm.xml</value>
         <value>classpath:alfresco/web-client-config-
actions.xml</value>
         <value>classpath:alfresco/web-client-config-forum-
actions.xml</value>
         <value>classpath:alfresco/web-client-config-wcm-
actions.xml</value>
         <value>classpath:alfresco/web-client-config-workflow-
actions.xml</value>
         <value>classpath:alfresco/extension/your-web-config-
file.xml</value>
         <value>file:c:\alfresco\your-other-web-config-
file.xml</value>
      </list>
   </constructor-arg>
</bean>
```

6.9. Työnkulku

Työnkulun tarkoitus on automatisoida tehtäviä. Alfresco tarjoaa kaksi erilaista vaihtoehtoa työnkulun toteuttamiseen: yksinkertainen työnkulku sekä kehittynyt työnkulku.

6.10. Yksinkertainen työnkulku

Yksinkertaisessa työnkulussa dokumentti siirtyy tilasta toiseen ja siihen voidaan määritellä keillä on oikeus muokata dokumenttia ja siirtää se seuraavaan tilaan. Alfrescossa on käytössä esimerkki yksinkertaisesta työnkulusta, joka on kuvattu kuvassa 9.



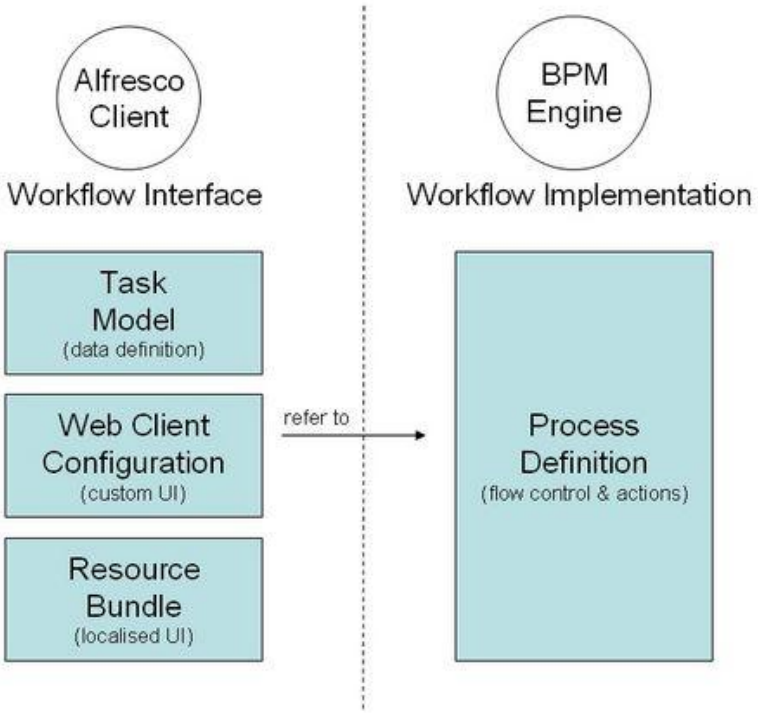
Kuva 9. Yksinkertainen työnkulku

Tässä työnkulussa dokumentti siirretään odottamaan tarkastusta ”Review space”, jonka jälkeen, riippuen hyväksytäänkö vai hylätäänkö dokumentti, se siirtyy joko tilaan ”Rejected Space”, hylätty tai ”Approved Space”, Hyväksytty. Yksinkertaisia työnkulkuja voidaan toteuttaa itse ja niissä voi olla kuinka paljon tiloja tahansa.

6.11. Kehittynyt työnkulku

Kehittynyt Työnkulku koostuu sarjasta toimenpiteitä ja siihen voi kuulua yksi tai useampi dokumentti. Työnkulku voidaan aloittaa automaattisesti tai käyttäjän toimesta ja sille määrätään käyttäjä. Kun käyttäjä, joka on valtuutettu työnkulkuun, on suorittanut toimenpiteen, hän merkitsee sen suoritetuksi, jonka jälkeen työnkulku siirtyy seuraavaan toimenpiteeseen. Kun viimeinen toimenpide on suoritettu, työnkulku poistuu ja se katsotaan suoritetuksi. Työnkulusta sekä sen toimenpiteistä tallentuu tietoja Alfrescoon ja niitä voidaan tarkkailla työnkulun edetessä.

Kehittynyt työnkulku muodostuu seuraavista osista:



Kuva 10. Kehittynyt työnkulku

Prosessin määrittäminen (Process Definition) määrittelee työnkulun askeleet ja siirtymät (valinnat). Askeleet voivat olla käyttäjän tai järjestelmän suorittamia. Alfresco käyttää jBPM prosessi-ohjelmistoa, joka käyttää omaa jPDL-kieltänsä. jPDL-kieli on Alfrescossa toteutettu xml-tiedostoissa.

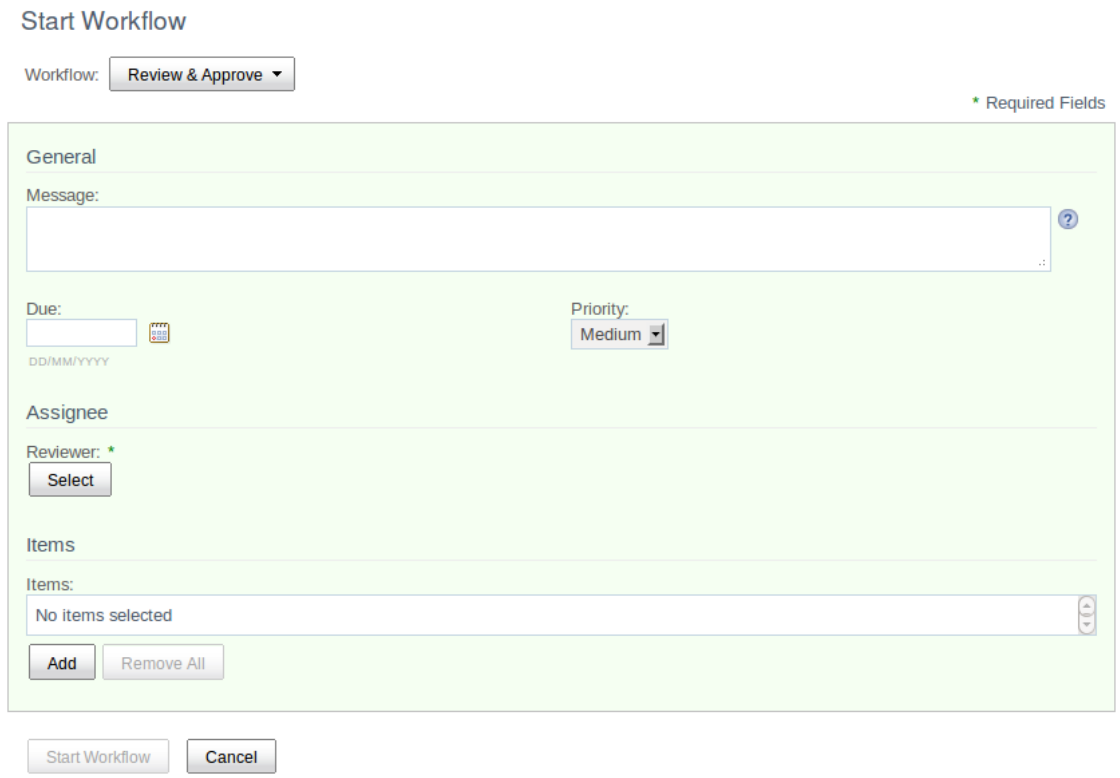
Toimenpidemalli (Task Model) kuvaa toimenpiteet työnkulussa. Jokainen kuvaus koostuu seuraavista elementeistä:

- Nimi ja titteli
- Ominaisuudet sekä assosiaatiot

Kuvausta käytetään käyttöliittymä-dialogeissa toimenpiteiden katselemisessa sekä hallinnoimisessa. Käytännössä toimenpidemalli on samanlainen kuin mikä tahansa muu malli Alfrescossa.

Webb-käyttöliittymän konfigurointi on omanlaisensa konfiguraatiotiedosto, jossa voidaan määritellä:

- Mitkä toimenpiteen ominaisuudet näytetään
- Millä ominaisuuksilla on pelkästään luku-oikeudet ja mitkä on vaadittu
- Kuinka ominaisuudet piirretään selaimessa
- Kuinka toimenpiteet keskustelevat käyttäjän kanssa



Kuva 11. Työnkulun aloitustoimenpide

6.12. Prosessimääritelmä

Alfrescossa työnkulku kulkee eteenpäin niin kutsuttuja uimalinjoja pitkin. Uimalinjoja voi sisältyä prosessiin useita ja niitä voi käyttää useampi toimenpide. Työnkulku voi siirtyä uimalinjalta toiselle ja se voi palata samalle uimalinjalle, mutta se voi kulkea vain yhtä uimalinjaa pitkin kerrallaan. Jokaiselle uimalinjalle määritellään roolit, jotka viittaavat käyttäjiin tai ryhmiin.

Uimalinjojen lisäksi määritellään toimenpiteet, joista jokaiseen määritellään mitä uimalinjaa ne käyttävät ja mille toimenpiteelle työnkulku seuraavaksi siirtyy. On olemassa kaksi erityistä toimenpidettä: aloitustoimenpide sekä lopetustoimenpide. Nämä toimenpiteet kertovat mistä toimenpiteestä työnkulku alkaa ja mihin se päättyy. Yksinkertainen esimerkki työnkulusta:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.1" name="wf:adhoc">
   <swimlane name="initiator"/>
```

```
   <start-state name="start">
      <task name="wf:submitAdhocTask" swimlane="initiator"/>
      <transition name="" to="adhoc"/>
   </start-state>

   <swimlane name="assignee"/>

   <task-node name="adhoc">
      <task name="wf:adhocTask" swimlane="assignee"/>
      <transition name="" to="completed"/>
   </task-node>

   <task-node name="completed">
      <task name="wf:completedAdhocTask" swimlane="initiator"/>
      <transition name="" to="end"/>
   </task-node>

   <end-state name="end"/>

</process-definition>
```

- **process-defintion** – aloittaa prosessimääritelmän ja nimeää sen
- **swimlane** – määritellään uimalinja ja annetaan sille nimi
- **start-state, task-node ja end-state** – toimenpiteet, joissa start- ja end-state ovat erityisiä aloitus- sekä lopetustoimenpiteitä
    - **task name** – annetaan toimenpiteelle nimi ja määritellään mitä uimalinjaa se käyttää
    - **transition** – määrittelee mihin toimenpiteeseen työnkulku seuraavaksi siirtyy

6.13. Prosessin käyttöönottaminen

Prosessi voidaan ottaa käyttöön rekisteröimällä se ”workflowDeployer” spring beanissä.

```
<bean id="myworkflows.workflowBootstrap" parent="workflowDeployer">
   <property name="workflowDefinitions">
      <list>
         <props>
            <prop key="engineId">jbpm</prop>
            <prop
key="location">alfresco/extension/workflow/custom_adhoc_processdefinit
ion.xml</prop>
            <prop key="mimetype">text/xml</prop>
         </props>
      </list>
   </property>
</bean>
```

- **location** – prosessimääritelmä-tiedoston sijainti. Niin kuin muutkin laajennukset, myös nämä sijoitetaan extension-hakemiston alle.
- **mimetype** – tiedoston mimetyyppi, joko text/xml tai sitten application/zip
- **redeploy** – true/false, jos määritelty true, uusi versio prosessista luodaan joka kerta, kun Alfresco käynnistetään. Voidaan käyttää kehittämisvaiheessa, mutta normaalisti määrätty falseksi.

6.13.1. Toimenpidemalli

Toimenpidemalli on samanlainen malli kuin mikä tahansa muu malli Alfrescossa. Mallissa kuvataan ominaisuudet, jotka kuuluvat toimenpiteisiin. Jokainen toimenpide on määritelty mallissa omina tyyppeinään. Myös toimenpidemallissa voidaan tuoda muita malleja, periä tyyppejä sekä käyttää piirteitä. Esimerkki aloitustoimenpiteen tyypistä viitaten esimerkkiprosessimallin edellä:

```
<type name="wf:submitAdhocTask">
  <parent>bpm:startTask</parent>

  <properties>
     <property name="wf:notifyMe">
        <type>d:boolean</type>
        <default>false</default>
     </property>
   </properties>

   <mandatory-aspects>
      <aspect>bpm:assignee</aspect>
   </mandatory-aspects>
</type>
```

Toimenpidemalli täytyy myös ottaa käyttöön käyttäen – context tiedostoa. Tässä käytetään Workflow Deployer beaniä.

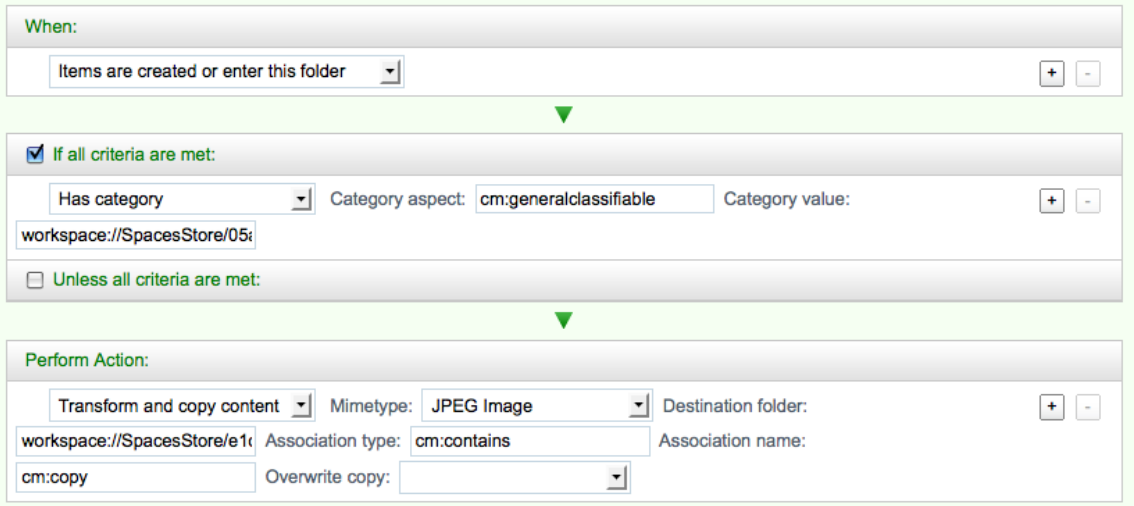
6.14. Toiminnot

Alfresco tarjoaa valmiita toimintoja, joita voi liittää tiedostopolkuun suoraan käyttöliittymästä. Näitä toimintoja ovat sellaiset, kuten ”lisää piirre”, ”vaihda mimetyyppi” jne. Käyttäjä voi päättää milloin toiminnot tulevat voimaan. Toiminnoille voidaan antaa myös kriteerit milloin ne tulevat voimaan, joka on mallia ”jos metatietokenttä x sisältää arvon y”. Toiminnoilla voidaan myös ajaa JavaScriptejä, jolloin voidaan suorittaa monimutkaisempia toimintoja.

6.15. Kustomoidut toiminnot

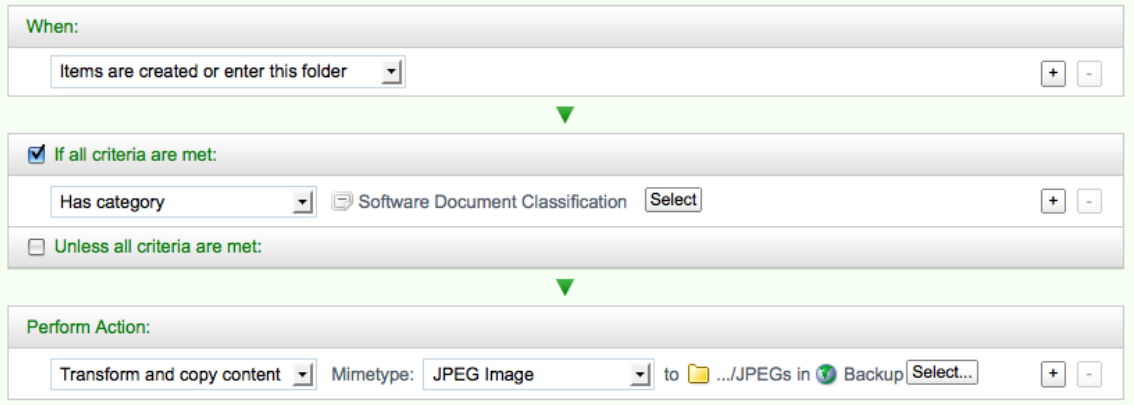
Kustomoiduilla toiminnoilla voidaan luoda kokonaan uudenlaisia toimintoja. Periaatteessa tämä on sama asia kuin JavaScriptin ajaminen, mutta nämä toiminnot voidaan nimetä ja lisätä Alfrescon omien toimintojen joukkoon jolloin niiden käyttäminen on huomattavasti helpompaa.

Kustomoiduille toiminnoille voidaan myös luoda omia UI-elementtejä. Omat UI-elementit voivat helpottaa huomattavasti toimintojen käyttämistä. Esimerkki toiminnosta, johon ei ole tehty UI-elementtejä:



Kuva 12. Säännön määritteleminen ilman kustomoitua toimintoa

Ja esimerkki toiminnosta, johon on tehty UI-elementti:



Kuva 13. Säännön määrittäminen kustomoidulla toiminnolla

Tämä toiminto tässä tapauksessa muuttaisi tiedoston JPEG-muotoon ja siirtäisi sen kansioon JPEG.

6.16. Ajoitetut toiminnot

Ajoitetuilla toiminnoilla tarkoitetaan toimintoja, jotka tulevat voimaan ennalta määritetyin aika-ajoin. Näitä aikoja voi olla, vaikka joka viides minuutti tai joka maanantai tai sitten vaikka, joka päivä klo: 12:00. Myös ajoitetulla toiminnolla voidaan ajaa omia JavaScriptejä.

Ajoitetut toiminnot ovat scheduled-action-services-context-tiedostosa, joka sijoitetaan tiedostopolkuun tomcat/alfresco/shared/alfersco/classes/extension. Ajoitettu toiminto koostuu kolmesta osasta:

- Cron-lauseke
- Kyselypohja
- Toimintopohja

Cron-lauseke määrää, milloin toiminto ajetaan. Kyselypohja määrittelee, mikä noodi on toiminnon kohde ja toimintopohja määrittelee toiminnon.

Esimerkki toiminnosta, joka ajaa JavaScriptin minuutin välein:

```
  <!--
    Execute the script /Company Home/Record Management/testscript.js
    -->
<!—toimintopohja →
  <bean                                          id="runScriptAction"
class="org.alfresco.repo.action.scheduled.SimpleTemplateActionDefiniti
on">
        <property name="actionName">
            <value>script</value>
        </property>
        <property name="parameterTemplates">
            <map>
                <entry>
                    <key>
                        <value>script-ref</value>
                    </key>

<value>${selectSingleNode('workspace://SpacesStore',          'lucene',
'PATH:"/app:company_home/cm:Record_x0020_Management/cm:testscript.js"'
)}</value>
                </entry>
```

```
            </map>
        </property>
        <property name="templateActionModelFactory">
            <ref bean="templateActionModelFactory"/>
        </property>
        <property name="dictionaryService">
            <ref bean="DictionaryService"/>
        </property>
        <property name="actionService">
            <ref bean="ActionService"/>
        </property>
        <property name="templateService">
            <ref bean="TemplateService"/>
        </property>
    </bean>

    <!--
   Run the script every minute - select the single node company home
that is not used ...
    -->
    <bean                                                    id="runScript"
class="org.alfresco.repo.action.scheduled.CronScheduledQueryBasedTempl
ateActionDefinition">
        <property name="transactionMode">
            <value>UNTIL_FIRST_FAILURE</value>
        </property>
        <property name="compensatingActionMode">
            <value>IGNORE</value>
        </property>
        <property name="searchService">
            <ref bean="SearchService"/>
        </property>
        <property name="templateService">
            <ref bean="TemplateService"/>
        </property>
        <property name="queryLanguage">
            <value>lucene</value>
        </property>
        <property name="stores">
            <list>
                <value>workspace://SpacesStore</value>
            </list>
        </property>
        <!—kyselpohja →
        <property name="queryTemplate">
            <value>PATH:"/app:company_home"</value>
        </property>
        <!— cron-lauseke →
        <property name="cronExpression">
            <value>0 0/1 * * * ?</value>
        </property>
        <property name="jobName">
            <value>jobD</value>
        </property>
        <property name="jobGroup">
            <value>jobGroup</value>
        </property>
        <property name="triggerName">
            <value>triggerD</value>
        </property>
        <property name="triggerGroup">
            <value>triggerGroup</value>
```

```
        </property>
        <property name="scheduler">
            <ref bean="schedulerFactory"/>
        </property>
        <property name="actionService">
            <ref bean="ActionService"/>
        </property>
        <property name="templateActionModelFactory">
            <ref bean="templateActionModelFactory"/>
        </property>
        <property name="templateActionDefinition">
            <ref bean="runScriptAction"/> <!-- This is name of the
action (bean) that gets run -->
        </property>
        <property name="transactionService">
            <ref bean="TransactionService"/>
        </property>
        <property name="runAsUser">
            <value>System</value>
        </property>
    </bean>
```

7. Alfrescon vastaavuus eAMS:n vaatimuksiin

Tarkastellessa eAMS:n vaatimuksia, voidaan huomata siitä löytyvän hyvin paljon yhtäläisyyksiä ECM:n kanssa. eAMS vaatii sellaisia ominaisuuksia, kuten metatiedot, prosessien automatisoiminen, versiointi sekä elinkaaren hallinta, jotka kaikki löytyvät ECM:sta.

7.1. Alfrescon tarjoamat ratkaisut

Tarkastellaan eAMS:n vaatimuksia aihepiirrettäin ja pohditaan voiko Alfresco tarjota ratkaisun niiden toteuttamiseen.

Alfresco mahdollistaa metatietojen tallentamisen tietorakenteeseen asiakirjoille sekä käsittelyprosesseille ja siinä voidaan laatia useita sisältötyyppejä, jotka vastaavat asiakirjatyyppejä.

Alfrescossa pystytään automatisoimaan prosessia laatimalla työnkulkuja. Työnkuluille voidaan tallentaa myös metatietoja, joita voidaan käyttää käsittelyvaiheiden ohjaamiseen. Työnkuluilla ja toimenpiteillä on mahdollista toteuttaa käsittelyprosessit sekä käsittelyvaiheet.

Alfrescoon voidaan määritellä toimintoja, joita voidaan käyttää asiakirjan automaattiseen metatietojen päivittämiseen. Toimintoja voidaan käyttää asiakirjan laatimisen yhteydessä tai käsittelyvaiheissa. Ajastetuilla toiminnoilla voidaan suorittaa sellaisia vaatimuksia, kuten asiakirjan poistaminen sen määräajan umpeuduttua.

Alfresco mahdollistaa käyttäjäryhmien sekä käyttäjien oikeuksien määrittelemisen. Oikeuksilla voidaan salata tietoa ulkopuolisilta tai estämään niiden muokkaus.

7.2. Havainnot

Havaintojen perusteella voidaan todeta, että Alfresco pystyy toteuttamaan eAMS:n vaatimukset. Kuitenkaan Alfresco ei pysty vastaamaan näihin haasteisiin ilman laajennusta, sillä monet eAMS:n vaatimukset eivät toteudu suoraan Alfrescossa. Näin ei voida olettaakaan tapahtuvan, sillä organisaatioilla on aina eri näkemykset sisällönhallinnan asettamista vaatimuksista. Alfrescon vahva puoli on kuitenkin nimenomaan sen laajennettavuus, joka on tehty suhteellisen helpoksi ja tehokkaaksi.

8. Testiympäristö

EAMS:n vaatimuksia toteutettiin testiympäristössä. Testiympäristö toimii demomallina, josta voidaan nähdä joitain ominaisuuksia toteutettuna siihen pisteeseen asti, että voidaan nähdä ominaisuuden olevan toteutuskelpoinen. Ominaisuudet valittiin sillä perusteella, jotta ne kattaisivat mahdollisimman suuren osan eAMS:n vaatimuksista.

Työympäristönä toimi Linux-käyttöjärjestelmä, Ubuntu ja työkaluina käytettiin mm. Eclipseä, terminaalia sekä Gedit-tekstinkäsittelyohjelmaa.

Testiympäristö rakennettiin Alfrescon Document Library komponenttia laajentamalla. Kaikki laajentavat tiedostot sijoitettiin tiedostopolkuun tomcat/shared/alfresco/.

8.1. Ominaisuudet, jotka toteutettiin testiympäristössä

Tehtiin testisivusto, jonne laadittiin EAMS:n mukainen tehtäväluokitus. Tehtäväluokituksen kolmannen eli alimman tason kansioille laadittiin oma sisältötyyppi, eams-Kansio, jolle annettiin niitä metatietoja, joiden on tarkoitus periytyä tehtäväluokkaan laadituille asiakirjoille.

Asiakirjalle laadittiin SÄHKE2:n pakolliset metatiedot laatimalla uusi sisältötyyppi, Asiakirja, jolle määriteltiin SÄHKE2:n vaatimat metatiedot. Document Libraryyn laadittiin toiminto, jolla voidaan tuoda tiedostoja, jotka ovat sisältötyypiltään Asiakirjoja ja saavat täten vaaditut metatiedot. Jotta metatiedot saatiin perittyä tehtäväluokasta asiakirjalle, tehtiin toiminto, joka käyttää JavaScriptiä metatietojen päivittämiseen. Toiminnolle annettiin ehto toteutua aina, kun tiedosto tuodaan tietojärjestelmään.

Tehtiin tehtäväluokkaan ajoitettu toiminto, joka etsii kaikki tiedostot, joille on määritelty hävittämisajankohta. Toiminto tarkistaa ajoittain onko aikamäärä umpeutunut ja poistaa tiedoston mikäli niin on tapahtunut.

Päivitettäessä asiakirjan metatietoa Julkisuusluokka, tarkistetaan arvo, jonka mukaan riippuen siitä saako kenttä arvon julkinen, osittain salassa pidettävä vai salainen, päivitetään tiedoston oikeudet.

Tehtiin testikäsittelyprosessi, joka pohjautuu JHS-suositukseen [LIITE]. Käsittelyprosessissa valitaan asiakas, joka saa tiedon käsittelyprosessin etenemisestä sähköpostilla. Toiminto toteutettiin JavaScriptiä käyttämällä. Käsittelyprosessissa asiakas voi hakea sähköpostilla oikaisupyyntöä, jonka vuoksi käsittelyprosessissa voidaan palata käsittelyvaiheissa takaisinpäin.

8.2. Metatietojen muuttaminen SÄHKE2:n vaatimusten mukaisiksi

SÄHKE2 asettamiin vaatimuksiin tietojärjestelmältä kuuluu pakolliset asiakirjojen metatiedot. Metatietoja varten laadittiin sisältömalli, joka sijoitettiin tiedostopolkuun alfresco/extension/model ja mallille annettiin nimi eams-model.xml.

```
<model                                                    name="eams:eams-model"
xmlns="http://www.alfresco.org/model/dictionary/1.0">

   <!—Valinnainen metadata mallista -->
   <description>eams Model</description>
   <author>Henri Lämsä</author>
   <version>1.0</version>

   <!-- Tuodaan Alfrescon ydinmallit Dictionary sekä Content, joista
jälkimmäinen toimii pohjana kaikelle sisällölle Alfrescossa -->
   <imports>
      <!—Tuodaan Alfresco Dictionary-malli -->
      <import        uri="http://www.alfresco.org/model/dictionary/1.0"
prefix="d"/>
      <!—Tuodaan Alfrescon Content-malli -->
      <import           uri="http://www.alfresco.org/model/content/1.0"
prefix="cm"/>
   </imports>

   <!—Määritellään mallin nimitila, joiden avulla voidaan käyttää
mallia muissa malleissa tuomalla ne -->
   <namespaces>
      <namespace            uri="http://www.eams.com/model/content/1.0"
prefix="eams" />
   </namespaces>
```

Seuraavaksi mallien ominaisuuksille määriteltiin rajoitukset. Sähke2-mallissa määritellään joillekin metatiedoille arvot, jotka ne voivat saada. Seuraavana esimerkki valintalistasta, jossa määritellään asiakirjan kieli.

```
<constraints>

        <constraint name="eams:languageList" type="LIST">
            <parameter name="allowedValues">
                <list>
                    <value>fi</value>
                    <value>sv</value>
                    <value>en</value>
                </list>
            </parameter>
         </constraint>
         …
</constraints>
```

Rajoituksien jälkeen määriteltiin asiakirjalle oma tehtäväluokan kansiolle sisältötyyppi, eams-kansio, joka peritään Alfrescon sisältömallin kansiosta. Eams-kansiolle määriteltiin pakolliset piirteet restrictions sekä retention eli rajoitukset ja säilytys. Nämä piirteet pitävät sisällään ne metatiedot, jotka periytyvät asiakirjoille.

```
<types>
```

```
      <type name="eams:folder">
         <parent>cm:folder</parent>
         <mandatory-aspects>
            <aspect>eams:restriction</aspect>
            <aspect>eams:retention</aspect>
         </mandatory-aspects>
      </type>
      …
```

Seuraavaksi toteutettiin sisältötyyppi asiakirjoille. Asiakirja peritään sisältötyypistä cm:content, joka toimii pohjana kaikelle sisällölle Alfrescossa. Cm:content sisältötyyppi pitää sisällään ominaisuuksia, kuten nimi, kuvaus, laatimispäivämäärä jne.

```
<type name="eams:doc">
   <title>Asiakirja</title>
      <parent>cm:content</parent>
         <properties>
            <property name="eams:language">
               <title>Kieli</title>
               <type>d:text</type>
               <mandatory>false</mandatory>
               <default>fi</default>
               <constraints>
                  <constraint ref="eams:languageList" />
               </constraints>
             </property>
              …
          <mandatory-aspects>
            <aspect>eams:restriction</aspect>
            <aspect>eams:retention</aspect>
         </mandatory-aspec
      </type>
```

Tyyppien jälkeen määriteltiin piirteet ”rajoitukset” sekä ”säilytys”. Näistä molemmat pitävät sisällään liudan piirteitä, jotka liittyvät aihealueeseensa. Myös SÄHKE2:n metatietomallissa nämä metatiedot esiteltiin omina ryhminään. Näiden määrittelemiseen piirteisiin oli myös konkreettinen syy, sillä näitä metatietoja käytetään myös eams-kansiossa sekä käsittelyprosessissa. Niinpä, kun ne määritellään piirteinä, voidaan ne liittää myös muihin tyyppeihin ja vältytään turhalta toistolta.

```
<aspects>
      <aspect name="eams:restriction">
         <properties>
         <property name="eams:restrictionPublicityClass">
            <title>Julkisuusluokka</title>
            <type>d:text</type>
            <mandatory>false</mandatory>
            <constraints>
               <constraint ref="eams:restrictionPublicityClassList" />
            </constraints>
         </property>
```

```
        …
</aspect>
```

Kun malli oli laadittu, oli vuorossa määritellä ominaisuuksille käyttäjäystävällisemmät merkintätavat eli ominaisuuksien nimeäminen. Tätä varten laadittiin eams-model.properties- tiedosto, joka sijoitettiin samaan tiedostopolkuun kuin aisaparinsa, eams-sisältömalli eli eams-model. Eams-model.properties näyttää kutakuinkin tältä:

```
eams-model.aspect.eams_restriction.description=Käyttörajoitus
eams-model.aspect.eams_restriction.title=Käyttörajoitus
eams-model.aspect.eams_retention.description=Säilytys
eams-model.aspect.eams_retention.title=Säilytys
eams-model.description=Alfresco eams
eams-model.property.eams_function.description=Tehtäväluokituksen
mukainen tehtävä, johon kohde on liitetty
eams-model.property.eams_function.title=Tehtävä
eams-model.property.eams_language.description=Kuvailtavan       kohteen
kieli
eams-model.property.eams_language.title=Tehtävä
…
```

Viimeisenä vaiheena piti malli saada rekisteröityä niin, että Alfresco käynnistyessään lukee mallin ja osaa käyttää sitä. Samaisessa tiedostossa rekisteröidään mallin käyttämä properties-tiedosto. Laadittiin mallille context-tiedosto, eams-model-context.

```
<!—Uusien mallien rekisteröiminen -->
    <bean                          id="eamsModel.dictionaryBootstrap"
parent="dictionaryModelBootstrap" depends-on="dictionaryBootstrap">
        <property name="models">
            <list>
                <value>alfresco/extension/model/eams-model.xml</value>
            </list>
        </property>
        <property name="labels">
            <list>
                <value>alfresco/extension/model/eams</value>
            </list>
        </property>
    </bean>
```

Nyt malli oli saatu rekisteröityä, mutta sen käyttäminen oli vielä mahdotonta, sillä Document Library ei anna käyttäjän valita tiedostolle, sen tuontivaiheessa, sisältötyyppiä. Jotta ongelma saatiin korjattua, täytyi tiedoston tuomisesta järjestelmään huolehtivaan komponenttiin tehdä muutoksia. Tämä toiminto saatiin suoritettu muokkaamalla yhtä tiedostoa. Kyseessä on flash-upload.get.js-tiedosto, joka on osa Alfrescon web-scriptiä. Tämän web-scriptin tarkoitus on yksinomaan mahdollistaa tiedoston tuominen Document Library:iin.

Jotta vältyttiin koskemasta Alfrescon ydintiedostoihin, laajennettiin tiedoston tuomis -web scriptiä ylikirjoittamalla flash-upload.get.js:

```
/**
 * Custom content types
 */
function getContentTypes()
{
   // TODO: Data webscript call to return list of available types
   var contentTypes = [
   {
      id : "eams:doc",
      value : "eams_doc"
   },
   {
      id : "cm:content",
      value : "cm_content"
   }
   ];

   return contentTypes;
}

model.contentTypes = getContentTypes();
```

Muokkausta ei tarvinnut tehdä kovin paljoa, sillä Alfrescon valmis web-script tukee sisältötyypin vaihtamista, mutta koska Alfrescon Document Library sisältää vain yhden sisältötyypin, ei lomakkeessa ole tarvetta näkyä valintalistaa, jolla valita sisältötyyppi. Niinpä, kun listaan lisättiin tiedostotyyppi, web script osasi laatia valintalistan lomakkeeseen automaattisesti. On hyvin todennäköistä, että tämä ominaisuus on tulossa lähiaikoina Alfrescoon.

Tiedoston tuonti-lomake käännettiin vielä suomen kielelle ylikirjoittamalla web-scriptin properties-tiedosto, flash-upload.get.properties. Nyt sisältötyypin valitseminen oli mahdollista tiedostoja tuodessa järjestelmään. Metatiedotkin näkyivät jo asiakirjoja tarkastellessa, mutta ne olivat kaikki vinksallaan. Tämä siksi, että uudelle sisältötyypille ei ole määritelty konfiguraatiota.

Sharen kaikki laajentavat konfiguraatiot tulevat share-config-custom-tiedostoon, joka sisällytetään web-extension-kansioon. Tiedostoon tehtiin seuraavanlaiset määritelmät, jotka muokkaavat metatietokenttien näkymistä käyttöliittymässä:

```
<config evaluator="node-type" condition="eams:doc">
      <forms>
         <form>
            <field-visibility>
               <!—Peritään sisältötyypistä cm:content -->
               <show id="cm:name" />
```

```
                <show id="cm:title" force="true" />
                <show id="cm:description" force="true" />
                …
                <!--  eams Asiakirjaominaisuudet -->
                <show id="eams:language" />
                <show id="eams:function" for-mode="view" />
                <show id="eams:status" for-mode="view" />
                <show id="eams:type" for-mode="view" />

                <!--  eams rajoitukset-piirre -->
                <show id="eams:restrictionPublicityClass" />
                <show    id="eams:restrictionSecurityPeriodEnd"    for-
mode="view" />
                <show id="eams:restrictionSecurityReason" />
                ...
            </field-visibility>
            ...
```

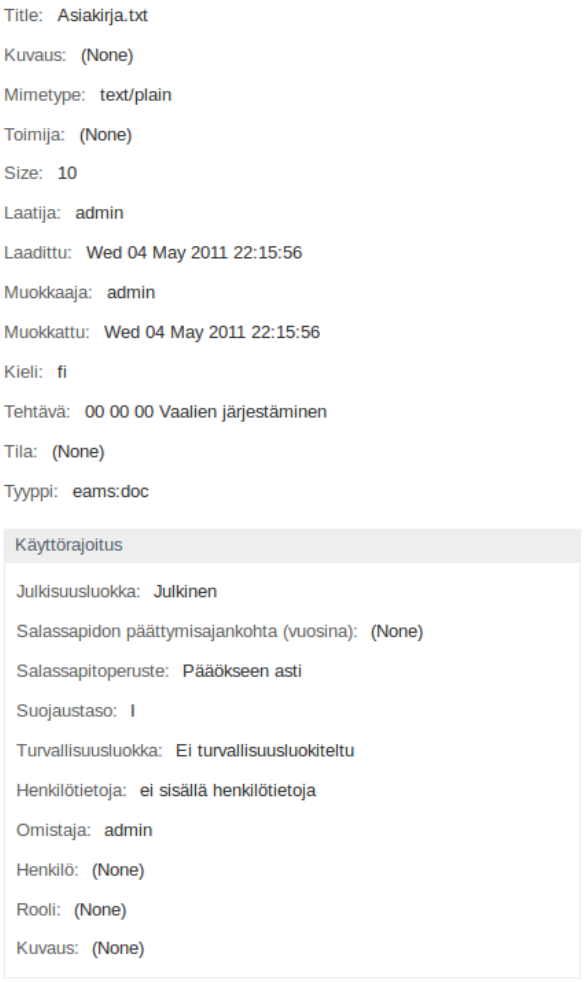
Tagissä field-visibility määritellään ne kentät, jotka näkyvät lomakkeissa. ”For-mode”-määritteellä voidaan määrittää milloin metatietokenttä näytetään. ”View” tarkoittaa sitä, että metatietokenttä näytetään vain silloin, kun tarkastellaan metatietoja ja ”edit” silloin, kun metatietoja muokataan. Metatiedot function, status ja type saavat arvon tietojärjestelmältä eikä niitä saa muokata. Jättämällä ne pois muokkaus-näkymästä estetään käyttäjiä muokkaamasta niitä.

```
</field-visibility>
            <appearance>
                <field id="eams:language" />
                    <control template="controls/selectone.ftl">
                        <control-param   name="options">fi,sv,en</control-
param>
                    </control>
                <field id="eams:function" />
                …
                <set    id="restriction"    appearance="bordered-panel"
label="Käyttörajoitus" />
                <set     id="retention"     appearance="bordered-panel"
label="Säilytys" />

                <field              id="eams:restrictionPublicityClass"
set="restriction" />
                    <control template="controls/selectone.ftl">
                        <control-param    name="options">Julkinen,Osittain
salassa pidettävä,Salassa pidettävä</control-param>
                    </control>
```

Appearance-tagissä sijaitsevat määritykset siitä kuinka kentät näkyvät lomakkeissa. Konfiguraatioissa voidaan käyttää ftl-pohjia, joihin voi määritellä näkymää käyttämällä mm. html-kieltä. Konfiugraatiossa käyttörajoitus- sekä säilytyspiirteiden ominaisuuksille tehtiin ryhmät eli sets, joiden avulla voidaan esim. piirtää reunat kenttien ympärille erottamaan ne muista metatietokentistä.

Eams-kansiolle tehtiin samantapainen lomake-konfiguraatio ja sen lisäksi määriteltiin eams-kansio Alfrescon oman kansion eli cm:folder:in alasisältötyypiksi. Tämä mahdollistaa Alfrescon kansioiden sisältötyypin muuttamisen eams-kansioksi suoraan käyttöliittymästä.



Kuva 14 eAMS:n metatiedot

8.3. Metatietojen periytyminen asiakirjoille tehtäväluokasta

Periytymistä varten laadittiin toiminto, joka lukee asiakirjan ”vanhemman” eli sen kansion (tehtäväluokan) metatiedot missä asiakirja on. Tämä tehtiin käyttämällä Alfrescon sääntö-toimintoa, joka ajaa JavaScriptin aina, kun asiakirja tuodaan tehtäväluokkaan. Samassa toiminnossa päivitetään myös metatiedon Tyyppi

arvo, joka on nimenomaan asiakirjan sisältötyyppi. Lisäksi laatija määritellään rajoituksiin omistajaksi sekä hallinnoijaksi.

JavaScript liitettiin Alfrescoon sijoittamalla se Alfrscon erilliseen scripteille varattuun kansioon, josta sitä voidaan käyttää säännöissä suoraan käyttöliittymästä.

```
document.properties["eams:function"] = document.parent.parent.name;
document.properties["eams:type"] = document.typeShort;
document.properties["eams:restrictionOwner"]                              =
person.properties.userName;
document.properties["eams:restrictionAccessRightName"]                   =
person.properties.userName;

if   (document.parent.properties["eams:restrictionPublicityClass"]   !=
null)
   document.properties["eams:restrictionPublicityClass"]                  =
document.parent.properties["eams:restrictionPublicityClass"];
if   (document.parent.properties["eams:restrictionSecurityReason"]   !=
null)
   document.properties["eams:restrictionSecurityReason"]                  =
document.parent.properties["eams:restrictionSecurityReason"];
if   (document.parent.properties["eams:restrictionProtectionLevel"]   !=
null)
   document.properties["eams:restrictionProtectionLevel"]                 =
document.parent.properties["eams:restrictionProtectionLevel"];
if    (document.parent.properties["eams:restrictionSecurityClass"]    !=
null)
   document.properties["eams:restrictionSecurityClass"]                   =
document.parent.properties["eams:restrictionSecurityClass"];
if    (document.parent.properties["eams:restrictionPersonalData"]    !=
null)
   document.properties["eams:restrictionPersonalData"]                    =
document.parent.properties["eams:restrictionPersonalData"];
if (document.parent.properties["eams:retentionPeriod"] != null)
   document.properties["eams:retentionPeriod"]                            =
document.parent.properties["eams:retentionPeriod"];
if (document.parent.properties["eams:retentionReason"] != null)
   document.properties["eams:retentionReason"]                            =
document.parent.properties["eams:retentionReason"];

document.save();
```

Scriptissä document viittaa asiakirjaan, joka tuodaan järjestelmään ja parent sitä kansiota missä asiakirja sijaitsee. Properties[”property”] viittaa asiakirjan ominaisuuksiin eli metatietoihin. Koska ei voida olettaa, että kaikki tehtäväluokan metatiedot omaisivat arvon, tehtiin if-lauseella tarkistus, mikä tarkistaa onko metatietoon syötetty arvoa. Lopuksi komennolla document.save() tallennetaan asiakirja.

Document Libraryssä tehtiin sääntö käyttämällä ”Manage Rules”-käyttöliittymää ja annettiin sille ehto ajaa JavaScript aina, kun asiakirja tuodaan tehtäväluokkaan.

8.4. Elinkaari

Kun asiakirja saapuu käsittelyprosessissa viimeiseen käsittelyvaiheeseen ”valmis”, asiakirjalle annetaan metatiedot sen hävittämistä koskien. Tehtiin toiminto, joka laskee tehtäväluokan säilytysajan perusteella päivämäärän jolloin asiakirja poistuu. Päivämäärä tallennetaan yhdessä säilytysajan kanssa asiakirjan metatietoihin.

```
var log ="";
var                                logFile                                =
companyhome.childByNamePath("retentionPeriodEnd_log.txt");
var removeDate = new Date();

if (logFile == null)
{
logFile = companyhome.createFile("retentionPeriodEnd_log.txt");
}

removeDate.setFullYear(removeDate.getFullYear()                           +
document.parent.parent.properties["eams:retentionPeriod"]);

document.properties["eams:retentionPeriodEnd"] = removeDate;

log   +=   document.properties["eams:retentionPeriodEnd"]   +   "\n"   +
"removeDate: " + removeDate + "\n";

document.save();

logFile.content += log;
```

8.5. Ajoitetut toiminnot: automaattinen hävitys

Ajoitetut määritellään scheduled-actions-services-context.xml-tiedostossa, joka voidaan sijoittaa polkuun classes/alfresco/extension.

```
…
<property name="parameterTemplates">
            <map>
                <entry>
                    <key>
                        <value>script-ref</value>
                    </key>
<value>${selectSingleNode('workspace://SpacesStore',          'lucene',
'PATH:"/app:company_home/app:dictionary/cm:scripts/cm:scheduled-
delete.js"' )}</value>
                </entry>
            </map>
        </property>
        …
```

JavaScript määritellään runScriptAction beanissä parameterTemplates property – kohdassa. Scriptin hakemiseen käytetään Lucene-search – toimintoa.

Querytemplatella määritellään ne noodit mitä ajoitettu toiminto koskee. Tässä tapauksessa valitaan ne noodit mitkä omaavat piirteen retention eli säilytys. QueryTemplaten arvoksi annettiin sen sisältömallin nimitila, jossa piirre sijaitsi sekä piirre eams:retention.

```
<property name="queryTemplate">
            <value>PATH:"//\*"                                    ASPECT:"{
uri="http://www.eams.com/model/content/1.0"
prefix="eams"}eams:retention"</value>
        </property>
        <property name="cronExpression">
            <value>0 0/1 * * * ?</value>
        </property>
```

Ominaisuudella cronExpression määritellään se aikaväli milloin toiminto toteutetaan. Nyt toiminto määriteltiin suoritettavaksi joka minuutin välein siksi, että voitiin helposti testata toiminto. Valmiissa toteutuksessa ajaksi voitaisiin määritellä, vaikka tietty ajankohta päivästä, sillä tiedostojen poistamisaika on SÄHKE2:ssa määritelty päivämääräksi.

Toiminnolle laadittiin JavaScript, joka toteuttaa itse toiminnon.

```
var log = "";
var logFile = companyhome.childByNamePath("remove_log.txt");
var presentDate = new Date();
var removeDate;

if (logFile == null)
{
logFile = companyhome.createFile("remove_log.txt");
}

var                                aspect                                =
"PATH:\"/app:company_home/cm:company_home/cm:sites/cm:test/*\"+ASPECT:
\"{http://www.eams.com/model/content/1.0}retention\"";
var searchQuery = aspect;
var nodes = search.luceneSearch(searchQuery);

var node = null;
for each (node in nodes)
{
   if(node)
   {
      if (node.properties["eams:retentionPeriodEnd"])
      {
         if (node.properties["eams:retentionPeriodEnd"] != null)
         {
            removeDate = node.properties["eams:retentionPeriodEnd"];
            if(removeDate < presentDate)   {
```

```
                log += "File Name:" + node.properties.name + "\t" +
"Remove Date:" + removeDate + "\n";
                node.remove();
            }
         }
      }
   }
}

logFile.content += log;
```

Scriptissä kirjoitetaan logitiedosto, joka sijoitetaan tiedostopolussa rakenteen CompanyHome (eli se tiedostorakenteen pohjalle) juureen. Logitiedostoon kirjoitetaan tiedoston nimi ja päivämäärä asiakirjan poistamisen yhteydessä. Scriptin alussa tallennetaan muuttujaan presentDate tämänhetkinen päivämäärä. Myös scriptissä käytetään Lucene-hakua etsimään ne noodit, jotka omaavat piirteen säilytys. Kun noodit on haettu, ne käydään läpi for each-silmukassa ja jokaisen kohdalla tarkistetaan metatieto retentionPeriodEnd eli Säilytysajan päättymisajankohta. Jos tämänhetkinen päivämäärä (presentDate) on suurempi kuin säilytysajan päättymisajankohta, asiakirja poistetaan.

8.6. Käyttäjä muuttaa asiakirjan julkisuusluokkaa

Asiakirja voi muuttua julkiseksi myös silloin, jos käyttäjä muuttaa julkisuusluokka-metatiedon arvoa. Näin ollen tietyt oikeudet pitää purkaa asiakirjasta ja päinvastoin silloin, jos metatieto saa arvon salainen. Vaatimus toteutettiin taas käyttämällä JavaScriptiä sekä sääntöjä.

Tehtiin sääntö, jossa toiminto toteutetaan silloin, kun asiakirjaa päivitetään. Tällä hetkellä testiympäristössä toiminto ajetaan aina, kun asiakirja päivitetään. Parempi vaihtoehto olisi tarkistaa, mikäli julkistamisajankohta on muuttunut ja ajaa vain siinä tapauksessa, jos niin on käynyt. Tällä hetkellä toiminto raskauttaa metatietojen päivittämistä turhan takia.

```
if      (document.properties["eams:restrictionPublicityClass"]      ==
"Julkinen")
   document.setPermission("Read");
else    if    (document.properties["eams:restrictionPublicityClass"] ==
"Osittain salassa pidettävä")
{
   document.removePermission("Read");
   document.setPermission("Read",
document.properties["eams:restrictionOwner"]);
   document.setPermission("Write",
document.properties["eams:restrictionOwner"]);
```

```
   document.setPermission("Delete",
document.properties["eams:restrictionOwner"]);
}
else
{
   document.removePermission("Read");
   document.setPermission("Read",
document.properties["eams:restrictionOwner"]);
   document.setPermission("Write",
document.properties["eams:restrictionOwner"]);
   document.setPermission("Delete",
document.properties["eams:restrictionOwner"]);
}
```

JavaScript tarkistaa minkä julkistamisluokan arvon asiakirja sai ja tekee toimintoja sen mukaan. Funktiolla setPermission voidaan antaa oikeuksia ja siinä määritellään kenelle oikeudet annetaan. Jos käyttäjää tai ryhmää ei määritellä, kaikki saavat oikeudet. removePermission poistaa oikeudet ja toimii samalla periaatteella kuin setPermission.

8.7. Testikäsittelyprosessi

Käsittelyprosessi toteutettiin käyttämällä Alfrescon yksinkertaista työnkulkua. Jokaisen tehtäväluokan hierarkian alimmaiselle tasolle laadittiin kansiot, jotka esittävät käsittelyvaiheita. Kun asiakirja tuodaan käsittelyprosessin ensimmäiseen käsittelyvaiheeseen, sille tallennetaan metatiedot. Käyttäjän valitsemalla ”Hyväksy” käyttämällä asiakirjan käyttöliittymän toimintoja, asiakirja siirtyy seuraavaan käsittelyvaiheeseen. Kun asiakirja saapuu uuteen käsittelyvaiheeseen, sen metatietoihin tallennetaan kohtaan tila uusi arvo, joka periytyy nykyisestä käsittelyvaiheesta. Asiakirjan saapuessa käsittelyvaiheeseen päätös annetaan käyttäjälle mahdollisuus hylätä asiakirjan eteneminen, jolloin se siirtyy takaisin käsittelyvaiheeseen. Viimein asiakirjan saapuessa viimeiseen käsittelyvaiheeseen valmis, liitetään siihen piirre, joka antaa metatiedot asiakirjan hävittämiselle.

Yksinkertaisen työnkulku tehtiin määrittelemällä säännöt työnkulun aloittamiselle jokaiseen käsittelyvaiheeseen. Säännöt määriteltiin käyttämällä käyttöliittymän sääntöjen hallinta-työkalua. Työnkulut määriteltiin alkamaan asiakirjan saapuessa käsittelyvaiheeseen.

Toiminto, joka antaa asiakirjalle metatietoarvot sen saapuessa ensimmäisen kerran tietojärjestelmään, määriteltiin luvussa 8.3. Piirteen asiakirjan hävittämisestä toteutettiin säännöllä, joka käyttää Alfrescon valmiita toimintoja ”Add Aspect” eli lisää piirre.

9. Tulokset

Valmis demoympäristö sisältää eAMS:n tehtäväluokituksen. Tietojärjestelmään on tallennettu eAMS:n pakolliset metatiedot. Asiakirjan metatiedot, kuten tyyppi, joka viittaa asiakirjan sisältötyyppiin sekä tehtävä, joka viittaa siihen tehtäväluokkaan, johon asiakirja kuuluu, päivittyvät automaattisesti asiakirjan saapuessa järjestelmään. Asiakirja poistetaan automaattisesti sen säilytysajan päättyessä. Kun asiakirjan julkisuusluokka muutetaan julkiseksi, sen rajoitukset puretaan niin, että kaikilla on oikeus lukea asiakirja. Laadittiin käsittelyprosessi, johon kuuluu käsittelyvaiheet. Kun asiakirja siirtyy käsittelyvaiheesta toiseen, sen tilametkatietoarvo päivitetään vastaamaan käsittelyvaihetta.

Demoympäristön pohjalta voidaan todeta, että Alfrescossa voidaan toteuttaa ne vaatimukset, jotka eAMS:n tietojärjestelmältä odotetaan. Alfresco tarjoaa ketterän sekä vapaan ympäristön, joka vaatii tietotaitoa, mutta antaa samalla mahdollisuudet suurten muutoksien sekä laajennuksien toteuttamiseen. Tämän mahdollistaa avoimen lähdekoodin standardien sekä komponenttien käyttäminen sekä Alfrescon arkkitehtuuri, joka on tehty nimenomaan laajennusta sekä muokkausta silmällä pitäen.

10. Pohdinta

Työstä voi olla monenlaista hyötyä riippuen siitä, minkälaista tietoa aiheesta etsitään. Pääasiassa työssä pohdittiin miten Alfresco sopii eAMS:n tietojärjestelmäksi, mutta sitä voidaan käyttää myös oppaana miten perehtymään Alfrescoon ja pääasiassa sen laajennettavuuteen. Työ tarjoaa hyvän pohjan miten Alfrescoa voidaan laajentaa riippumatta siitä, minkälaista laajennusta ollaan tekemässä. Lisäksi työ antaa tietoa yleisesti eAMS:sta sekä SÄHKE2-

standardista ja antaa esimerkin miten eAMS:n tietojärjestelmä voidaan toteuttaa. Tätä tietoa voidaan käyttää pohjana myös silloin, kun käytetään jotain toista tietojärjestelmää Alfrescon sijaan. Ammatillisesta näkökulmasta, työ tarjoaa yleiskuvan tämän päivän sisällönhallintamenetelmistä sekä toteutustavoista.

Tulokset on toteutettu aidossa ympäristössä, joten niiden voidaan katsoa olevan luotettavia. Kuitenkin täytyy huomioida, että Alfresco päivittyy koko ajan ja siksi tulokset voivat ajan mittaan vanheta. Työssä perehdyttiin hyvin vähän tietoturvaan, jota eAMS:n tietojärjestelmältä vaaditaan ja jota vaaditaan yleisestikin sisällönhallintajärjestelmältä. Tämä on iso kokonaisuus, joka vaatisi lisää tutkimusta. Tämä kokonaisuus jätettiin pois, sillä työ olisi voinut paisua liian suureksi.

Mielestäni lähtökohtiin nähden työ onnistui hyvin ja olen itse tyytyväinen projektiin. Opin paljon sisällönhallinasta, Alfrescon toimivuudesta sekä siitä miten laajentaa Alfrescoa. Samalla opin miten viedä eteenpäin näinkin mittavaa projektia.

Projekti oli hyvä valinta opinnäytetyön aiheeksi. Se oli haasteellinen ja vastaa mielestäni hyvin suuntautumistani.

Lähteet

Arkistolaitos. 2008. Sähköisten asiakirjallisten tietojen käsittely, hallinta ja säilyttäminen. http://www.arkisto.fi/uploads/normit/valtionhallinto/maarayksetjaohjeet/normiteksti_suomi.pdf. 13.4.2011

Arkistolaitos. Metatietomalli http://www.arkisto.fi/fi/saehke2-maeaeraeys/. 13.4.2011

Kansallisarkisto. Arkistonmuodostussuunnitelma AMS 2011 http://www.ams-opas.fi/sahkoinen-ams/tulostusversio/ 13.4.2011

Boiko, B. 2005. Content Management Bible 2nd Edition. Indianapolis, Indiana. Wiley Publishing, Inc.

Kansallisarkisto. Sähköisen AMS: n laadintaprosessi. http://www.ams-opas.fi/sahkoinen-ams/ams-rakenne/. 13.4.2011

Kampffmeyer, U. 2006. ECM Enterprise Content Management. Project Consult. http://www.project-consult.net/Files/ECM_White%20Paper_kff_2006.pdf. 8.4.2011.

Potts, J. 2008. Alfresco Developer Guide. Pact Publishing