Katholieke
Universiteit
Leuven
Faculteit
Ingenieurswetenschappen
Departement
Computerwetenschappen

P&O, Computerwetenschappen
(hoofdrichting)
Positiebepaling met WiFi-signalen

Groep H3
verslag
13 december 2006

Cedric Cuypers
Chris Adriaensen*
Daniel Balog
Guy Van Den Broeck
Jef Maerien
Philippe Dellaert*
Wouter Van Ranst

* groepscoördinator
** secretaris

0. Inhoud

0. Inhoud **2**
1. Inleiding **3**
2. Ontwerp **3**
2.1. Machine-niveau 3
2.2. Conceptueel niveau 4
2.3. Implementatie-niveau 5
2.4. Evaluatie 7
3. Implementatie **8**
3.1. Positioningsytem **8**
3.1.1. Fingerprinting-methode *8*
A. Fingerprint 9
B. Scan 9
3.1.2. K-nearest-neighbour-methode *10*
A. Afstand tussen fingerprint en scan voor 1 station 10
I. Verschil van gemiddelden 10
II. Verschil van medianen 10
III. Correlatie tussen de verdelingen 10
IV. Hypothesetesten 12
V. Niet-lineaire transformatie: exponentiëel 1 13
VI. Niet-lineaire transformatie: exponentiëel 2 13
B. Afstand tussen fingerprint en scan 13
I. Oneindig-norm 14
II. Gemiddelde van afwijkingen 14
III. 2-norm 14
IV. Gewogen 2-norm 14
3.1.3. Training *14*
3.1.4. Testen *16*
3.1.5. Besluit *18*
3.1.6. UML-diagram 20
3.1.7. Evaluatie 22
3.1.8. Vooruitzicht 22
3.1.9. Bronnen 22
3.2. Client **23**
3.2.1. UserGUI 23
3.2.2. MapGUI 25
3.2.3. UML-diagram 26
3.2.4. Evaluatie 26
3.2.5. Vooruitzicht 26
3.3. Communication **27**
3.3.1.UML-diagram 29
3.3.2. Evaluatie 29
3.3.3. Vooruitzicht 29
3.4. Server **30**
3.4.1. Database 30
3.4.2. DatabaseCore 30
3.4.3. UML-diagram 31
3.4.4. Evaluatie 32
3.4.5. Vooruitzicht 32
4. Team **33**
4.1. Peer-assesment 33
4.2. Werkbelasting 33
4.3. Evaluatie 33
4.4. Vooruitzicht 34
Bijlage Werkbelasting 35

1. Inleiding

Positiebepaling met WiFi-signalen geeft ons de mogelijkheid om binnenshuis de positie van een ontvanger te bepalen. De rol van ontvanger wordt in onze toepassing ingevuld door een *Personal Digital Assistent* (PDA).

De PDA bevindt zich in onze toepassing in een museum, galerij, … Het verschaft een student of een groep studenten op een *location of interest* (LOI) van interessante *multiple choice* vragen ivm de locatie. Alle antwoorden worden opgeslagen en kunnen ten allen tijde geraadpleegd worden.

De gebruiker communiceert met het systeem dmv een gebruiksvriendelijke *Graphical User Interface* (GUI). De communicatie tussen de PDA en de server verloopt via *Remote Method Invocation* (RMI). Op deze of een andere server bevindt zich een database waarin alle nuttige informatie wordt opgeslagen. De communicatie hiermee verloopt via *Java Database Connectivity* (JDBC). Door gebruik te maken van de *k-nearest-neighbours*-methode (die op zijn beurt gebruik maakt van de *fingerprinting*-methode) kan de server de positie van de PDA tot op ongeveer 4 meter nauwkeurig bepalen. Alle programma's zijn geschreven in de programmeertaal *JAVA*.

2. Ontwerp

We bekijken het ontwerp van onze toepassing op meerdere niveaus, van een abstract naar een gedetailleerd niveau.

2.1. Machine niveau

Het eerste niveau waarop we ons ontwerp bekijken is op *machine-niveau* (zie Fig. 1). De PDA neemt op een bepaald tijdstip enkele metingen van de sterkten van de signalen van de nabijgelegen WiFi-stations. Deze metingen worden doorgegeven aan de server via RMI dankzij de draadloze communicatie. De server bepaalt hieruit of de PDA zich binnen een bepaalde afstand van een LOI bevindt en raadpleegt de database of er quiz-vragen beschikbaar zijn voor die locatie via JDBC.

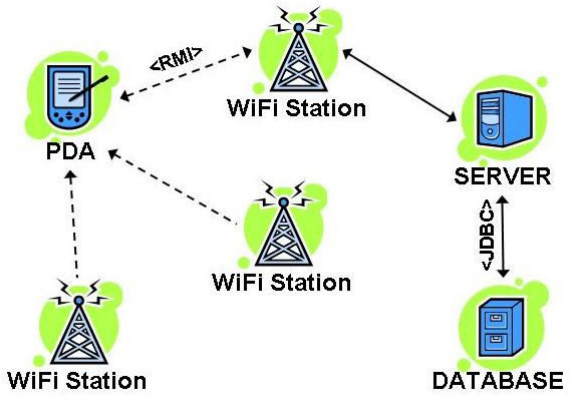


Fig. 1 Machine-niveau

2.2. Conceptueel niveau

Het hierop volgende niveau dat we bekijken is het *conceptuele niveau* (zie Fig. 2). Deze toont de verschillende onderdelen in de machines. Hier volgt een klein overzicht van de onderdelen en hun bijhorende functie.

- **UserGUI**
  Is verantwoordelijk voor de gebruiksvriendelijke grafische communicatie met de gebruiker.
- **ClientCore**
  Is het centrale onderdeel op de PDA. Deze is verantwoordelijk voor de correcte communicatie tussen de onderdelen op de PDA.
- **WiFiScanner**
  Neemt metingen van de sterkten van de signalen van nabijgelegen WiFi-stations.
- **Com. (Communication)**
  Staat in voor de communicatie tussen PDA en server.
- **ServerCore**
  Is het centrale onderdeel op de server. Deze is verantwoordelijk voor de correcte communicatie tussen de onderdelen op de server.
- **PositioningSystem**
  Bepaalt adhv de metingen en bijhorende informatie in de database of de PDA zich binnen een bepaalde afstand bevind van een locatie.
- **DatabaseCore**
  Staat in voor de communicatie met de database.

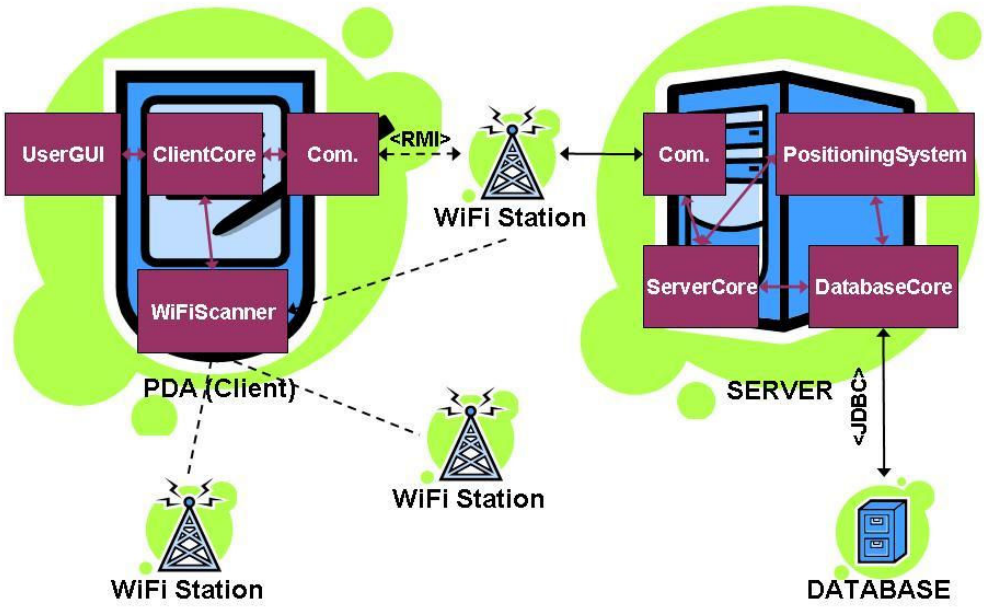


Fig. 2 Conceptueel niveau

Aan de hand van dit niveau delen we onze toepassing op in 4 onderdelen. We bespreken deze later in detail.

- **Positioningsystem**
  Hieronder verstaan we *PositioningSystem*.
- **Client**
  Hieronder verstaan we de *UserGUI* en de *ClientCore*.
- **Communication**
  Hieronder verstaan we *Communication* en de bijhorende communicatie tussen PDA en server via RMI.
- **Server**
  Hieronder verstaan we de *ServerCore*, *DatabaseCore* en de database zelf.

2.3. Implementatie-niveau

Het laatste, meest gedetailleerde niveau dat we bekijken is het *Unified Modeling Language*-diagram (UML-diagram, zie Fig. 3). Deze toont de implementatie van de onderdelen met hun attributen en methoden. Op dit punt tonen we enkel het UML-diagram van de pakketten van het ontwerp en de pakketen *User* en *Question*. De meer gedetailleerde diagramma van de andere onderdelen worden later behandeld bij hun respectievelijk onderdeel behandeld.

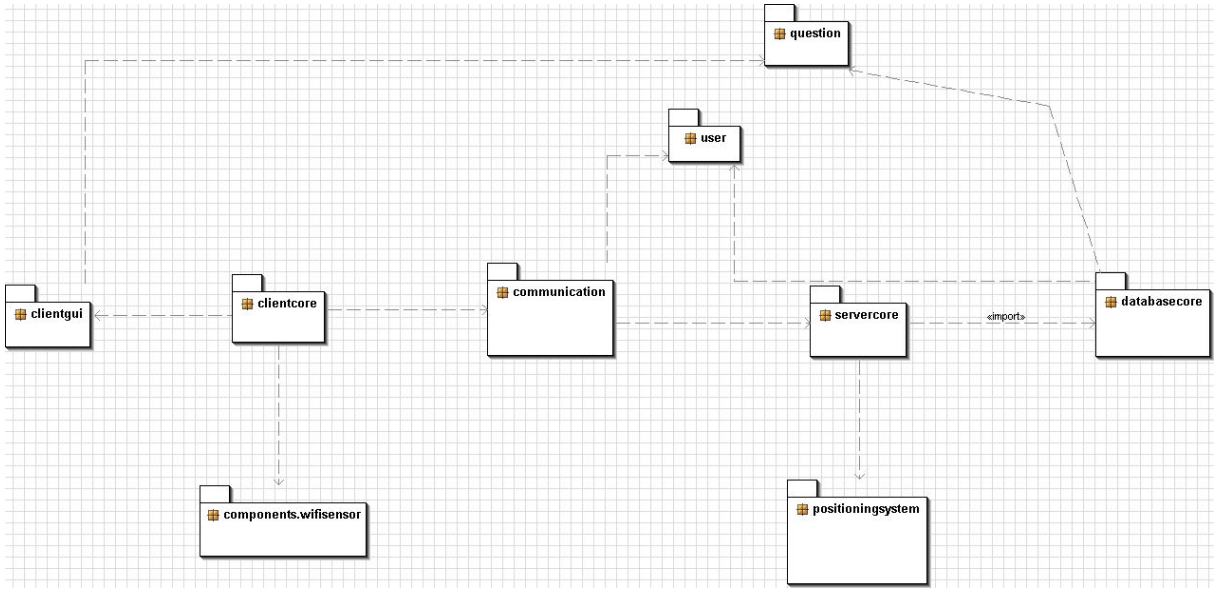


Fig. 3 UML-diagram van de packages

Het *User*-pakket (zie Fig. 4) is een pakket dat de gebruikers van onze toepassing voorstelt. Deze klassen worden ook gebruikt om te communiceren tussen PDA en server.

Het *Question*-pakket (zie Fig. 5) bevat klassen die nodig zijn om vragen, lijsten van vragen en antwoorden aan te maken en door te sturen via RMI.

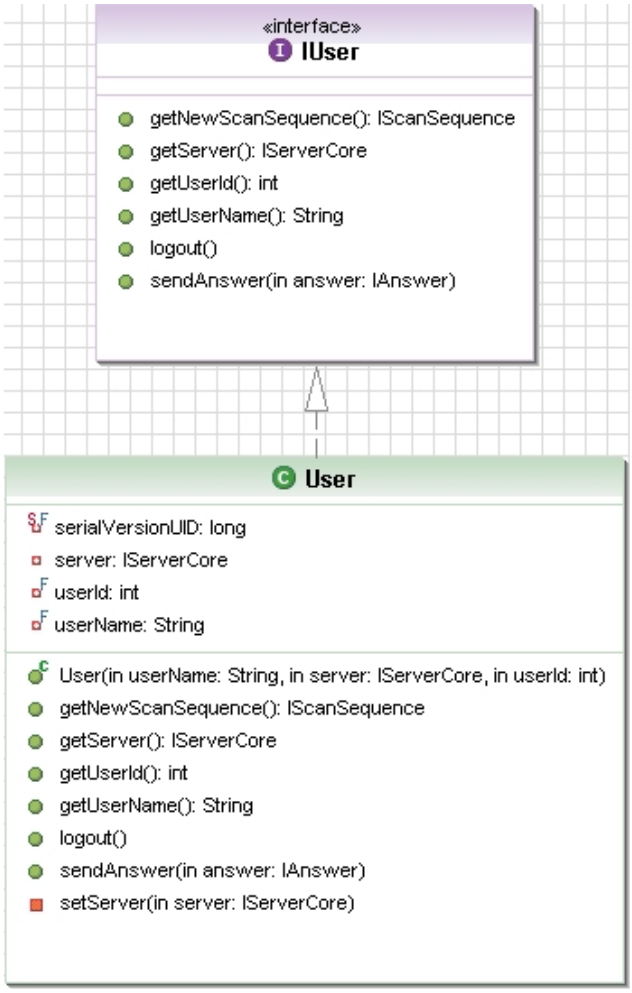


Fig. 4 User-package

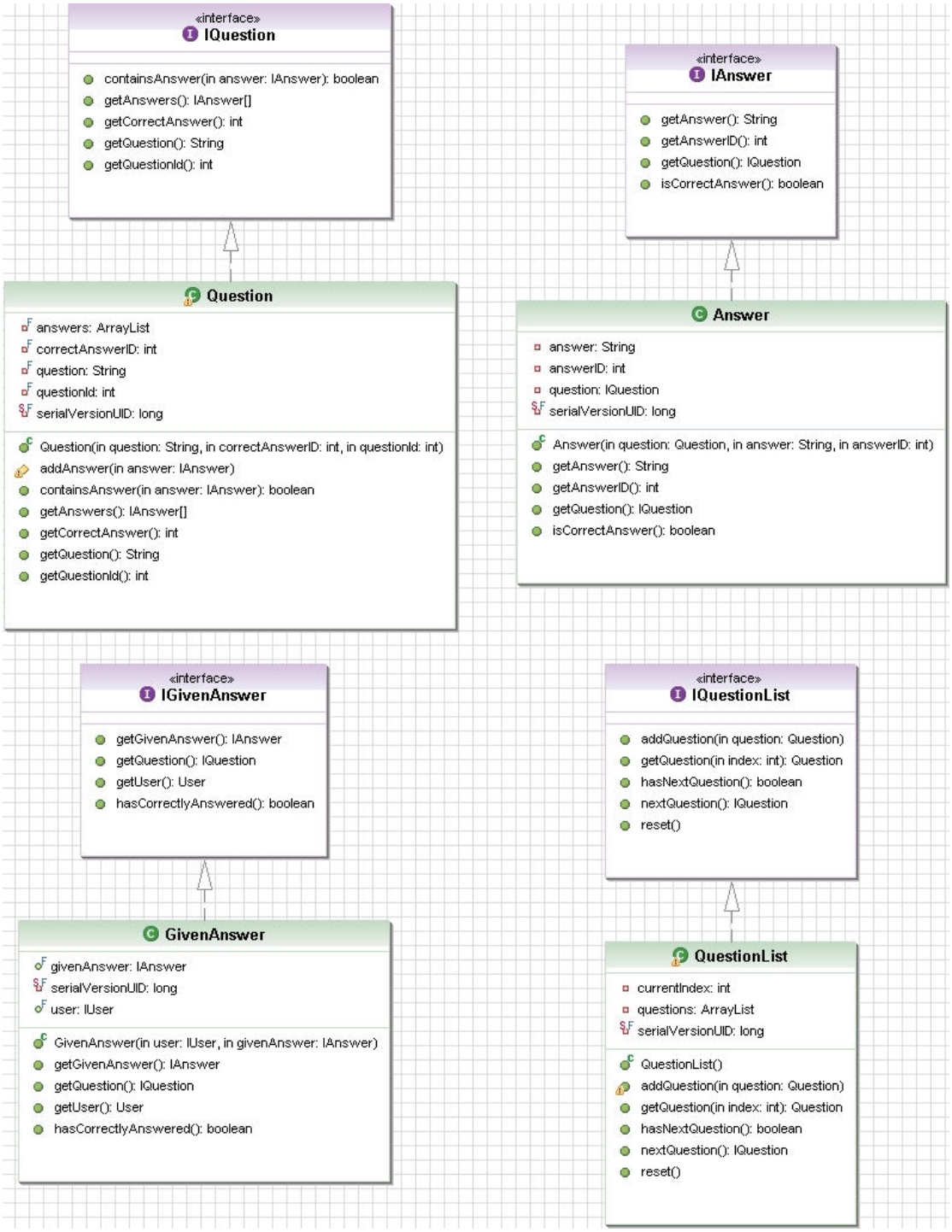


Fig. 5 Question-package

2.4. Evaluatie

Elke functie heeft zijn eigen component in ons ontwerp. Dit zorgt ervoor dat deze relatief onafhankelijk staat van de rest het programma. Dit maakt het testen relatief eenvoudig en zorgt ervoor dat er een goed overzicht is over het programma. Ook zorgt dit dat de componenten eventueel gemakkelijk vervangen kunnen worden, indien een bepaald deel op een andere manier geïmplementeerd zou worden. Daarnaast laat het voldoende ruimte voor eventuele uitbreidingen. In het tweede semester kan de code dus relatief eenvoudig hergebruikt worden.

3. Implementatie

We bekijken de implementatie van de verschillende onderdelen van ons ontwerp.

3.1. Positioningsystem

Team: ***Cedric, Chris & Guy***

Omvat: ***PositioningSystem***

In het oorspronkelijk ontwerp gebruikten we de *fingerprinting*-methode en zouden we de *k-nearest-neighbour*-methode pas in het 2de semester implementeren. Dankzij goede vooruitgang hebben we reeds dit semester de *k-nearest-neighbour*-methode geïmplementeerd. Om deze goed te begrijpen bekijken we eerst de *fingerprinting*-methode. (Een volledig gedetailleerd overzicht van alles positiebepalingsklassen is weggelaten voor de grootte van het verslag te beperken.)

Elke methode bestaat in het algemeen uit 2 fases:

- *Training* = data invoeren in het systeem
- *Positiebepaling* = positie bepalen

3.1.1. Fingerprinting-methode

Een eerste klasse van positiebepalingsmethodes gebruikt de *fingerprinting*-methode, die een signaal-gebaseerde methode is. Deze methode maakt dus enkel gebruik van de ontvangen signaal-sterkten van de verschillende WiFi-stations en een bijhorende *Radio-Map* (RM). De RM is een plattegrond (raster) van de omgeving met daarop de verschillende stations aangeduid.

De *fingerprinting*-methode gaat de totale ruimte discretiseren en opmeten. Onder dit opmeten verstaan we het opmeten van de ontvangen signaalsterkten van de verschillende WiFi-stations op een bepaalde positie. Deze signaalsterkten per station horende bij een bepaalde plaats noemt men een *fingerprint* (FP). Men neemt een aantal FP's van de omgeving en plaats ze mee op de RM. De gezochte positie wordt benaderd door de dichtstbijzijnde *fingerprint*. Hoe men de dichtstbijzijnde *fingerprint* bepaald staat compleet vrij en kan de positiebepaling sterk verbeteren. Het gebruiken van statistiek hierin wordt aangeraden.

Een voordeel van deze methode is dat dat men de benadering kan verbeteren door de trainings-fase te vergroten (dus meer FP's). Deze benadering kan niet oneindig verbeterd worden als gevolg van de storingen en ruis op het signaal. Een belangrijk nadeel van de methode is dat ze geen extrapolatie toelaat (dit vormt geen probleem aangezien we in onze toepassing geen gebruik maken van extrapolatie).

We bespreken eerst kort wat we onder de termen *fingerprint* en *scan* verstaan.

A. Fingerprint

Een *fingerprint* is gekoppeld aan een positie op de RM en heeft bijgevolg coördinaten (x,y) of (x,y,z) zoals aangegeven op de RM. Een *fingerprint* bevat ook voor elk omliggend (meetbaar) station een set van metingen. In plaats van de sets bij te houden distileren we uit de sets van metingen een stochastische verdeling. We veronderstellen dat elk set i van metingen gaussisch verdeeld is met gemiddelde μ_i en standaard deviatie σ_i. De gaussische dichtheidsfunctie voor set i is (zie Fig. 6)

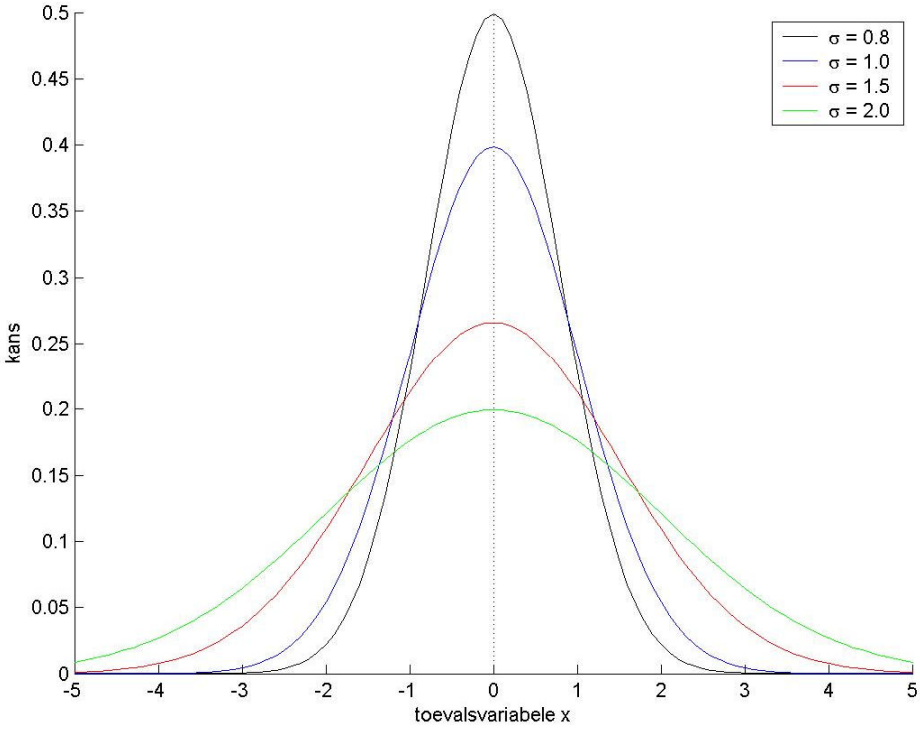


Fig. 6 Gaussische verdeling met gemiddelde $\mu = 0$ en standaard deviatie σ

$$f_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2}$$

voor $x \in \mathbb{R}$, $\mu_i \in \mathbb{R}$ en $\sigma_i > 0$

We moeten enkel het gemiddelde μ_i en de standaard deviatie σ_i bijhouden. We houden ook de mediaan bij (deze is minder gevoelig aan uitschieters) en het aantal metingen n.

B. Scan

Een scan wordt gemaakt wanneer men de positie wil bepalen. De scan maakt voor elk omliggend (meetbaar) station een set van metingen. Hiervan houden we ook enkel het gemiddelde μ, standaard deviatie σ, mediaan en aantal metingen n bij.

Een scan is ook gekoppeld aan een positie en heeft bijgevolg coördinaten (x, y) of (x, y, z) op de RM. Deze coördinaten zijn niet gekend en de positiebepaling heeft juist tot doel deze coördinaten te schatten of te benaderen door $(\tilde{x}, \tilde{y})$ of $(\tilde{x}, \tilde{y}, \tilde{z})$.

3.1.2. K-nearest-neighbour-methode

Een volgende klasse van positiebepalingsmethodes gebruikt de *k-nearest-neighbour*-methode, die een uitbreiding is van de *fingerprinting*-methode. We zoeken in deze methode niet enkel de dichtstbijzijnde *fingerprint* maar de K dichtstbijzijnde FP's. De gezochte positie wordt hier dan benaderd door het (gewogen) midden van de posities van de FP's.

Men kan ook deze methode op vele manieren implementeren aangezien men vrij is in de definitie van de dichtstbijzijnde *fingerprint*.

A. Afstand tussen fingerprint en scan voor 1 station

In de eerste plaats moeten we de afstand tussen de *fingerprint* en een *scan* definiëren voor één enkel station.

I. Verschil van gemiddelden

De simpelste mogelijkheid is ongetwijfeld gewoon het verschil nemen tussen de gemiddelde waarden van het ontvangen signaal van dat bepaald station op de fingerprint en op de plaats waar we nu staan. Het voordeel is dat dit een zeer simpele methode is. Nadeel is dat de variaties van de verschillende signalen in deze methode niet worden mee ingerekend. Ook wordt er hier geen rekening gehouden met het feit dat het verschil tussen de ontvangen signalen duidt op een niet-lineair verband van afstanden. Als er slecht 10 dB verschil is tussen de signalen duid dit op een verschil van ongeveer 3.16 meter is. Een verschil van 20 dB verschil duid op ongeveer 10 meter verschil en een verschil van 30 dB duid op 30 meter verschil.

II. Verschil van medianen

De volgende mogelijkheid van afstand is de mediaan waarde. Een mediaan heeft dezelfde voor en nadelen als het gemiddelde, met het kleine verschil dat het minder gevoelig zal zijn aan eventuele grote afwijkingen. Dit zal geen groot verschil geven aangezien we toch maar over een relatief korte tijdspanne meten. Enkele afwijkingen zullen waarschijnlijk toch bijna onmogelijk gedetecteerd worden aangezien de hoeveelheid meetgegevens te beperkt is.

III. Correlatie tussen de verdelingen

De correlatie stelt de *'gelijkheid'* voor tussen verdelingen. De formule voor de correlatie tussen 2 verdelingen met dichtheidfunctie $f_1(x)$ resp. $f_2(x)$ is

$$corr_{12} = \int_{-\infty}^{\infty} f_1(x) f_2(x) dx$$

of

$$corr_{12} = \int_{-\infty}^{\infty} f_{corr_{12}}(x) dx$$

met $f_{corr_{12}}(x)$

$$f_{corr_{12}}(x) = f_1(x) f_2(x)$$

Als de verdelingen gaussisch zijn verkrijgen we mits een kleine vereenvoudiging

$$f_{corr_{12}}(x) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2} e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

We nemen voor de afstand tussen de 2 sets van metingen de inverse van de correlatie van die 2 sets.

$$distance_{12} = \frac{1}{corr_{12}}$$

De integraal is niet analytisch op te lossen en moet dus numeriek opgelost worden. Een probleem dat zich dan stelt is het feit dat we niet over een oneindig interval kunnen integreren. We kunnen het interval zo inkorten dat we geen belangrijk verschil in de waarde van de integraal ondervinden. We gebruiken voor de numerieke integratie een *automatisch integratie met de regel van Simpson*. Dit algortime gaat in elk deelinterval de functie benaderen door een 2de graads veelterm, hiervoor worden de uiteinden en het midden van het interval gebruikt.

We bekijken een voorbeeld met 3 sets van metingen.

$$set_1 : \mu_1 = 0 \text{ en } \sigma_1 = 1$$
$$set_2 : \mu_2 = 2.5 \text{ en } \sigma_1 = 2$$
$$set_3 : \mu_2 = 2 \text{ en } \sigma_1 = 0.8$$

Berekenen we de correlatie tussen set_1 en set_2 en de correlatie tussen set_1 en set_3 dan zien we dat desondanks het gemiddelde van set_3 het dichtst bij het gemiddelde van set_1 ligt (zie I.), met de correlatie set_2 het dichtst bij set_1 ligt (zie Fig. 7).

$$corr_{12} = 0.0473.. \text{ dus } distance_{12} = 21.11..$$
$$corr_{13} = 0.0459.. \text{ dus } distance_{13} = 21.74..$$

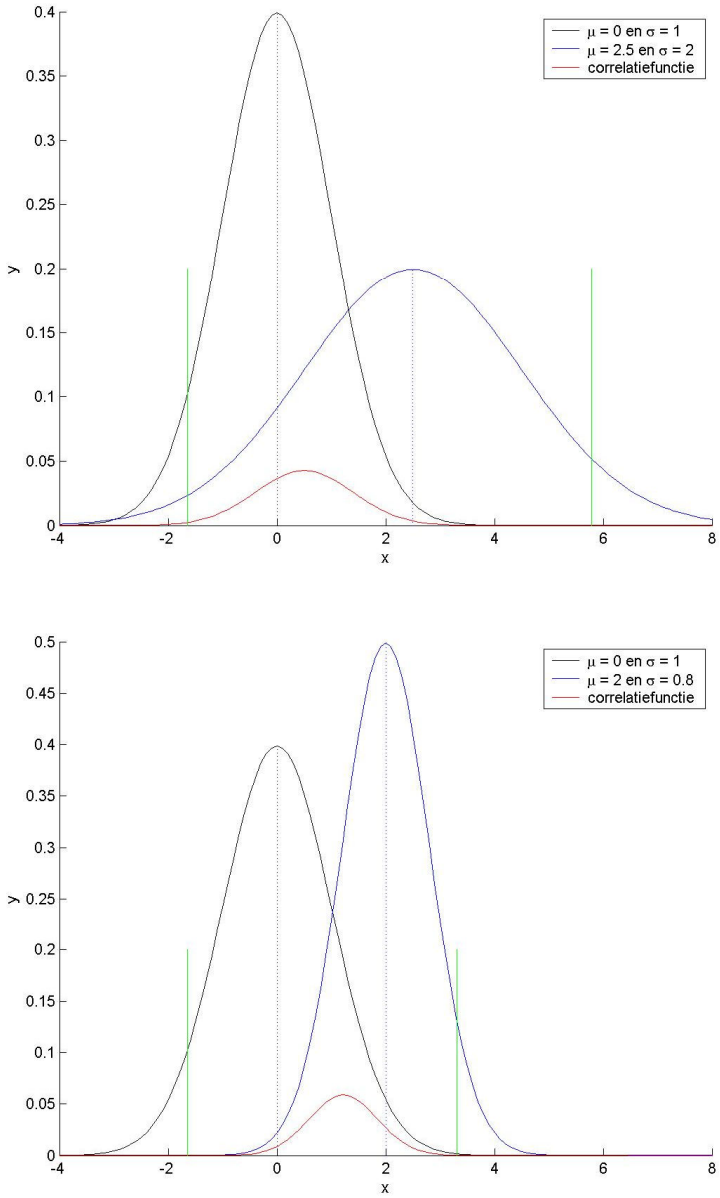


Fig. 7 Gaussische verdelingen van de 3 sets en de correlatiefuncties

Voordeel van deze methode is dat het wel alle informatie gebruikt die er in de metingen ter beschikking is. Nadeel is dat het meer rekenwerk vraagt dan het simpel gemiddelde en dat het geen rekening houdt met het niet-lineaire verband tussen signaalsterkte en afstand.

IV. Hypothesetesten

Twee andere methodes halen we uit de statistiek. We gaan een hypothesetest ondernemen om te bepalen wat de kans is dat een uitspraak waar is. Hoe groter de kans dat de uitspraak waar is (de gemiddeldes van de sterktes van de twee signalen zijn gelijk) hoe groter de kans dat we op de betreffende fingerprint zijn. Dit staat ook bekend als de *Student-t Test* en wordt vaak gebruikt om na te gaan of 2 reeksen metingen uit dezelfde populatie komen. In beide gevallen is de nulhypothese (H0) dat de gemiddeldes van de populaties gelijk zijn en de alternatieve hypothese (H1) dat de

gemiddeldes niet gelijk zijn..
De eerste relatief simpelste methode is ervan uit gaan dat ook de standaard deviaties gelijk zijn.

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1}{n_1} + \frac{\sigma_2}{n_2}}}$$

Een tweede methode gaat ervan uit dat de standaarddeviaties verschillend zijn.

$$t = \frac{\mu_1 - \mu_2}{s_{\mu_1 - \mu_2}}$$

met

$$s_{\mu_1 - \mu_2} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{(n_1 + n_2 - 2)}\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}$$

Beiden zijn Student-t verdeeld. Omdat het aantal metingen voor de fingerprint zo hoog is, benadert de verdeling (met 100-den vrijheidsgraden) de normaalverdeling. We benaderen de bovenkwantiel-kans met een lineair stuksgewijze interpolatie. Als totaal resultaat krijgen we dus de kans in percent dat de gemiddeldes niet gelijk zijn.

Voordeel van deze methodes is dat het ook weer alle data gebruikt dat ter beschikking is. Mogelijk nadeel is dat het werkt op de ontvangen signaalsterktes in plaats van de afstand.

V. Niet-lineaire transformatie: exponentiëel 1

Deze methode berekent het verschil in dB tussen de sterkte van de ontvangen signalen en de signalen van de fingerprint. Dit verschil gebruiken we dan om de vermoedelijke minimum afstand te berekenen tussen de gebruiker en de fingerprint. Eigenschappen hiervan zijn dat het wel rekening houdt met het feit dat dB niet lineair evolueert als we verder van een station gaan. Nadeel is dat de ruis ook exponentieel wordt getransformeerd.

VI. Niet-lineaire transformatie: exponentiëel 2

Een alternatief dat we ook uitproberen bestaat erin eerst de niet-lineaire transformatie naar afstand te maken, en vervolgens het verschil in afstand als afstandsmaat te gebruiken.

B. Afstand tussen fingerprint en scan

Na een van deze methodes te hebben toegepast, krijgen we voor elke *fingerprint* een array van afstanden. We moeten nu deze afstanden voor elk station herleiden naar een enkele afstand. Een

transformatie dus van een n-dimensionale vectorruimte naar een 1-dimensionaal geordend veld.

I. Oneindig-norm

De oneindig norm neemt het grootste verschil. Dit zegt echter heel weinig aangezien het zeer sterk beïnvloed kan worden door een enkel station. Ook is het zeer waarschijnlijk dat een beetje ruis zelfs onze metingen in de war kunnen sturen.

II. Gemiddelde van afwijkingen

Het gemiddelde van de afwijkingen zal minder beïnvloed worden door eventuele piekwaarden door omgevingsvariabelen, maar zal door eventuele positieve en negatieve verschillen in de signalen zelfs voor grote verschillen uitgemiddeld worden.

Dit kan opgelost worden door de absolute waarde van deze afwijkingen te nemen (wat dus eigenlijk overeen komt met de 1 norm). Nadeel is hier ook weer dat 1 of enkele relatief grote afwijkingen het volledig resultaat negatief kan beïnvloeden. Het maakt ook niet uit of deze afwijking op grote of kleine afstand gebeurt, alle afwijkingen tellen even hard door.

III. 2-norm

Als we normen nemen, denken we automatisch ook aan de euclidische norm, met deze norm zullen we de grootste afwijkingen ook weer iets meer laten doorwegen, wat zowel goed als slecht zou kunnen zijn.

IV. Gewogen 2-norm

We kunnen bij de 2-norm (en ook de 1-norm) gewichten aan de waarden toekennen, maar dit zou ons in eerste instantie te ver brengen. We zullen een simpele toekenning testen. We vermenigvuldigen elk verschil van signaal tot fingerprint met de gemiddelde waarde van de fingerprint voor dat station, en delen dit door de som van de sterktes van alle stations voor die fingerprint.

3.1.3. Training

Omdat elk groep gebruik maakte van *fingerprints* hebben we onze demonstratie-ruimte nl. het computerlokaal in gebouw 200B opgesplitst in 4 gebieden waar elke groep *fingerprints* ging opmeten. Er werd als concensus voor gekozen om de fingerprints op een discreet rooster te leggen met 1.8 meter afstand ertussen. Op die plaatsen werd telkens in de 4 windrichtingen (N,O,Z & W) een 300-tal metingen gedaan. Ter illustratie tonen we de resultaten voor enkele stations (zie Fig. 8 en Fig. 9). De plattegrond van het lokaal is herkenbaar met de ingang op het punt $(5,0)$.

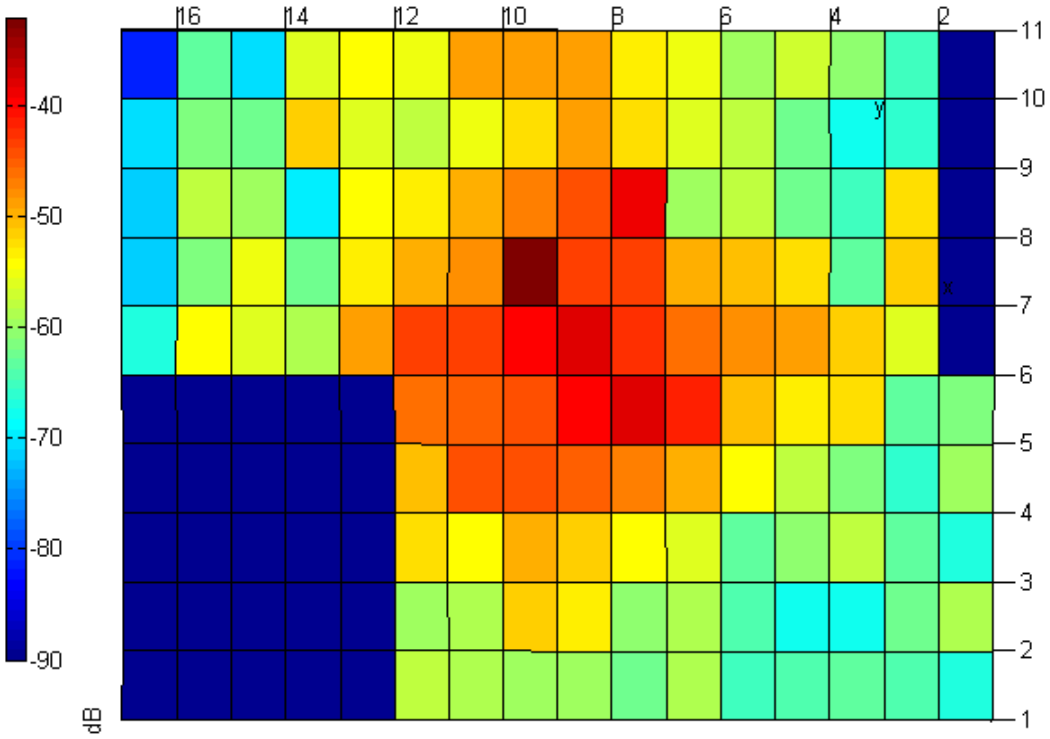


Fig. 8 Signaalsterkte per fingerprint voor station 3 in dB

De signaalsterkte neemt vrij continu af van -40dB tot -90dB en de locatie van het station is goed zichtbaar. De donkerblauwe regio's zijn gebeiden die niet zijn opgemeten.

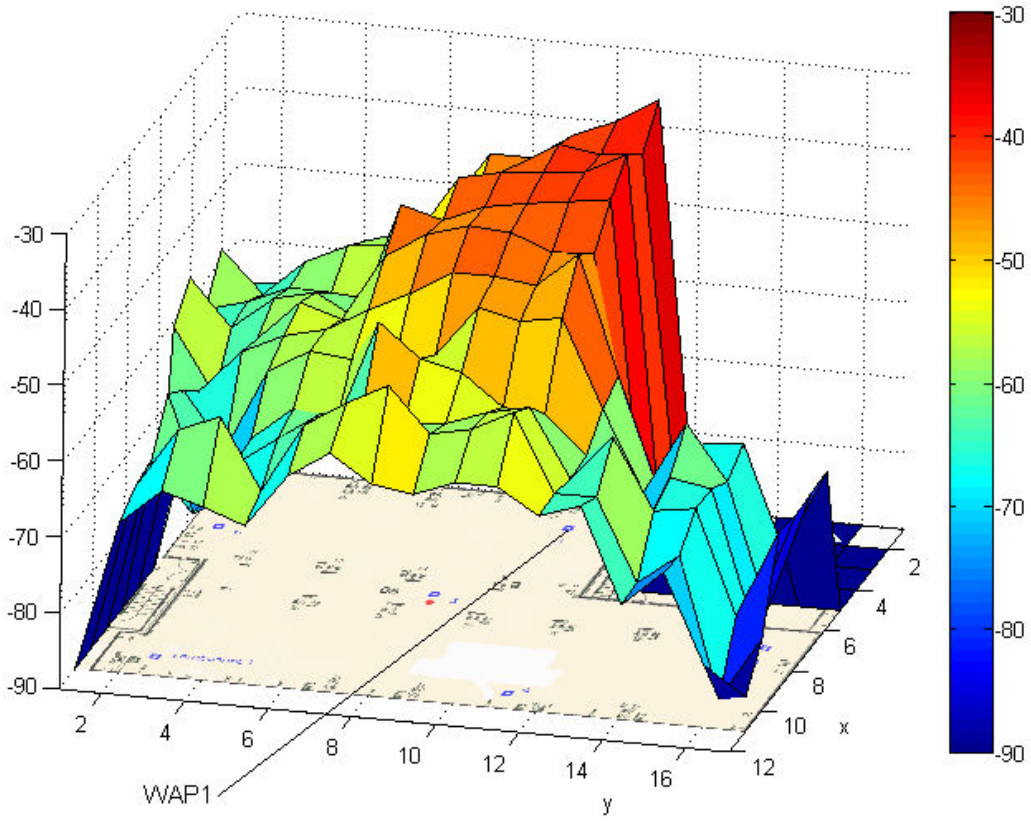


Fig. 9 Signaalsterkte per fingerprint voor station 1 in dB

We hebben ervoor geopteerd om het bekomen oppervlak niet te gaan interpoleren met bijvoorbeeld splines. Dit om zeer lokale discontinuïteiten (als gevolg van resonantie, een paal, een computereiland, …) te behouden. Bovendien loopt het oppervlak meestal zo gelijkmatig dat het ons echt niet nodig leek.

Het leek ons ook niet zinvol om iets als een “PDA-constante” in rekening te brengen die een verschil van PDA tot PDA zou compenseren. Elke groep heeft zijn gebied met een ander toestel opgemeten om dit probleem te kunnen verrekenen. Wij zien echter absoluut geen significant verschil in de vorm van een discontinuïteit op de grenzen tussen de verschillende gebieden. Het lijkt ons een *non-issue*.

3.1.4. Testen

Door de grote hoeveelheid gevonden methodes is het niet eenvoudig om te beslissen welke nu werken en welke niet. Om toch zo objectief mogelijk te kunnen kiezen bedachten we een systeem om alle mogelijke combinaties van methodes met elkaar te vergelijken. Onze methodes zijn immers combinaties van 2 ‘klassen’ van methodes: een eerste om tussen de fingerprint en de scan de afstand te bepalen (7 methodes) en een tweede om de totale afstand tussen de fingerprint en de scan te berekenen (4 methodes).
Om de verschillende methodes te kunnen vergelijken stelden we een 4x7 matrix op met alle mogelijke combinaties. Deze matrix bevat de score van elke combinatie.

We maten 608 fingerprints op en vervolgens namen we de opgemeten data van 3 minuten per fingerprint en splitsten die op in kleinere metingen van 3 à 4 seconden. In die korte metingen is de ruis significant en kunnen we de robuustheid van de methodes hiervoor testen. Dan lieten we alle verschillende methodes los op die stukjes data en berekenden we de afstand tussen de gevonden locatie en de locatie waar de data was opgemeten. Dit geeft ons voor elke methode een totale fout in meter voor 22800 positiebepalingen. Het betreft hier wel een ideale situatie op de dag en het uur wanneer de fingerprints zijn opgemeten. Later zal blijken dat in de praktijk de signaalsterktes niet zo tijdsinvariant zijn (door een verschil in temperatuur, verschil in vochtigheid, verschil in populatie,…)

Totale fout in meter voor alle fingerprints in het ideale geval:								
	Correlatie	Exponentieel 1	Exponentieel 2	Hypothese 1	Hypothese 2	Mean	Median	
Gemiddelde	64415	76477	40119	110188	145778	41032	46359	524368
∞-Norm	71284	97237	47282	48919	61059	47282	56467	429530
2-Norm	67249	86155	42859	100621	143532	41350	48131	529897
Gewogen 2-Norm	67332	87722	43644	104779	148597	41737	48367	542178
	270280	347591	173904	364507	498966	171401	199324	

Tabel 1 Scorematrix totaal

Wanneer we delen door het aantal tests krijgen we een gemiddelde fout in meter. Dit kan dienst doen als een indicatie voor de resolutie van onze plaatsbepaling.

Gemiddelde fout in meter voor alle fingerprints in het ideale geval:								
	Correlatie	Exponentieel 1	Exponentieel 2	Hypothese 1	Hypothese 2	Mean	Median	
Gemiddelde	2,822	3,350	1,757	4,827	6,386	1,797	2,031	5,742
∞-Norm	3,123	4,259	2,071	2,143	2,675	2,071	2,473	4,704
2-Norm	2,946	3,774	1,877	4,408	6,287	1,811	2,108	5,803
Gewogen 2-Norm	2,949	3,843	1,912	4,590	6,509	1,828	2,119	5,937
	2,960	3,806	1,904	3,992	5,464	1,877	2,183	

Tabel 2 Scorematrix gemiddeld

We moeten spijtig genoeg vaststellen dat de meer complexe methodes het niet zo goed doen. De zeer eenvoudige *1-norm van het verschil van gemiddeldes* heeft een gemiddelde fout van 1.79 meter, ongeveer 1 rasterpunt. De enige methode die beter doet is de exponentieël 2 die een niet-lineaire transformatie uitvoert op het verschil in dB en een resolutie van 1.76 meter haalt. Deze berekeningen gebeuren op een totaal van 22800 reeksen metingen en kunnen dus vrij accuraat worden verondersteld. Om zeker te zijn dat er lokaal geen extreme fouten zijn onderzoeken we de gemiddelde fout per fingerprint voor de beste methode (*exponentiëel 2 met de 1-norm*, zie Fig. 10 en Fig. 11).

De fout is lokaal nooit groter dan 4 meter en meestal zelfs rond het gemiddelde van 1.76 meter. Centraal in het gebouw waar er in alle richtingen meer stations beschikbaar zijn is de fout zeer klein, tot gemiddelde 0.5 meter. Aan de randen en hoeken van het gebied durft de fout al eens groot te worden.

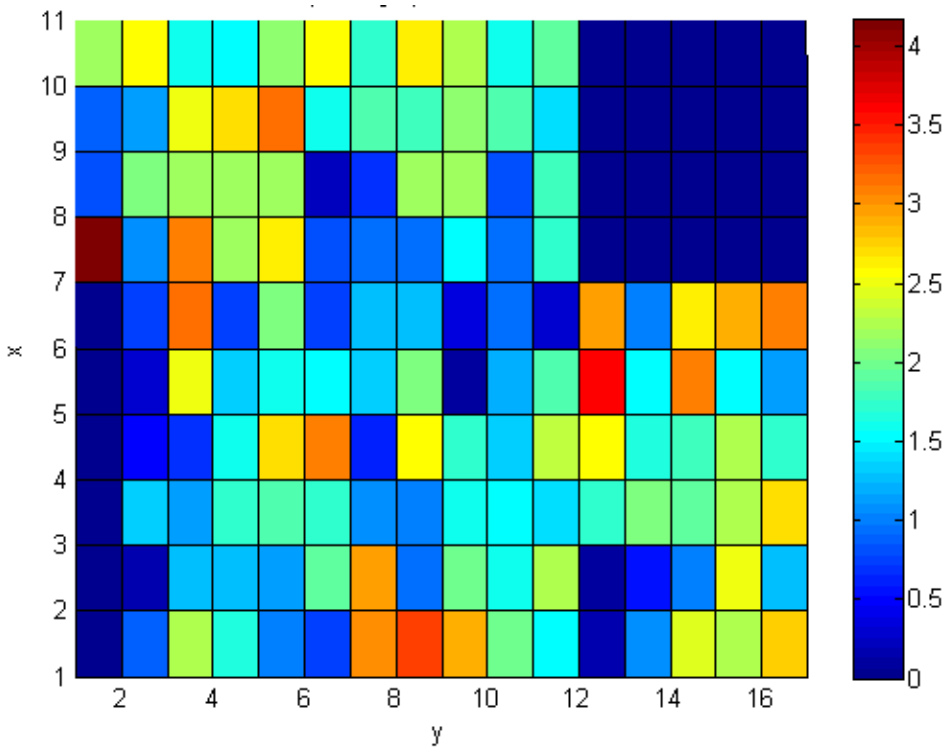


Fig. 10 Ideale fout per fingeprint voor *exponentiëel 2 met de 1-norm*

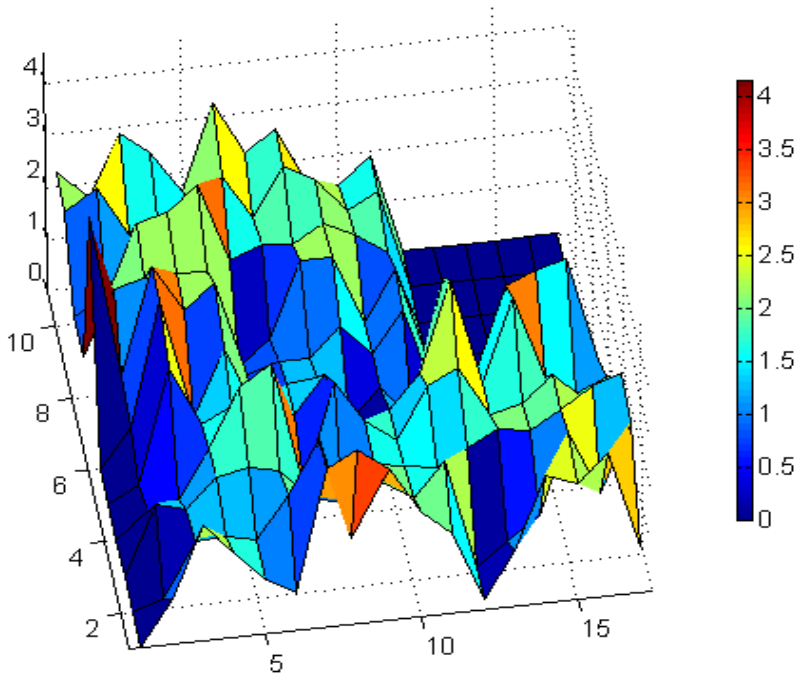


Fig. 11 Ideale fout per fingerprint voor *exponentieel 2 met de 1-norm* (in 3D)

Deze tests gaan er van uit dat de totale situatie sinds het opmeten van de *fingerprints* niet meer sterk veranderd is. In de praktijk blijkt deze eenvoudige aanpak echter te naïef. Het is noodzakelijk om gemiddeld gezien resolutie in te boeten om extreme uitschieters te kunnen opvangen. Daarom gebruiken we naast de *1-norm van het verschil van gemiddeldes* ook de *k-nearest-neighbour*-methode.

We onderzoeken op fig.12 de theoretische fout voor de 1-norm van het verschil van de gemiddeldes voor toenemende k (het aantal beste benaderingen waar we mee rekening houden). Voor k gelijk aan 1 herkennen we de 1.79 meter van tabel 2. Voor toenemende k wordt zoals verwacht de fout in het ideale geval groter. Voor onze toepassing kiezen we k gelijk aan 5 met een gemiddelde fout van 2.5 meter, maar met een goede robuustheid voor uitschieters.

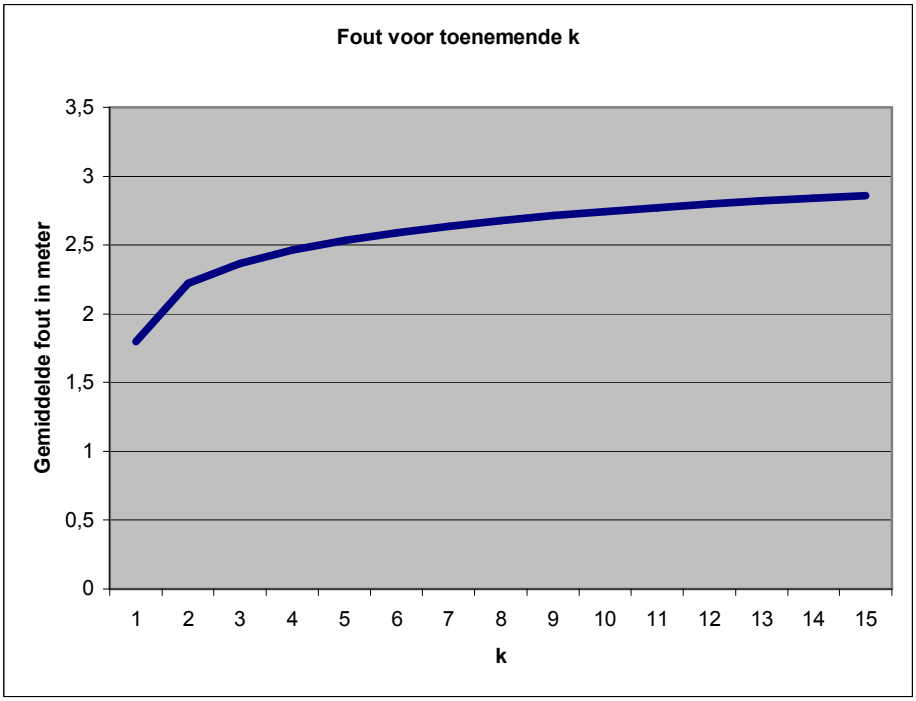


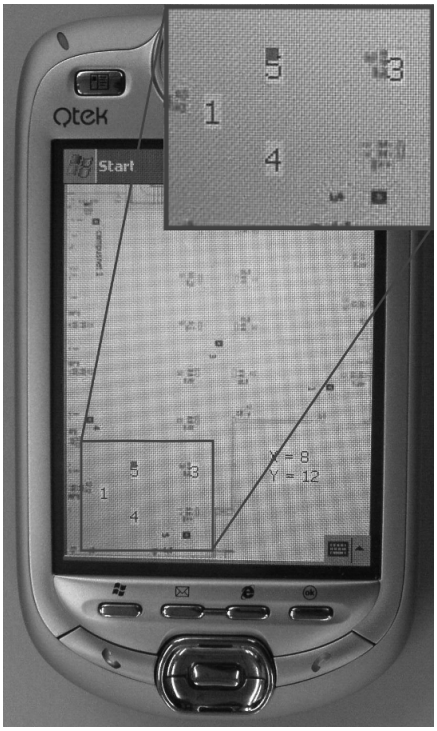
Fig. 12 Ideale fout voor mean difference *met de 1-norm* voor toenemende k

3.1.5. Besluit

In onze toepassing gebruiken we het verschil van de gemiddelden om de afstand tussen een fingerprint en een scan voor 1 enkel station en het gemiddelde van de afwijkingen om de totale afstand tussen een fingerprint en een scan te berekenen, omdat in de vorige paragraaf bleek dat deze combinatie het beste resultaat levert en de ruis niet exponentieel vergroot.

Het voordeel van de *k-nearest-neighbour*-methode is dat ze veel accurater is dan enkel de dichtstbijzijnde fingerprint nemen, omdat door ruis en dergelijke de dichtstbijzijnde fingerprint (zeer) ver van de eigenlijke positie kan vallen. Als men de k-dichtstbijzijnde neemt echter, zal een eventuele uitschieter minder invloed hebben.

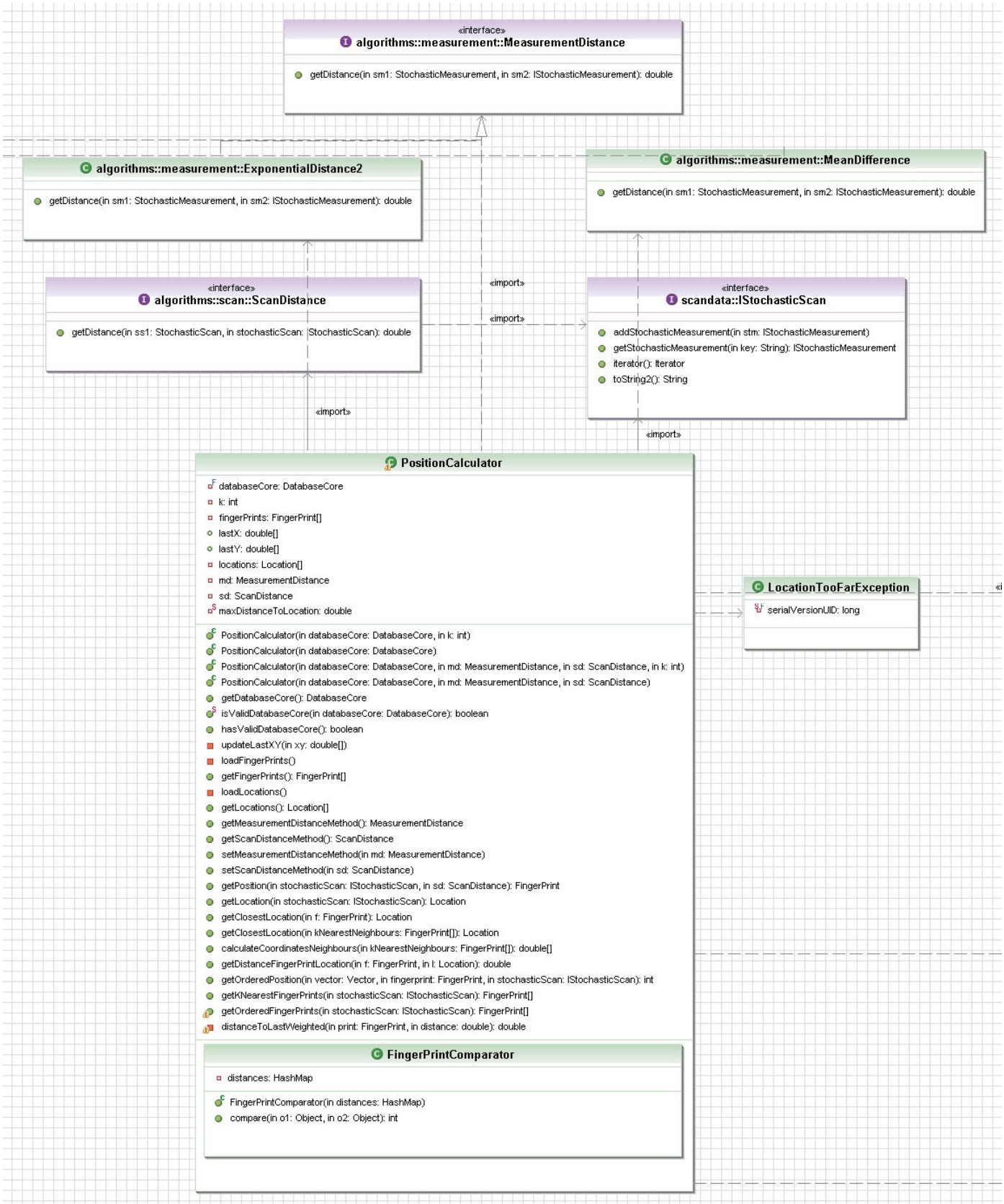
Om de *k-nearest-neighbour*-methode grafisch te illustreren, schreven we een applicatie die het grondplan van het lokaal weergeeft op de PDA met daarop de k-dichtstbijzijnde fingerprints, aangeduid met een nummer, en de berekende positie, aangeduid met een rood vierkantje. De applicatie neemt telkens 7 metingen samen en berekent aan de hand daarvan telkens de k-dichtstbijzijnde fingerprints en de positie en beeldt deze af op het grondplan (zie onderstaande figuur). De coördinaten van de berekende positie worden ook apart weergegeven rechts onderaan.



Via deze applicatie kan men beter verstaan hoe de k-*nearest-neighbour*-methode werkt, aangezien in het uiteindelijke project de gebruiker nooit rechtstreeks te maken krijgt met coördinaten of fingerprints. Ook kan visueel afgeleid worden hoe accuraat de k-*nearest-neighbour*-methode ruwweg is: er is meestal een fout van 1 tot 3 roosterpunten, met soms een uitschieter.

Een volgende stap in het verbeteren van de plaatsbepaling is rekening houden met de vorige berekende plaatsen. Omdat dit voor onze huidige applicatie niet nodig is hebben we er nog niet veel aandacht aan besteed. Toch hebben we naar volgend semester toe al een implementatie voorzien die rekening houdt met het vorige punt. Uit tests met de grafische weergave (mbv van *MapGUI*, zie 3.2.2.)met dit algoritme blijkt dat het zeer goed presteert en een zichtbare verbetering is ten opzichte van enkel *k-nearest-neighbour*-methode.

3.1.6. UML-diagram



3.1.7. Evaluatie

Het harde werken heeft zijn vruchten afgeworpen. De bekomen nauwkeurigheid van de positiebepaling is volgens de eisen.

Het regelmatig crashen van het programma *Raki* en batterij-problemen met de PDA's hebben aanzienlijke vertragingen veroorzaakt.

3.1.8. Vooruitzicht

Volgens semester kan de *k-nearest-neighbour*-methode die rekening houdt met de vorige locatie zeker van pas komen. Bedoeling is om deze methode nog te verbeteren en/of nog betere technieken toe te passen.

3.1.9. Bronnen

- *Accuracy Characterization for Metropolitan-scale Wi-Fi Localization*; Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca & John Krumm; januari 2005; http://placelab.intel-research.net/publications/pubs/IRS-TR-05-003.pdf

- *Large-Scale Localization from Wireless Signal Strength*; Julia Letchner, Dieter Fox & Anthony LaMarca; 2005; http://www.cs.washington.edu/homes/letchner/Papers/wifi-location.pdf

- *Hybrid Method for Localization Using WLAN*; Binghao Li, Andrew Dempster, Chris Rizos & Joel Barnes; 2006; http://sar.informatik.hu-berlin.de/teaching/2006-s%20Interplanetary%20Internet%20Seminar/papers/2_Hybrid%20Method%20for%20Localization%20Using%20WLAN.pdf

- *Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks*; Tsung-Nan Lin & Po-Chiang Lin; ??; http://140.112.41.70/G/2004%5B1%5D.09.06.doc

- *Statistiek en Wetenschap*; Jan Beirlant, Goedele Dierckx & Mia Hubert; derde, herwerkte uitgave 2005; ACCO; ISBN 90-334-6066-1

- *Inleiding tot de numerieke wiskunde, Deel 1: directe methodes*; A. Bultheel; tweede, herzien uitgave 1993; L. Wouters

3.2. Client

Team: *Daniel*

Omvat: *UserGUI* en *ClientCore*

3.2.1. UserGUI

Bij het nadenken over het ontwerp van een GUI, is het belangrijk dat men zich eerst goed afvraagt voor wie de applicatie bestemd is, en wat die mensen er mee moeten bereiken. Onze toepassing zal gebruikt worden door leerlingen van het lager (en eventueel ook middelbaar) onderwijs. De studenten moeten bv. tijdens een museumbezoek een aantal vragen oplossen op bepaalde plaatsen (LOI's).

Het is de bedoeling dat een student of groep studenten een PDA krijgt, en rond gaat bv. in een museum. Voor elke LOI zijn er een aantal vragen.

De PDA moet een zekere functionaliteit aanbieden aan zijn gebruiker. De gebruiker moet een vragenlijst kunnen opvragen voor een bepaalde LOI. De vragen moet hij kunnen oplossen, en doorsturen naar de server. Ook is het handig als de gebruiker kan opvragen hoe veel LOI's hij al heeft bezocht. Al snel volgde de eerste iteratie van de uiteindelijke GUI (zie Fig. 12).

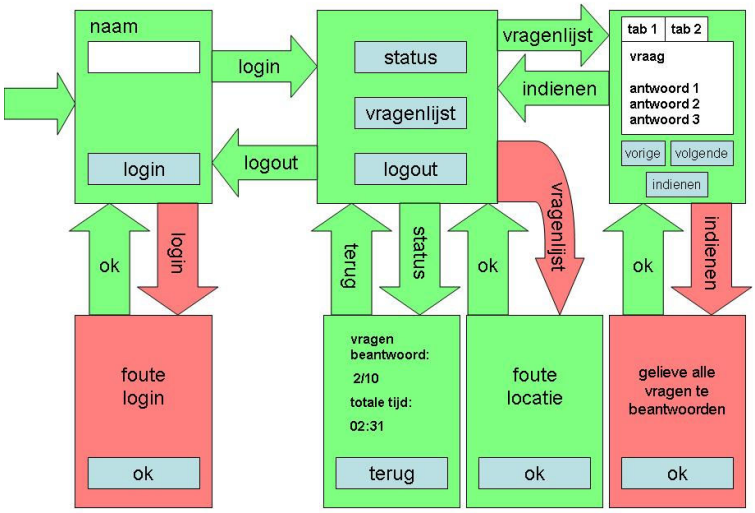


Fig. 12 Eerste iteratie van GUI

Om het tijdelijk concept uit te testen moest er een evaluatie volgen met proefpersonen rond de juiste leeftijdscategorie. Vooral het intuïtief zijn van de GUI was van groot belang. Het is niet de bedoeling dat de leerlingen eerst een les krijgen over hoe ze de toepassing moeten gebruiken.

Om de GUI te evalueren werd er een papieren versie voorgesteld aan een aantal onafhankelijke studenten van het lager onderwijs. Aan elk

van de proefpersonen werd uitgelegd wat de bedoeling nu feitelijk was: namelijk dat ze met een PDA een museum rond moesten gaan, en dat ze vragen moesten beantwoorden over de schilderijen. Verder werd er van hen verwacht dat ze zouden vertellen wat ze in elk gegeven situatie zouden doen.

Eerst werd er dus een blaadje met het loginscherm getoond. Verder werd er (afhankelijk van hun gekozen actie) een ander, passend schermpje getoond. Aangezien er werd gestreefd naar een intuïtief design, werden alle misinterpretaties en verwarringen genoteerd voor gebruik bij een nieuwe iteratie (zie Tabel 3).

Proefpersoon 1	Proefpersoon 2	Proefpersoon 3
- Begreep de bedoeling aanvankelijk niet van het hoofdscherm. (Het moest eerst uitgelegd worden) - Verwardheid rond de tabstructuur in het vragenlijst.	- Begreep de bedoeling van het hoofdscherm niet. - Logout knop bleef ongebruikt	- Begreep de bedoeling van het hoofdscherm niet. - Vond status knop niet nodig. (Hij kon zelf wel onthouden welke schilderijen hij had bezocht.) - Verwardheid rond wanneer men moest uitloggen.

Tabel 3 Proefpersonen

Al snel bleek dat er enige onduidelijkheden waren met het ontwerp. De drie knoppen status, vragenlijst en logout in het hoofdmenu waren niet zo duidelijk. Ook de vragenlijst zelf was even wennen. Blijkbaar was de combinatie van tab-structuur (1 tab per vraag) met nog een vorige en volgende knop er te veel aan.

In het hoofdmenu moest men zich afvragen of de knoppen status en logout zelfs nodig waren, aangezien men ze niet zo nodig had. De belangrijkste informatie van status (namelijk het aantal vragen beantwoord) kon even goed op het hoofdmenu zelf worden getoond. De proefpersonen maakten enkel gebruik van de logout knop, wanneer ze alle vragen hadden opgelost (of nooit, in het geval van proefpersoon 2). Dit betekende dat ook die knop niet nodig was, aangezien het kon worden geautomatiseerd.
In de vragenlijst zelf bleek, dat het niet nodig was om terug te gaan naar een vorige vraag. Bladeren doorheen de vragen maakte het overzicht enkel onduidelijker. Er moest dringend een simpelere variant op komen (met gereduceerde functionaliteit als compromis).
Het resultaat was een veel efficiënter grafische interface, dat aangenamer was om te besturen, vanwege de simpelheid (zie Fig. 13).

Om voor verdere evaluatie te zorgen, moest de nieuwe iteratie getest worden met dezelfde proefpersonen. Deze keer was de GUI al geïmplementeerd, en kon er op de PDA zelf geëvalueerd worden. De resultaten waren al wat positiever. Wel was er nog een verwarring ontstaan bij het aanklikken van sommige knoppen. Er is namelijk een vertraging bij het overgaan van één scherm naar een ander. Afhankelijk van hoe snel de server gegevens kan verwerken kan dit soms enkele seconde duren.

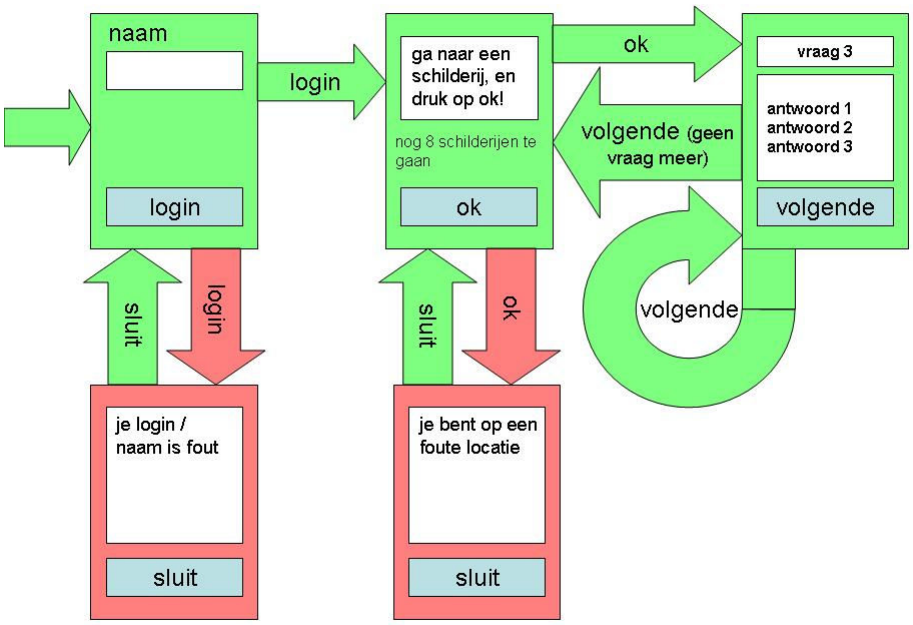


Fig. 13 Tweede iteratie van GUI

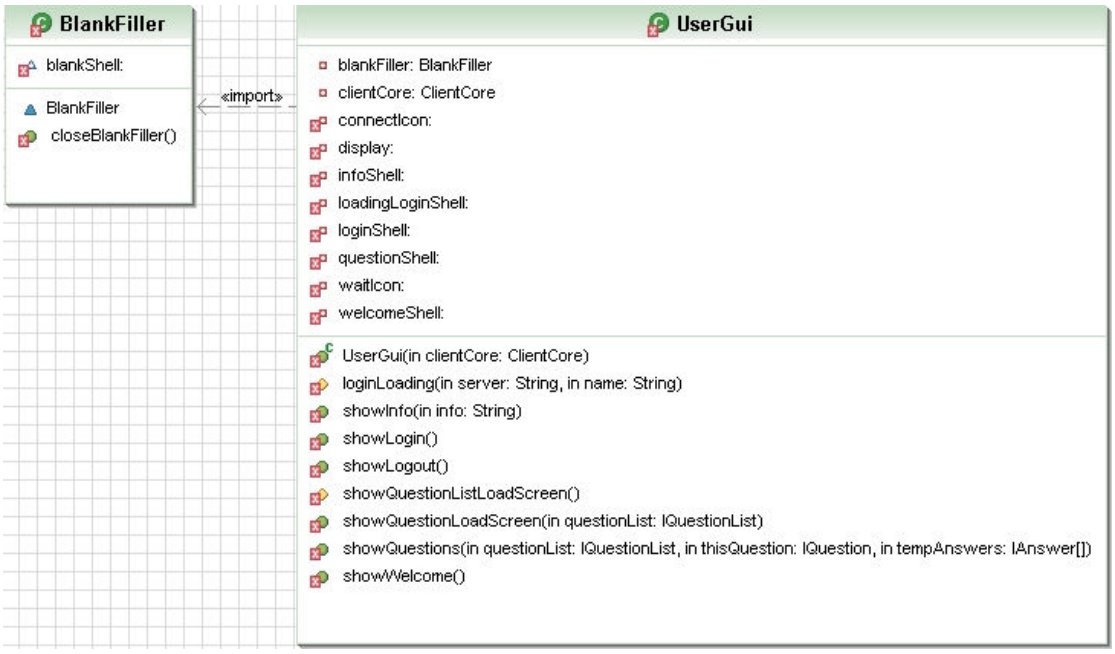
Om ervoor te zorgen dat de proefpersonen niet dachten dat het programma was vastgelopen, werden er nog laadschermen geïmplementeerd tussen verschillende vensters in. Als men nu op “ok” klikte om een vragenlijst te krijgen, kreeg men dus eerst een scherm met het boodschap om eventjes te wachten.

3.2.2. MapGUI

Buiten de standaard GUI moest er ook nog een speciale applicatie worden gemaakt, om de plaatsbepaling te testen en te demonstreren (zie 3.1.5.). De *MapGui* maakt gebruikt van SWT om een plattegrond te tonen van het lokaal, dat heel het scherm vult. Een rood stipje geeft aan waar de gebruiker zich bevindt. De cijfers 1 tot 4 (die ook op het schermpje worden getoond) geven 4 verschillende *fingerprints* aan. Het laatste cijfer (5) geeft de vorige positie weer. Telkens wanneer er nieuwe metingen binnenkomen, worden alle punten opnieuw getekend. Als gevolg kan men dus in “real-time” doorheen het lokaal om te kijken hoe de plaatsbepaling werkt.

Aangezien deze applicatie niet bij de rest van het programma hoort, werd het niet geëvalueerd door doelgebruikers.

3.2.3. UML-diagram



3.2.4. Evaluatie

Het testen op proefpersonen kan zeer nuttig blijken. Wanneer je zelf bezig bent met een GUI, besef je zelf niet meer of het nu duidelijk of onduidelijk is.

SWT gaf ook een beetje problemen omdat het anders reageerde op de PDA dan op de PC.

De GUI (en ook MapGUI) was volgens de eisen.

3.2.5. Vooruitzicht

Volgend semester zal er meer worden geëvalueerd. Dat zal ook nodig zijn, aangezien de GUI een stuk ingewikkelder gaat worden. Gelukkig is de ervaring met SWT er al.

3.3. Communication

Team: *Wouter*

Omvat: *Communication* en de bijhorende communicatie tussen PDA en server via RMI

Om communicatie tussen twee Java applicaties te verzorgen, zijn er verschillende mogelijkheden: *SOAP, XML RPC, RMI, CORBA* met *IDL* en *DCOM*. Een overzicht.

Positieve punten	***Negatieve punten***
SOAP	
Platformonafhankelijk	Zelf gedefinieerde objecten zijn **moeilijk** door te geven (enkel klassen van de JDK): deze moeten zelf geserialiseerd worden.
XML RPC	
Eenvoudig	Er kunnen **geen** zelfgedefiniëerde objecten doorgegeven worden.
Platformonafhankelijk	
RMI	
Objecten zoals lokaal	Enkel Java
Platformonafhankelijk	
CORBA met IDL	
Professioneel	(Te) veel mogelijkheden
	Ingewikkeld
DCOM	
Verouderde technologie	Overkill
	Te ingewikkeld
	Enkel Microsoft

Ons ontwerp voorzag dat objecten op de PDA moeten kunnen aangeroepen worden zoals lokale objecten. Dit is eveneens mogelijk met *RMI*, *XML RPC* en *SOAP*, maar bij deze laatste 2 kunnen respectievelijk geen en pas na eigen serialisatie objecten doorgestuurd worden. Dit is een extra moeilijkheid en dus te vermijden. De keuze viel dus op *RMI*.
Een probleem bij de implementatie van *RMI* is dat het object dat de server ter beschikking stelt, hetzelfde is voor alle clients. Aangezien de server dan niet weet door welke PDA een methode wordt opgeroepen, werd het *Factory Design Pattern* geïmplementeerd (zie Fig. 14). De klasse die de PDA's krijgen als ze verbinden met de server (de klasse ObjectFactory), wordt gebruikt om aan te melden. De PDA roept in die klasse een methode op met als argument de naam van de gebruiker. Op de server wordt deze informatie gecontroleerd, en als de gebruiker mag

inloggen, wordt een nieuw object gemaakt op de server (een object van de klasse User), en deze dient als return waarde van de loginprocedure.

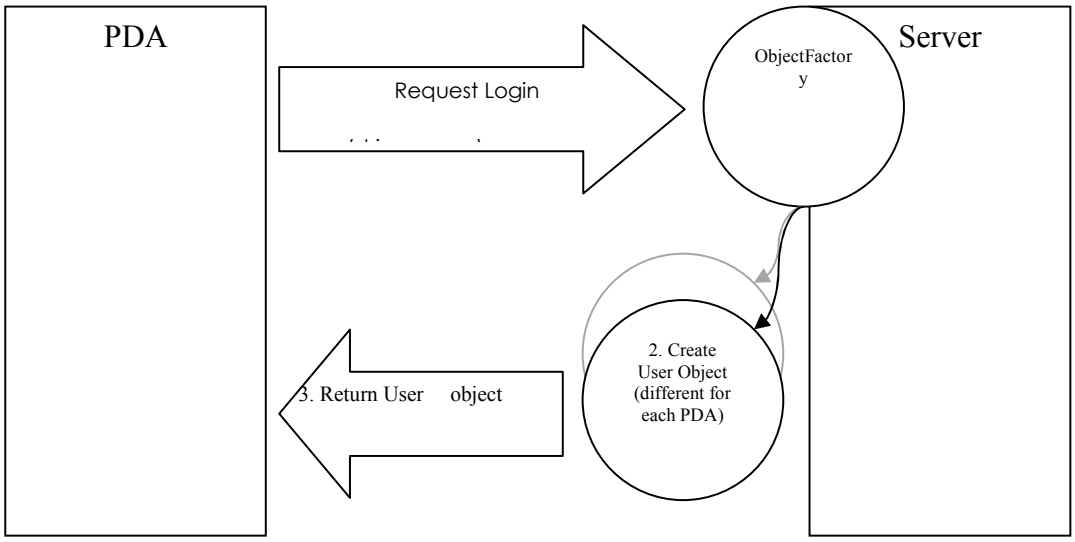
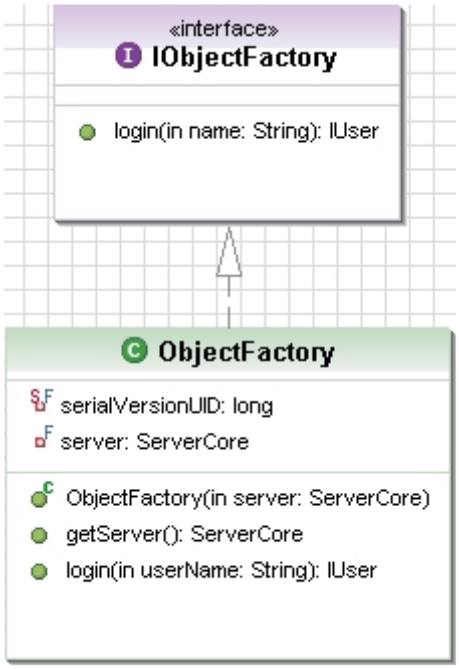


Fig. 14 Factory Design Pattern

De code van voor een eenvoudig RMI voorbeeld was te vinden op http://www.comp.hkbu.edu.hk/~jng/comp3320/rmi.html. Van hieraf werd er verder gebouwd, met het klassediagramma in het achterhoofd, tot volwaardige *ServerCore* en *ClientCore* objecten. Deze zijn de onderliggende laag die door de respectievelijke GUI's zal aangesproken worden.
RMI voor het eerst aan de praat krijgen in een groot project was geen simpele klus en zorgde dan ook voor de nodige problemen gedurende de eerste programmeersessies: ofwel konden de stubs niet aangemaakt worden, ofwel werden de stubs door Java niet gevonden. De oplossing bestond erin de rmiregistry in de classpath te starten.
Het project (client en serverapplicatie) werd achtereenvolgens getest op dezelfde computer, op verschillende computers (om SSH tunnels te leren kennen) en uiteindelijk op PDA. In deze laatste fase, die pas van kracht kon gaan begin december omdat er nog geen tunnelapplicatie beschikbaar was, kwamen verscheidene mankementen in de GUI en communicatie aan het licht.
Vooral *BindExceptions* en *NoRouteToHostExceptions* kwamen veel voor, en werden uiteindelijk teruggebracht tot het feit dat zowel de server als PDA objecten ter beschikking willen stellen via RMI (bijvoorbeeld een lijst met vragen door de server, en het antwoord daarop door de PDA). Een echte oplossing hierop werd niet gevonden, maar een *workaround* werkte wel: enkel op de server objecten aanmaken die door RMI gebruikt zullen worden. Praktisch betekende dit bijvoorbeeld dat de PDA vraagt aan de server om een nieuw antwoordobject aan te maken, dat vervolgens door de PDA van het daadwerkelijk ingegeven antwoord voorzien wordt, en dan aan een controlemechanisme op de server wordt doorgegeven.
Omdat RMI blijkbaar toch moeilijker en koppiger is dan eerst gedacht, wordt eraan gedacht om volgend semester op SOAP over te schakelen. Na analyse van de code is gebleken dat de communicatie op eenvoudige wijze enkel met klassen van de JDK (strings, arrays, ints)

kan gebeuren, waardoor het nadeel van SOAP dat gebruikersgedefinieerde klassen serialiseerbaar gemaakt moeten worden waarschijnlijk wegvalt. Ook wordt het gebruik van SSH tunnels overbodig, omdat SOAP eenvoudigweg over HTTPS werkt. Het gebruik van PockeTTY zorgt bij iedere test op PDA voor een zekere vertraging (extra instellingen, …).

3.3.1. UML-diagram



3.3.2. Evaluatie

RMI blijkt een moeilijke keuze te zijn geweest, aangezien RMI gedurende lange tijd niet aan de praat wilde. Dit heeft geleid tot een aantal minder elegante beslissingen.

3.3.3. Vooruitzicht

Volgend semester RMI aan de praat krijgen op elegante wijze. Het regelmatiger integreren van de onderdelen wordt een belangrijker punt.

3.4. Server

Team: *Philippe*

Omvat: ServerCore, *DatabaseCore* en de database zelf.

3.4.1. Database

De database bestaat uit 7 tables die gevisualiseerd zijn in Fig. 15. De belangrijkste table is FINGERPRINTS. Hierin worden de fingerprints opgeslagen die gebruikt worden om de positie te bepalen. Elke fingerprint bestaat uit een ongedefinieerd aantal measurements, die worden opgeslagen in de FP_MEASUREMENTS-table met de nodige informatie.

Een interessante locatie wordt opgeslagen met behulp van coordinaten zodat de plaatsbepaling kan bepalen welke locatie het dichtst bij de positie ligt. Deze locaties worden bijgehouden in de table LOCATIONS. Aan een locatie wordt dan een aantal vragen gelinkt via de QUESTIONS-table. Elke vraag bevat een aantal antwoorden die worden opgeslagen in de ANSWERS-table. Indien een locatie wordt verwijderd, worden de vragen gelinkt aan deze locatie ook automatisch verwijderd door de cascade van de foreign key. Hetzelfde geldt voor de antwoorden als er een vraag verwijderd wordt.

Als laatste zijn er nog twee tables USER en USER_ANSWERS. In USER worden de gebruikers. In USER_ANSWERS worden de antwoorden van de verschillende users opgeslagen. Indien een gebruiker of beheerder wordt verwijderd, dan worden ook automatisch zijn/haar antwoorden verwijderd. Indien een vraag wordt verwijderd, dan worden ook alle records in de USER_ANSWERS-table verwijderd die naar die vraag verwijzen.

3.4.2. DatabaseCore

Een *DatabaseCore*-object bevat informatie over de server en de login gegevens, waarmee een connectie kan aangemaakt worden binnen dit object (zie UML-diagram). Deze connectie kan dan gebruikt worden om de database aan te spreken. Er zijn een aantal belangrijke functies voorzien die noodzakelijk zijn voor de goede werking van onze toepassing.

Belangrijke functies zijn die die instaan voor het teruggeven van alle fingerprints en locaties (*getFingerprints()* en *getLocations()*). Daarnaast zijn er nog de functies om een gebruiker aan te melden, vragen op te vragen en antwoorden op te slaan die worden gegeven door de gebruikers. Er zijn ook enkele functies geimplementeerd om informatie terug te geven over een bepaalde gebruiker, alsook om te bepalen of een gebruiker al op een bepaalde locatie is geweest en om te controleren of een gebruiker alle locaties al bezocht heeft.

De uiteindelijke bedoeling van dit object is dat het aangemaakt wordt door *ServerCore* en dat zowel de *ServerCore* als *PositioningSystem* dat object kunnen gebruiken om de nodige zaken te laden en op te slaan in de database.

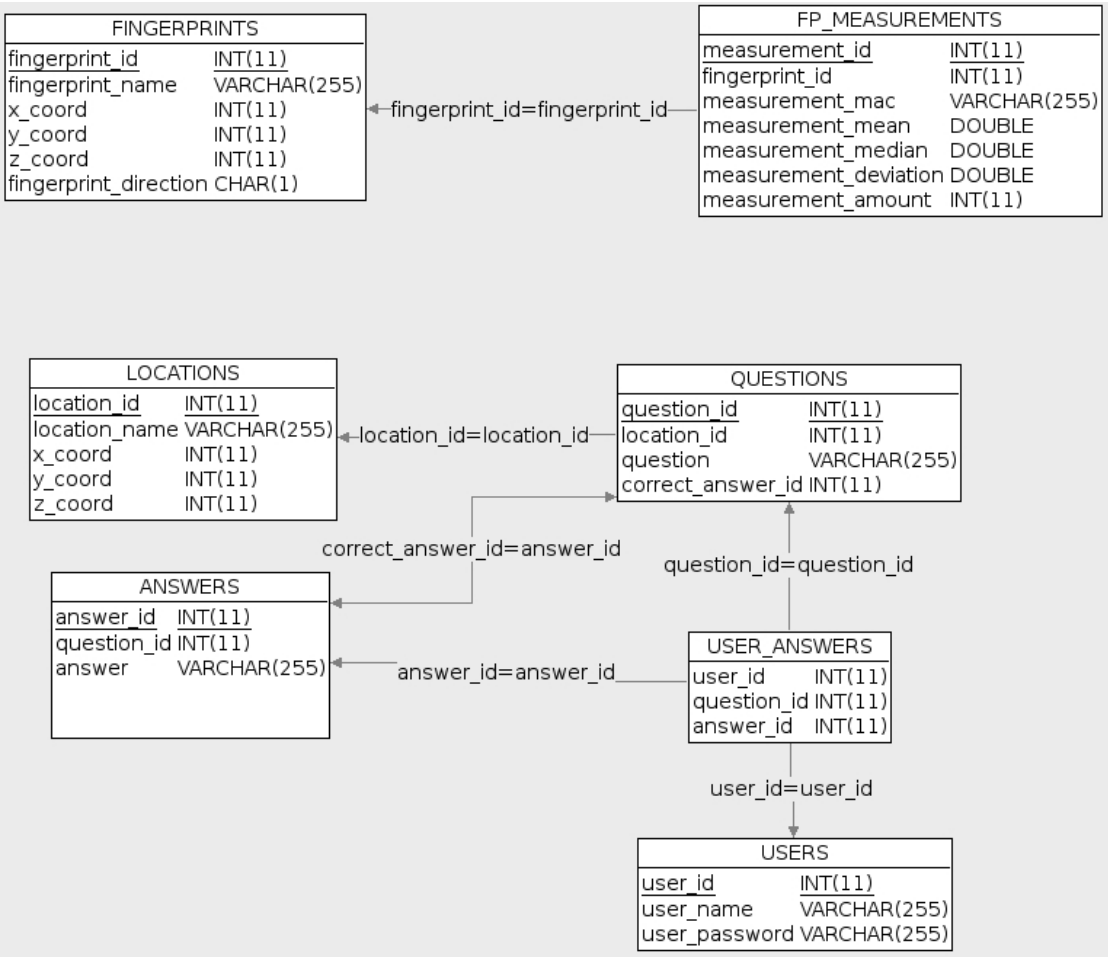
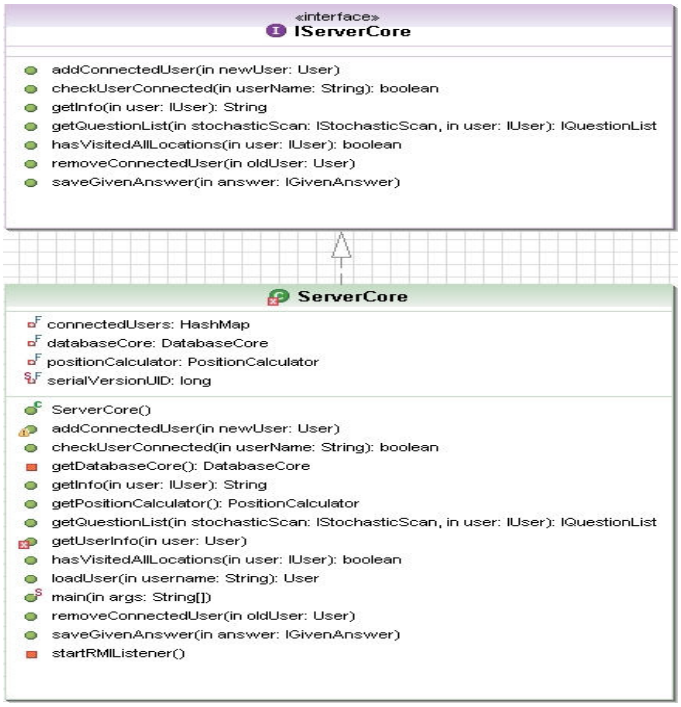


Fig. 15 Database-diagram

3.4.3. UML-diagram



3.4.4. Evaluatie

Database werkte perfect.

3.4.5. Database

Volgend semester zal de database iets uitgebreider worden. Dit zou geen belangrijke problemen mogen geven.

4. Team

We bespreken nu de samenwerking in team.

4.1. Peer-assesment

Iedereen van het team heeft elkaar en zichzelf anoniem beoordeeld met volgende codes die duiden op de bijdrage van die persoon tot het project (zie Tabel 4).

- + = meer dan gemiddelde bijdrage
- 0 = gemiddelde bijdrage
- - = minder dan gemiddelde bijdrage

Cedric	Chris	Daniel	Jef	Guy	Philippe	Wouter
+	+	0	-	+	0	0
0	+	0	0	+	+	0
+	+	0	+	+	+	+
0	+	0	-	+	0	0
0	0	+	0	+	0	+
+	+	0	0	0	0	0
0	+	0	+	0	0	+
3	**6**	**1**	**0**	**5**	**2**	**3**

Tabel 2 Peer-assesment

De onderste rij is de som van de bovenste rijen voor die persoon met volgende conversie (+ → +1; 0 → 0; - → -1).

4.2. Werkbelasting

Zie achteraan.

4.3. Evaluatie

Iedereen in het team was goed gemotiveerd en er hing een goede sfeer. Communicatie in het team gebeurde snel en vlot, zeker dankzij de start/stopvergadering bij elke zitting zodat iedereen goed op de hoogte bleef en zijn taken wist. Sommige individuen hebben het al iets moeilijker dan anderen om in team te werken en te denken.

Het hele team of subteams heeft ervaring opgedaan met de betrokken onderdelen (positiebepaling, RMI, SWT, ..) en het werken aan software met een redelijk aantal mensen.

4.4. Vooruitzicht

Het probleem dat sommige subteams toch nog moesten wachten op andere subteams kan verholpen worden door een nog betere planning te hanteren.

Aangezien het goede resultaat en de tevredenheid van het team gaan we het in het 2de semester zeer gelijkaardig aanpakken. Het proberen van meer systematisch te integreren maken we wel een belangrijker punt.