Architecture of E-Health Flanders platform

March 22, 2010

Part I

Documentation Beyound Views

0.1 Documentation roadmap

0.1.1 Description of the parts

0.1.2 How stakeholders might use the package

Platform tester

Patient

Government

0.2 View template

0.3 System overview

0.4 Mapping between views

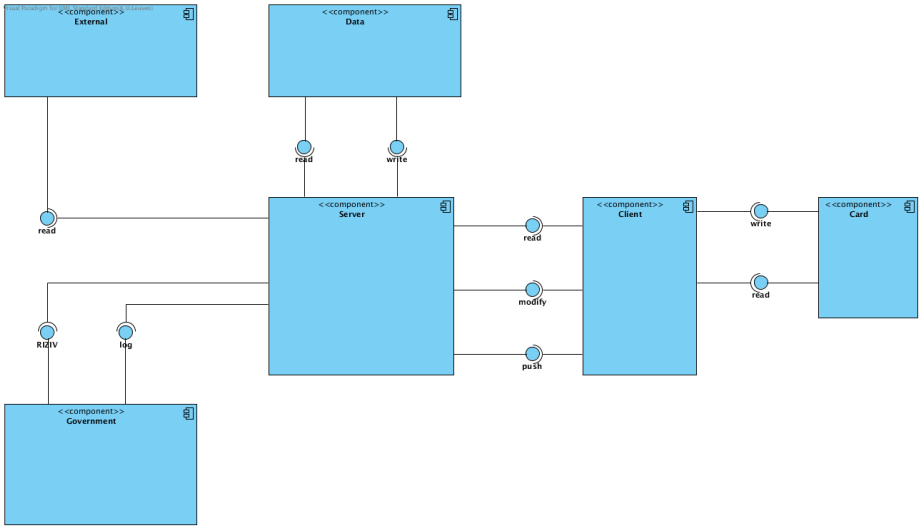
0.5 Directory

0.6 Glossary and acronym list

0.7 Background, design constraints, and rationale

Part II

Software Architecture Views



0.8 Module Uses View

0.8.1 Primary presentation

0.8.2 Element catalog

Elements and their properties

Relations and their properties

Element interfaces

Element behavior

0.8.3 Context diagram

0.8.4 Variability guide

0.8.5 Architecture background

Rationale

Analysis results

Assumptions

0.8.6 Other information

0.8.7 Related view packets

0.9 C&C Client and Server View: Overview

0.9.1 Primary presentation

0.9.2 Element catalog

Server

De server is de component waar clients verbinding mee maken. De interacties tussen clients en servers gebeuren op een veilige manier. De server is verbonden met clients, de overheid, eventuele externe componenten en de data server. Meer informatie kan gevonden worden in 0.10 Client and Server View: Server.

Client

De client component is de component die gebruikers (dokters, patiënten of apothekers) gebruiken om op een veilige manier met de server te interageren. Meer informatie kan gevonden worden in 0.11 Client and Server View: Client.

Data

De data component is een database waar alle informatie zoals patienten dossiers en dokter data op bewaard zijn. Hoe de data precies wordt opgeslaan is terug te vinden in het Deployment diagram.

Card

De card component stelt de e-Card van de gebruiker voor. De e-Card bevat gebruikersinformatie alsook twee keys. Een key voor identificatie van de gebruiker en een key voor authenticatie. Naast de gebruikersinformatie en de keys heeft de e-Card ook ruimte voor een aantal voorschriften op te slaan. Meer informatie is terug te vinden in:

Government

De government component bevat toegang tot de logging en RIZIV database.

External

De external component bevat de validated data sources die gebruikers van de client kunnen opvragen via de server.

0.9.3 Context diagram

TODO

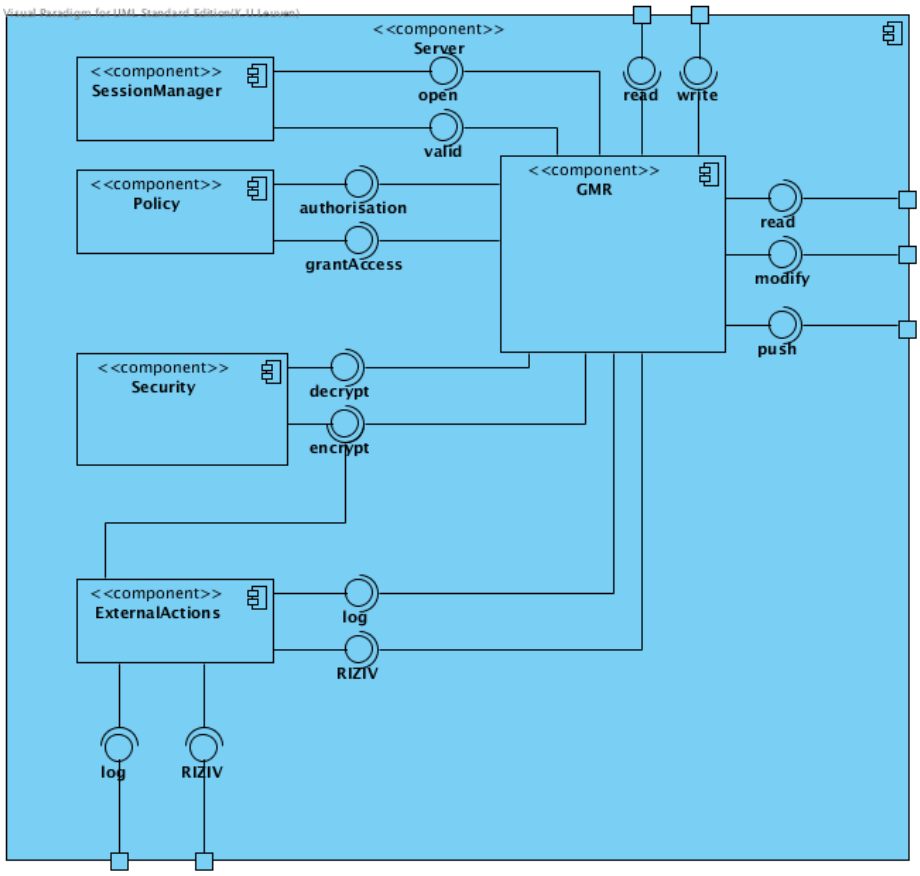
0.9.4 Variability guide

[None]

0.9.5 Architecture background

Rationale

Enkele design beslissingen die hier te zien zijn is het gebruik van de card reader voor het identificiëren van de gebruikers op de centrale database, meer hierover is te vinden in het ...TODO.



0.9.6 Related view packets

De volgende client-server views bekijken dit overview diagram in meer detail. Ook in het deployment diagram is extra informatie te vinden vooral dan in verband met de opslag van de global medical records en dokter data.

0.10 C&C Client and Server View: Server

0.10.1 Primary presentation

0.10.2 Element catalog

GMR

De GMR is de centrale component in de server component. De GMR delegeert alle inkomende *read*, *modify* of *push* acties naar de correcte andere componenten en doet dit uiteraard in de correcte volgorde.

Element interfaces

read

Resources provided:
read(data_key, id_key)

De id_key wordt geauthenticeerd en daarna wordt een sessie geopent. De session_key wordt samen met data die bij de data_key hoort teruggegeven.
read(data_key, session_key)

De data met data_key wordt gelezen en teruggeven aan het systeem.

Error handling:
Wanneer de data_key niet geldig is wordt een error teruggegeven.
Wanneer de id_key of session_key geen toegang heeft tot data_key wordt een error teruggegeven.
Wanneer de id_key of session_key niet geldig is wordt een error teruggegeven.

Element requirements:
data_key: de key om de data op te vragen.
id_key: de key die gebruikt wordt om de uitvoerder van de actie op te vragen.
session_key: de key die gebruikt wordt om een sessie te identificiëren.

Rationale and design issues:
Het openen van een sessie wanneer de eerste actie wordt uitgevoerd met een id_key kan ook aangepast worden om een expliciete login te doen. Dan zou de client eerst moeten inloggen met de id_key en dan kan hij later acties uitvoeren met de session_key.
Voor bepaalde acties kan men verwachten dat altijd een id_key wordt meegegeven.
Zie ook deployment voor het ophalen van data. Textuele data wordt altijd eerst teruggegeven, daarna pas andere data (media bestanden zoals figuren, geluid, video).

modify

Resources provided:
modify(data_key, data, id_key)
modify(data_key, data, session_key)
Het inloggen en valideren gebeurt zoals beschreven bij **read**. De aanpassing van de data verloopt door het mergen van data met de bestaande data. Wanneer de GMR een antwoord krijgt van de database dat alles correct is geschreven verstuurt de GMR een antwoord naar de client dat alle data correct is geschreven. Deze handelingen gebeuren asynchroon.

Error handling:
Gelijkaardige errors zoals bij **read**.
Wanneer het mergen van de data faalt zal een error teruggegeven worden.

Element requirements:
data_key: de key om de data op te vragen.
data: een patch van data die moet aangepast worden op locatie data_key.
id_key: de key die gebruikt wordt om de uitvoerder van de actie op te vragen.
session_key: de key die gebruikt wordt om een sessie te identificiëren.

Rationale and design issues:
Zie ook **read**.
Het schrijven van data gebeurt met behulp van een patch. Dit verkleint de hoeveelheid data die moet verstuurd worden aanzienelijk en heeft dan ook een belangrijk snelheids voordeel.
In het geval dat een merge actie zou falen, kan aan de laatste client gevraagd worden om de merge manueel te doen, of om een volledig document te verzenden.

push

Resources provided:
push(data_key, data, id_key)
push(data_key, data, session_key)
Het inloggen en valideren gebeurt zoals beschreven bij **read**. Het pushen van de data gebeurt gelijkaardig zoals bij **modify**.
Error handling:
Gelijkaardige errors zoals bij **read**.

Element requirements:
data_key: de key om de data op te vragen.
data: data die moet toegevoegd worden aan document met key: data_key.
id_key: de key die gebruikt wordt om de uitvoerder van de actie op te vragen.
session_key: de key die gebruikt wordt om een sessie te identificiëren.

Rationale and design issues:
Zie ook **read**.
Hier kan geen patch gestuurd worden omdat een push actie enkel toelaat nieuwe data toe te voegen aan een dossier.

Element behavior

De GMR component delegeert alle inkomende acties. Dat doet hij door inkomende berichten door de security component te laten decrypten. Dan krijgt hij onmiddellijk bevestiging dat de data compleet en correct is en van de een geauthenticeerde gebruiker komt. Indien dit niet het geval is stuurt de GMR een error terug naar de client.
Daarna wordt gecontroleerd of er al een sessie bestaat en of die al dan niet geldig is.
Wanneer dat gebeurd is zal het systeem de oproep doorgeven aan de policy component. Deze zal de GMR laten weten of de actie die de GMR binnen krijgt al dan niet toegelaten is.
Als de actie toegelaten is voor de gebruiker zal de GMR de actie uitvoeren en een gepaste respons sturen naar de client.
Als het nodig is worden acties gelogd of verzonden naar het RIZIV.

SessionManager

De SessionManager voorziet in session management. Deze component zorgt ervoor dat een gebruiker niet telkens opnieuw hoeft in te loggen wanneer hij een actie naar het systeem uitvoert.

Element interfaces

open

Resources provided:
open(id_key)
Opent een nieuwe sessie voor id_key, de session_key wordt teruggegeven.

Element requirements:
id_key: de key waarmee de client zich identificiëert.

Rationale and design issues:
Voorziet in het openen van sessies, waardoor de client niet telkens zijn identificatie key moet opnieuw ingeven.

valid

Resources provided:
valid(session_key)
Dit controleert of de session_key nog geldig is, een sessie verloopt na een bepaalde tijd. Er wordt teruggeven als de session_key nog al dan niet geldig is. Als ze nog geldig is wordt de id_key teruggegeven die verbonden is met deze session_key.

Element requirements:
session_key: de key die gebruikt wordt om een sessie te identificiëren.

Rationale and design issues:
Hoe lang een sessie geldig blijft zou kunnen aangepast worden afhankelijk van wie er in het systeem inlogt of afhankelijk van welke soort client. Wanneer de client toepassing op een smartphone draait zou er kunnen gekozen worden om de session_keys sneller te laten vervallen aangezien deze omgeving als minder veilig kan beschouwt worden. Indien we dit willen afdwingen moet wat extra informatie worden meegegeven bij het openen van de sessie.

Element behavior

De SessionManager is verantwoordelijk voor het openen van sessies (*open*) en het controleren of een bepaalde sessie geldig is (*valid*).

Policy

Deze component controleert de toegangsrechten van de gebruikers en laat weten welke gebruikers toegang hebben tot welke delen van de data en wat die toegang inhoudt (lezen en of schrijven).

Element interfaces

autorisation

Resources provided:
authorize(data_key, action, id_key)
De authorisatie component kijkt of de persoon met identificatie id_key toelating heeft om actie action uit te voeren op document met key data_key. Er wordt teruggegeven of de actie al dan niet is geauthoriseerd.

Element requirements:
data_key: de key verbonden met een data document.
action: een actie die uitgevoerd zal worden, dit kan read, modify of push zijn.
id_key: de key die de uitvoerder van de actie identificiëert.

Rationale and design issues:
zie ook deployment diagram voor toegang tot de dokter database.

grantAccess

Resources provided:
grantAccess(to_key, data_key, action, from_key, period)
De persoon verbonden aan from_key geeft toegangrechten voor periode period aan to_key om actie action uit te voeren op het document met key data_key.

Error handling:
De persoon met from_key heeft geen toelating de toegangsrechten voor actie action toe te kennen.
De persoon met to_key kan geen toegangsrechten krijgen om actie action uit te voeren.

Element requirements:
to_key: de identificatie key van de persoon aan wie toegangsrechten worden toegekend. data_key: de key verbonden met een data document.
action: een actie waarvoor rechten worden uitgedeeld. Action kan hier ook zijn het toekennen van toegangsrechten aan een andere persoon. from_key: de identificatie key van de persoon die de toegangsrechten uitdeelt. period: de periode hoe lang de persoon met to_key toegang krijgt tot de data.

Rationale and design issues:
Door de grantAccess toe te voegen kunnen toegangsrechten worden toegekend aan andere personen en kan dit voor een bepaalde tijdsduur. Action is hier niet enkel read, modify of push zodat bepaalde personen ook toelating krijgen om toegangsrechten door te geven. Zo zou een specialist bijvoorbeeld ook toegang kunnen krijgen om read toegang tot het dossier ook door te geven aan een andere specialist.
Zie ook deployment diagram voor toegang tot de key database.

Element behavior

De policy component zal authorisatie toekennen aan bepaalde acties, als die persoon de actie mag uitvoeren. Indien dat niet mag zal de authorisatie geweigerd worden. Bepaalde personen hebben ook rechten om nieuwe rechten toe te voegen. Deze worden via *grantAccess* toegevoegd.

Security

De gedetailleerde uitwerking van de security component is te vinden in 0.12 Client and Server View: Security.

ExternalActions

Element interfaces

De Element interface component voorziet de interfaces voor het loggen naar de overheid en het verzenden van de RIZIV data naar de overheid.

log

Resources provided:
log(data)
De data wordt gelogd naar de overheid toe. De component zorgt dat enkel data die mag gelogd worden verstuurd wordt.
Element requirements:
data: de data om te loggen.
Rationale and design issues:
De data die mag gelogd worden kan eventueel gedefinieerd worden door een externe instantie, in dit geval de overheid. Zie ook deployment diagram voor toegang tot de overheid.

RIZIV

Resources provided:
riziv(data)
De data wordt verzonden naar de overheid toe. De component zorgt data alle data die moet verzonden worden, verzonden wordt.
Error handling:
Wanneer de data niet volledig is wordt een error teruggeven.
Element requirements:
data: de data om te verzenden naar het RIZIV.
Rationale and design issues:
Zie ook deployment diagram voor toegang tot de overheid.

Element behaviour

Deze component zal de logging en RIZIV acties uitvoeren naar de overheid toe. In normale omstandigheden zullen deze acties onmiddellijk uitgevoerd worden. In het geval dat de server van de overheid niet kan bereikt worden zullen de acties gequeued worden totdat de server terug beschikbaar is. Op het moment dat de server opnieuw beschikbaar is zal de queue element per element verzonden worden naar de overheid.
Deze queue kan anders uitgewerkt worden voor log en RIZIV.

0.10.3 Context diagram

0.10.4 Variability guide

De **ExternalActions** component kan ook zo uitgewerkt worden dat de acties slechts op bepaalde tijdstippen naar de server van de overheid worden uitgevoerd. De acties worden dan altijd gequeued. Op bepaalde tijdstippen worden de queues dan uitgevoerd en leeggemaakt. Er zou ook kunnen rekening gehouden

worden met het aantal elementen in de queue. De queue kan dan worden uitgevoerd en leeggemaakt als een bepaalde treshold van elementen bereikt is. Tenslotte is ook een combinatie van beide vorige mogelijk. De queue kan normaal op een bepaald tijdstip leeggemaakt worden tenzij er te veel elementen in de queue zitten; dan zou de queue eerder worden geleegd. Omgekeerd kan het ook dat de queue altijd geleegd worden zodra een bepaald elementen in de queue zitten, maar ook op bepaalde tijdstippen om in het geval van low traffic de berichten niet te lang op de server te houden.
Het is mogelijk verschillende instellingen te gebruiken voor log en RIZIV.
Het is ook mogelijk om de interfaces zo te implementeren dat bepaalde berichten een hogere prioriteit hebben en altijd direct worden verzonden naar de overheid.

0.10.5 Architecture background

TODO!!!

Rationale

Analysis results

Assumptions

0.10.6 Other information

0.10.7 Related view packets

In het deployment view is te zien hoe de connect naar de externe servers tot stand komt.
In de interactie diagramma's zijn ook voorbeelden te vinden van hoe bepaalde interacties precies in hun werk gaan.

0.11 C&C Client and Server View: Client

0.11.1 Primary presentation

0.11.2 Element catalog

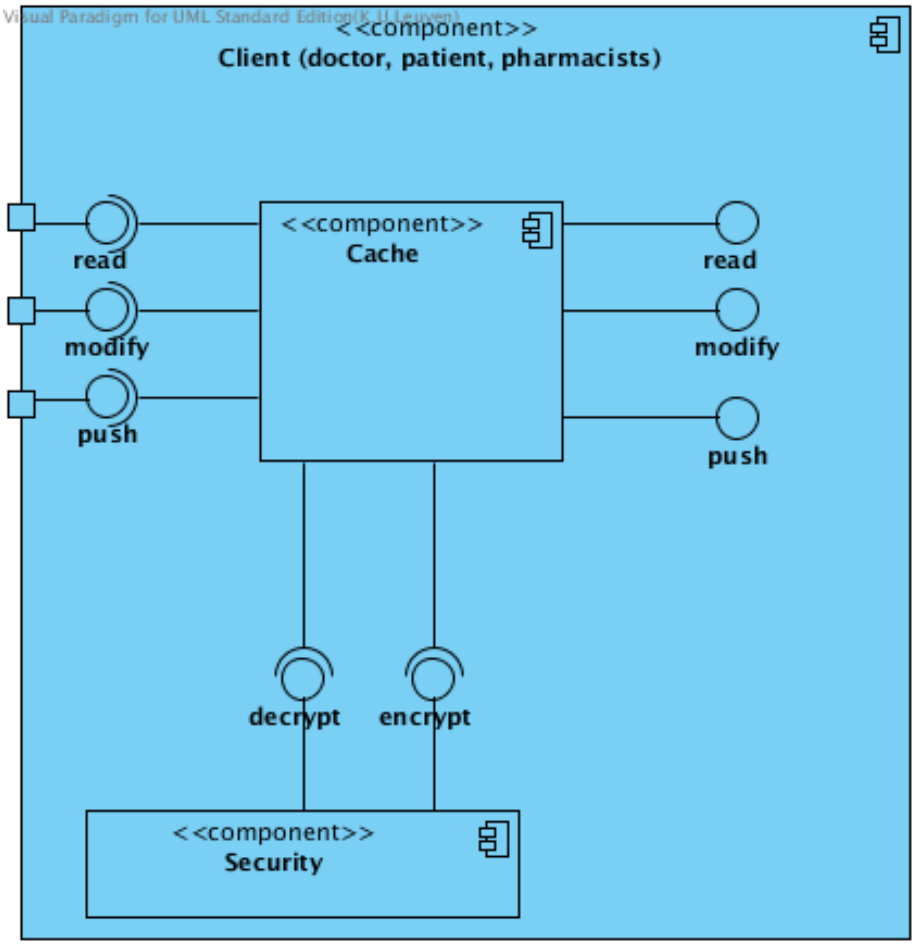
Cache

De cache component is een lokale cache van bestanden die ook op de server of externe servers aanwezig zijn. Dit element voorziet de toegang tot de data.

Element interfaces

read

Resources provided:
read(data_key)
De data verbonden met data_key wordt gelezen.


Visual Paradigm for UML Standard Edition(K.U.Leuven)
<<component>>
Client (doctor, patient, pharmacists)
read
modify
push
<<component>>
Cache
read
modify
push
decrypt
encrypt
<<component>>
Security

Element requirements:
data_key: de key om de data op te vragen.

Rationale and design issues:
Zie hieronder voor een beschrijving van het algemeen gedrag van de cache.

modify

Resources provided:
modify(data_key, data)
De aanpassing van de data verloopt door het mergen van data met de bestaande data.
Error handling:
Wanneer het mergen van de data faalt zal een error teruggegeven worden. Er kan gevraagd worden aan de uitvoerder van de actie om manueel een merge uit te voeren.
Element requirements:
data_key: de key om de data op te vragen.
data: een patch van data die moet aangepast worden op locatie data_key.

Rationale and design issues:
In het geval dat een merge actie zou falen, kan aan de client gevraagd worden om de merge manueel te doen, of om een volledig document te verzenden.

push

Resources provided:
push(data_key, data)
Het pushen van de data gebeurt gelijkaardig zoals bij **modify**.
Element requirements:
data_key: de key om de data op te vragen.
data: data die moet toegevoegd worden aan document met key: data_key.

Rationale and design issues:
Hier kan geen patch gestuurd worden omdat een push actie enkel toelaat nieuwe data toe te voegen aan een dossier.

Element behavior

Wanneer een item uit de cache gelezen wordt (*read*), verstuurt de cache eerst de vraag naar de server als het gegeven item up-to-date is. Als dit het geval is dan wordt het bestand uit de cache aan de gebruiker weergegeven, is dit niet het geval dan stuurt de server het meest recente bestand terug. Dit wordt dan opgeslagen in de cache en dan teruggegeven aan de gebruiker.
Het schrijven van een item (*write*) gebeurt door het item eerst lokaal te schrijven, daarna verstuurt de cache een write operatie naar de server. Deze write operatie verloopt door enkel een patch te sturen van de nieuwe data, het volledige document wordt dus niet weggeschreven maar enkel de wijzigingen. Wanneer de cache een antwoord gekregen heeft van de server dat de schrijfoperatie correct verlopen is mag de cache het geschreven item uit de cache verwijderen.

Een *push* operatie is gelijkaardig aan de schrijf operatie van hierboven. Het enige verschil is dat het in het geval van een *push* gaat over een operatie waarbij de gebruiker geen toegang heeft tot het te schrijven document. Hij kan enkel een data-veld toevoegen, een veld wijzigen of overschrijven is niet mogelijk.
Een laatste belangrijk iets is hoe de cache moet reageren in verband met het falen van het netwerk. Indien de cache geen reactie krijgt van het netwerk mag hij de lokale files teruggeven. In het geval dat deze cache gegevens oud zijn kan dit aan gebruiker gemeld worden. Op het moment dat de cache de server terug kan bereiken zal hij zijn wijzigingen gaan opslaan. Hiervoor wordt een merge algoritme gebruikt.

Security

De security component is op zich gelijk aan deze in de server. De gedetailleerde uitwerking is te vinden in 0.12 Client and Server View: Security.

0.11.3 Context diagram

TODO

0.11.4 Variability guide

Verschillende implementaties van de Client component zijn mogelijk. Wanneer de client lokaal op een pc wordt uitgewerkt voor een dokter zal de cache waarschijnlijk zo veel mogelijk data bijhouden.
Voor een client van de dokter die bijvoorbeeld op een smartphone draait is het onrealistisch om te verwachten dat deze alle data zal cachen. De client kan dan bijvoorbeeld zo uitgewerkt worden dat de cache van de smartphone iedere morgen voor de huisbezoeken gesynct wordt met de nodige patiënten van die dag.
De centrale pc van de dokter kan een gelijkaardige synchronisatie techniek implementeren voor geplande bezoeken van patiënten.
Nog een ander geval is het geval van de client voor de apotheek of voor de patiënt. In het geval van een client voor de patiënt zal de cache enkel het dossier van de patiënt zelf bijhouden. Voor de apotheek hoeft de cache helemaal geen data te cachen.
Een laatste geval is wanneer de cache op een webclient draait. In dit geval kan de caching aangepast worden naar de mogelijkheden van de webserver. Hier moet dan wel rekening gehouden worden met de wetgeving die centrale opslag van patiënten data verbiedt. Het is mogelijk dat een webserver zoveel zou gaan cachen dat dit zou kunnen gezien worden als een centrale opslag. Omdat dit niet toegelaten is moet men er bij de implementatie van deze cache rekening mee houden. De cache zou zo kunnen aangepast worden om slechts een bepaald aantal documenten bij te houden of enkel de documenten die in een bepaalde tijdspanne opgevraagd zijn.
Het schrijven van de cache naar de server kan ook per client ingesteld worden. Er kan na iedere schrijf actie naar de server geschreven worden, er kan geschreven worden als een dossier wordt gesloten. Het is ook mogelijk dat de server tijdelijk niet beschikbaar is, dan worden alle schrijfacties in een queue

gezet en die kunnen dan geschreven worden op het moment dat de server terug beschikbaar is.

0.11.5 Architecture background

De belangrijkste component van de client, de cache, is gekozen om een aantal redenen. De eerste is het verzorgen van `availability`. Wanneer er een netwerk failure is kan een dokter nog aan alle data die lokaal in de cache zit. In het geval dat de client een dokter is met een lokale pc zal deze cache waarschijnlijk vrij compleet zijn en ook up to date zijn, zeker wanneer het gaat om geplande bezoeken, aangezien de cache voordien al kan synchronizeren met de server.
Een ander belangrijke kwaliteitsvereiste die we hier niet mogen vergeten is de `performance`. Een lokale cache zal een belangrijke snelheids winst opleveren aangezien een stuk minder data over het netwerk moet verzonden worden.
Het wegschrijven van de wijzigingen in plaats van het volledige document bij een *write* operatie naar de server zorgt ook voor een belangrijke snelheids winst. Op deze manier zal veel minder data over het netwerk moeten worden verstuurd.

0.11.6 Other information

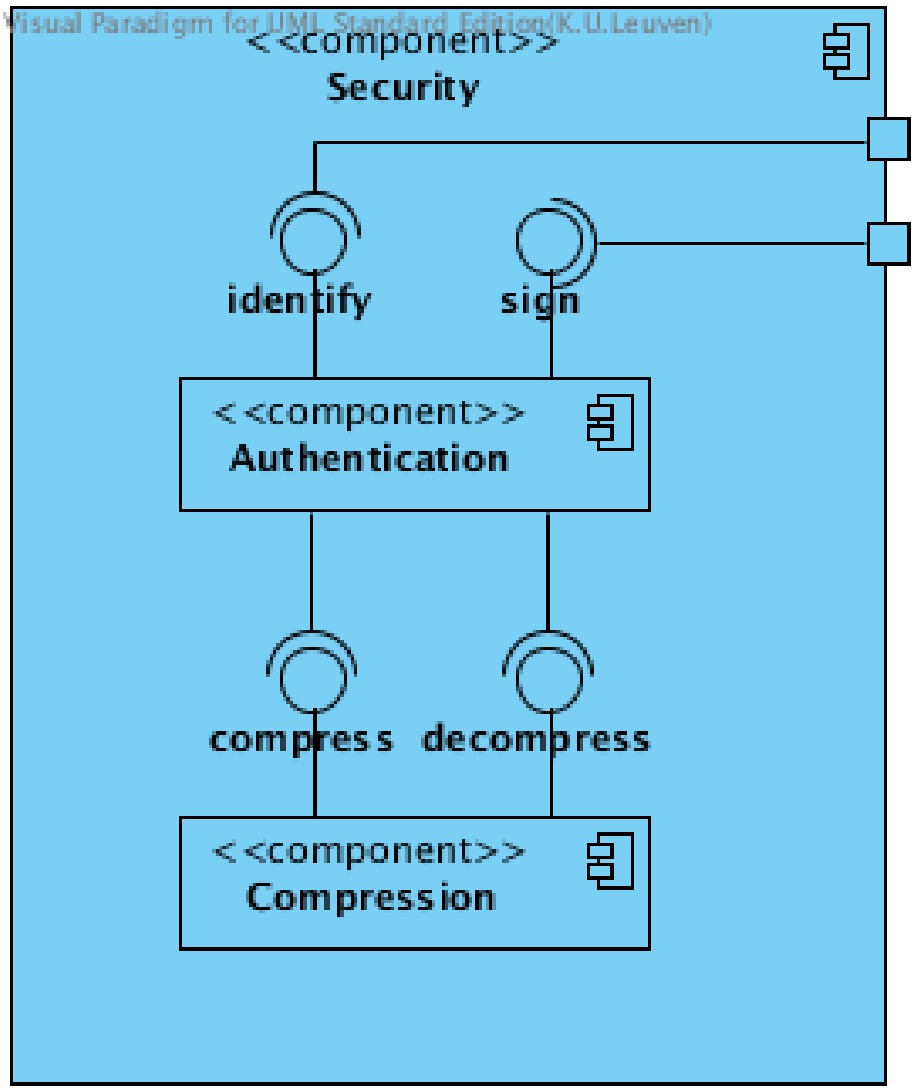
[None]

0.11.7 Related view packets

Deze client kant moet gezien worden in de volledige overview (zie 0.9 Client and Server Overview. Daarnaast is ook het deployment model belangrijk

0.12 C&C Client and Server View: Security

0.12.1 Primary presentation


Visual Paradigm for UML Standard Edition(K.U.Leuven)
<<component>>
Security
identify
sign
<<component>>
Authentication
compress
decompress
<<component>>
Compression

0.12.2 Element catalog

Authentication

Compression

Elements and their properties

Relations and their properties

Element interfaces

Element behavior

0.12.3 Context diagram

0.12.4 Variability guide

0.12.5 Architecture background

Rationale

Analysis results

Assumptions

0.12.6 Other information

0.12.7 Related view packets

0.13 Allocation Deployment View

0.13.1 Primary presentation

0.13.2 Element catalog

Elements and their properties

Relations and their properties

Element interfaces

Element behavior

0.13.3 Context diagram

0.13.4 Variability guide

0.13.5 Architecture background

Rationale

Analysis results

Assumptions

0.13.6 Other information

0.13.7 Related view packets

0.14 Interaction Diagrams

0.14.1 Primary presentation

0.14.2 Element catalog

Elements and their properties

Relations and their properties

Element interfaces

Element behavior