

# **IP5: Lokalisierung in drahtlosen Netzwerken ohne GPS**

## **Technische Dokumentation**

Martin Häfelfinger

Fabiano Marraffino

## Inhalt

IP5: Lokalisierung in drahtlosen Netzwerken ohne GPS .....	1
Technische Dokumentation .....	1
Komponenten.....	3
Software Architektur .....	4
Localizer.....	5
Localizer.Calculator .....	8
Commons.....	8
Commons.Location.....	11
Commons.WLAN .....	12
Datamining .....	13
Persistence .....	13
Positionsbestimmung mit WLAN.....	15
Referenzdatenbank .....	15
Datenverwaltung.....	15
Ermitteln der nächsten Nachbarn .....	16
<b>Positionsbestimmung</b> .....	17
Datenstruktur Grid .....	18
Analyse der WLAN Lokalisierung.....	19
Test-Aufbau .....	19
Vergleich der verschiedenen Nearest Neighbors Algorithmen.....	20
Ergebnisse .....	21
Visualisierung .....	21
Glossar .....	25

## Komponenten

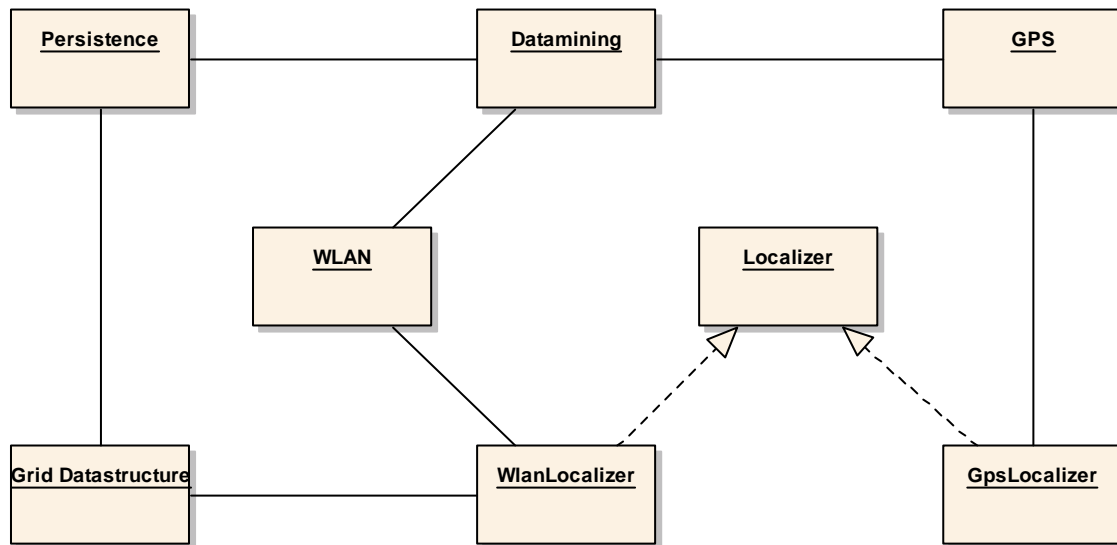


Abbildung 1: Komponenten und ihre Beziehungen

Ein *Localizer* wird benutzt um die momentane Position zu bestimmen. Für die API haben wir zwei Varianten entwickelt. Eine beruht auf *GPS* und die andere benutzt *WLAN*. Die API erlaubt es jedoch, weitere *Localizer* zu entwickeln und zu integrieren. Der *GPSLocalizer* benötigt ausschliesslich einen Gps-Empfänger, welcher erlaubt Satellitendaten zu empfangen und auszuwerten. Der *WLANLocalizer* benötigt eine Funknetz Karte und eine bereits bestehende Referenzdatenbank. Um diese zu erstellen, müssen die Positionen, welche später über *WLAN* ermittelt werden sollen, mit ihren *GPS*-Koordinaten und allen an diesem Punkt sichtbaren *Accesspoints* mit Hilfe des *DataMiner* gespeichert werden. Neben dem eindeutigen Namen des *Accesspoints* (die *Mac-Adresse*) wird auch dessen Signalstärke an dieser Position erfasst. Mit der *Grid Datenstruktur* werden die Referenzpunkte in einem Gitternetz (mit Breiten- und Längengrad als Koordinaten) eingelesen und dem *WLANLocalizer* zugänglich gemacht.

## Software Architektur

Die verschiedenen Aufgabenbereiche wurden auf verschiedene Packages aufgeteilt. Als Präfix geht dabei allen „Ch.Fhnw.MobileLocalization.“ voran.

Commons: Beinhaltet Klassen, welche von verschiedenen anderen Komponenten benötigt wird (u.A. *ReferencePoint* und eine Fabrik zum erzeugen verschiedener Objekte)

Commons.Location: Enthält Definitionen für einen Punkt auf der Erde (Koordinaten usw.)

Common.WLAN: Scannen und Erfassen von Drahtlosnetzwerken.

Datamining: Beinhaltet den *DataMiner*, welcher verantwortlich für das erfassen von Referenzdaten ist.

Localizer: Klassen zum Bestimmen der Position. Hier ist auch der Haupt-Lokalisierer *MobileLocalizer*

Localizer.Calculator: Strategien zur Berechnung der aktuellen Position

Localizer.Datastructure: Alles rund um die Grid-Datenstruktur zum speichern von Referenzpunkten

Persistence: Klassen zum schreiben und lesen von persistenten Daten

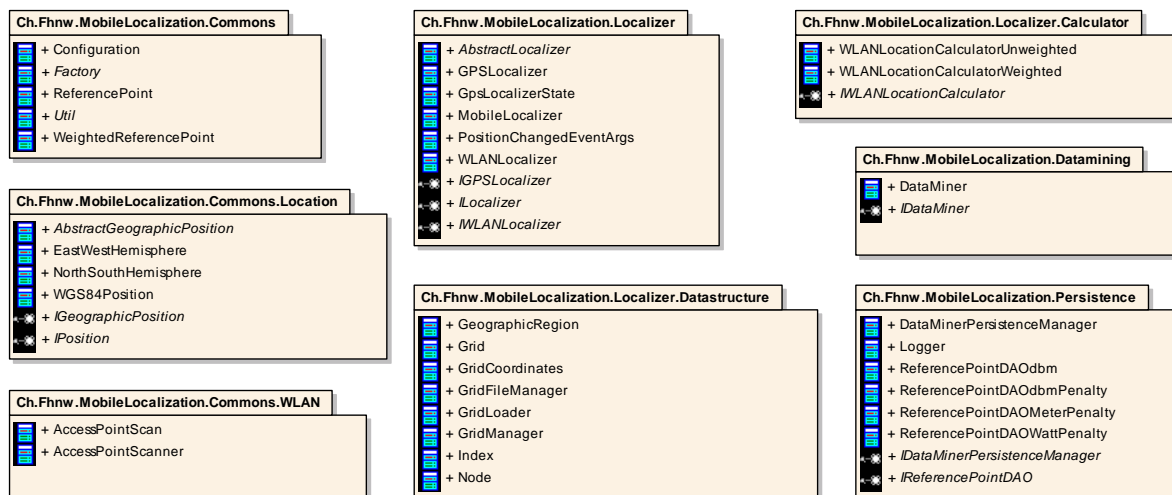


Abbildung 2: Packageübersicht und die Enthaltenen Interfaces und Klassen

## Localizer

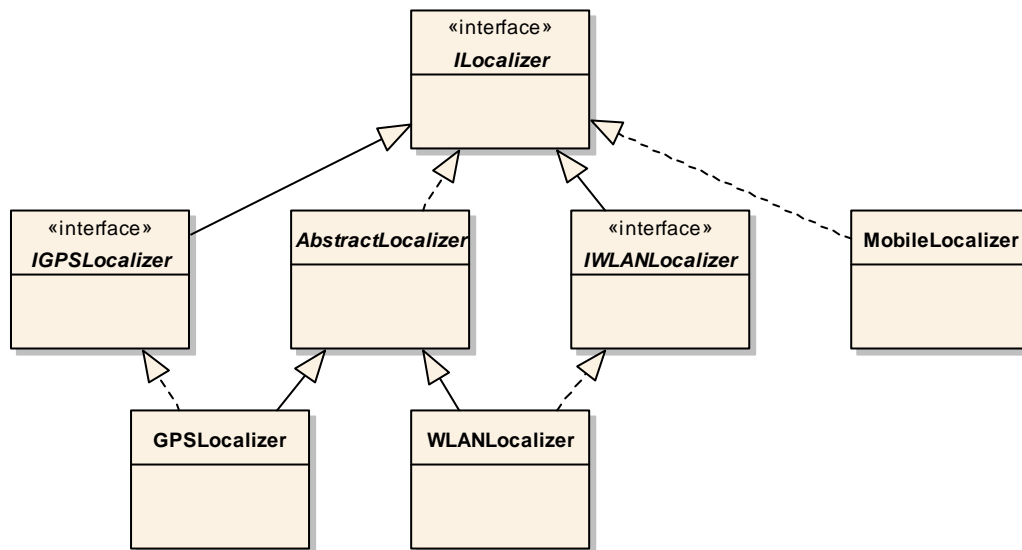


Abbildung 3: Klassenhierarchie der Localizer

### ILocalizer

Alle Varianten um eine Position zu bestimmen implementieren das Interface *ILocalizer*. Die Schnittstelle bietet eine Methode zur Bestimmung der aktuellen Position an. Ausserdem kann man sich an einen Event anhängen um mit `Start()` regelmässig die neue Position zu erhalten. Zusätzlich enthält sie Properties um das Intervall der Benachrichtigung zu ändern und die zuletzt berechnete Position und den Namen des Localizers zu erhalten.

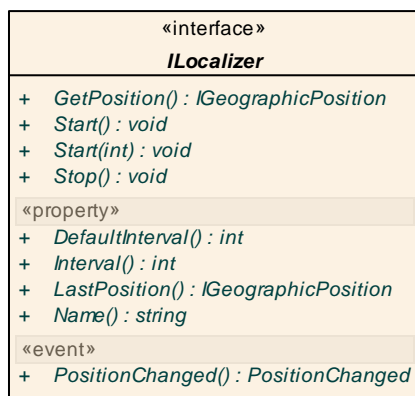


Abbildung 4: Das Interface ILocalizer

### AbstractLocalizer

*AbstractLocalizer* ist eine abstrakte Klasse und implementiert einen Grossteil von *ILocalizer*, so dass die konkreten Implementierungen (*GPSLocalizer*, *WLANLocalizer*) nur doch davon ableiten und das Verhalten von *GetPosition()* und *GetName()* hinzufügen müssen.

## MobileLocalizer

Der *MobileLocalizer* ist der Haupt-Lokalisierer und implementiert ebenfalls *ILocalizer*. Zusätzlich ist er eine Komposition aus verschiedenen Lokalisierern. Er bietet Methoden zum Hinzufügen und Entfernen von Lokalisierern und zum Bestimmen, welcher von ihnen bei der Positionsbestimmung bevorzugt werden soll. Versucht der *MobileLocalizer* die aktuelle Position herauszufinden, nimmt er zuerst den bevorzugten und anschliessend die anderen Lokalisierer, bis er ein Ergebnis erhält (oder bricht ab, falls niemand dazu in der Lage ist).

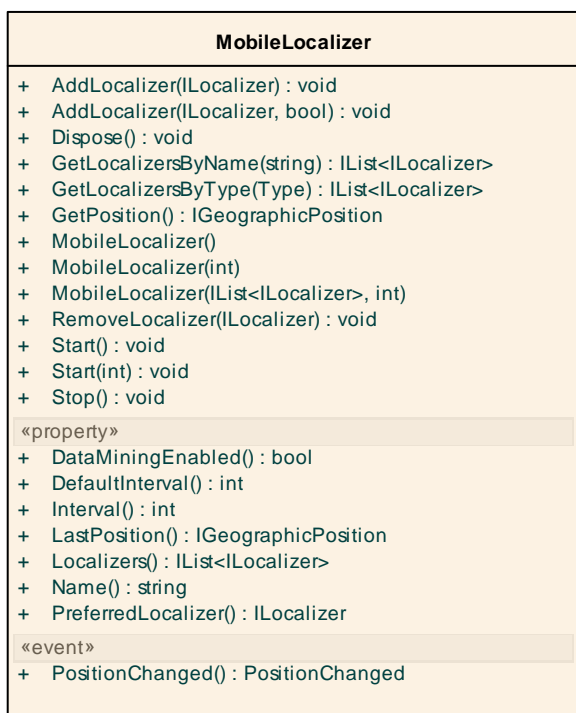


Abbildung 5: MobileLocalizer mit den öffentlichen Schnittstellen

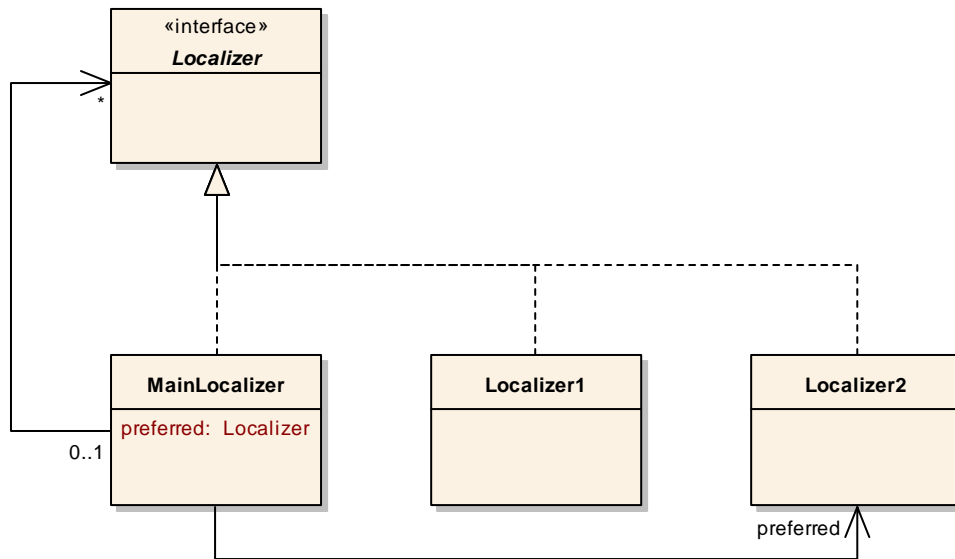


Abbildung 6: MobileLocalizer als Komposition von verschiedenen Localizern

## IGPSLocalizer

Das Interface *IGPSLocalizer* erweitert *ILocalizer* und definiert, welche zusätzliche Funktionen ein Lokalisierer anbieten muss. Dazu gehören Methoden um mit dem GPS-Empfänger zu interagieren und eine um festzulegen, wie viele Satelliten in Sichtweite sein müssen um die Positionsbestimmung als gültig zu bezeichnen.

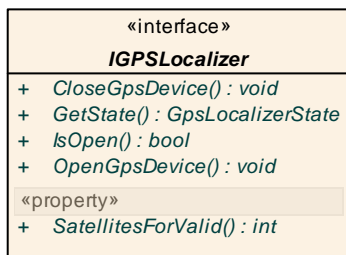


Abbildung 7: Das Interface IGPSLocalizer

## GPSLocalizer

*GPSLocalizer* erbt von *AbstractLocalizer* und implementiert *IGPSLocalizer*. Er benutzt eine Schnittstelle um vom GPS-Empfängergerät die Position zu erhalten

## IWLANLocalizer

Das Interface *IWLANLocalizer* erweitert *ILocalizer* und definiert zusätzlich eine Funktion um den *IWLANLocationCalculator* (siehe Package Localizer.Calculator) zu setzen.

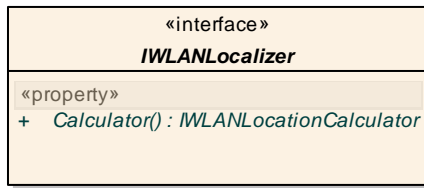


Abbildung 8: Das Interface IWLANLocalizer

## WLANLocalizer

*WLANLocalizer* erbt von *AbstractLocalizer* und implementiert *IWLANLocalizer*. Er benutzt einen *IWLANLocationCalculator* (siehe Package Localizer.Calculator) um seine Position zu berechnen.

## Localizer.Calculator

Ein *IWLANLocationCalculator* dient dem *WLANLocalizer* zur Berechnung der Position. Dabei übergibt man eine Liste mit *AccessPointScans* (Fingerprint), aus welcher der Calculator die Position ermittelt. (genauerer im Kapitel zur Positionsbestimmung)

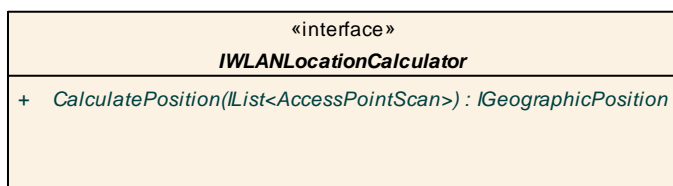


Abbildung 9: Das Interface IWLANLocationCalculator

## Commons

### Factory

*Factory* ist eine abstrakte Fabrik und bietet verschiedene Methoden um ein bestimmte Objekte zu erzeugen. Eine konkrete Fabrik leitet von *Factory* ab und implementiert deren abstrakten Methoden. Über die statischen Methoden kann eine Default-Factory gesetzt und geholt werden. Weitere Fabriken können mit einem Namen abgelegt und über diesen wieder geholt werden. Wird keine default Factory gesetzt wird die enthaltene *DefaultFactory* benutzt.

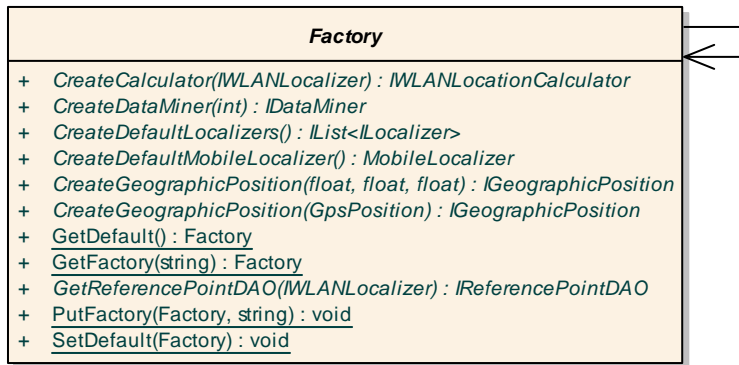


Abbildung 10: Die abstrakte Fabrik mit ihrer öffentlichen Schnittstellen

## DefaultFactory

Die von uns bereitgestellte *DefaultFactory* leitet von *Factory* ab und erzeugt folgende Objekte:

- *IWLANLocationCalculator*: *WLANLocationCalculatorWeighted*
- *IDataMiner*: *DataMiner*
- Default Localizers: einen *GPSLocalizer* und einen *WLANLocalizer*
- Default MobileLocalizer: *MobileLocalizer* mit den default Localizern
- *IGeographicPosition*: *WGS84Position*
- *IReferencePointDAO*: einen Singleton *ReferencePointDAO*

## ReferencePoint

Das Struct *ReferencePoint* dient als Referenzpunkt für eine Messung. Er speichert dabei die ermittelte geographische Position und einen „Fingerprint“ des Funknetzes. Ein Fingerprint gibt Auskunft über die an dieser Position messbaren Accesspoints in Form von MAC-Adresse der empfangenen Signalstärke.

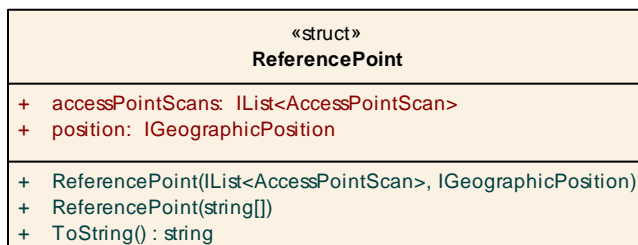


Abbildung 11: Das Struct ReferencePoint mit seiner öffentlichen Schnittstelle

## WeightedReferencePoint

Ein *WeightedReferencePoint* wird bei der Berechnung der Position mittels WLAN verwendet. Er besteht aus einem *ReferencePoint* und einer Distanz. Diese bezeichnet die Distanz zwischen dem Funknetz-Fingerprint der zur ermittelnden Position und des Fingerprint des Referenzpunktes. (Genauerer dazu im Kapitel zur Positionsbestimmung)

«struct» <b>WeightedReferencePoint</b>
+ distance: float + referencePoint: ReferencePoint
+ Compare(WeightedReferencePoint, WeightedReferencePoint) : int + Equals(Object) : bool + GetHashCode() : int + WeightedReferencePoint(ReferencePoint, float)

Abbildung 12: Das Struct *WeightedReferencePoint* mit seiner öffentlichen Schnittstelle

## Commons.Location

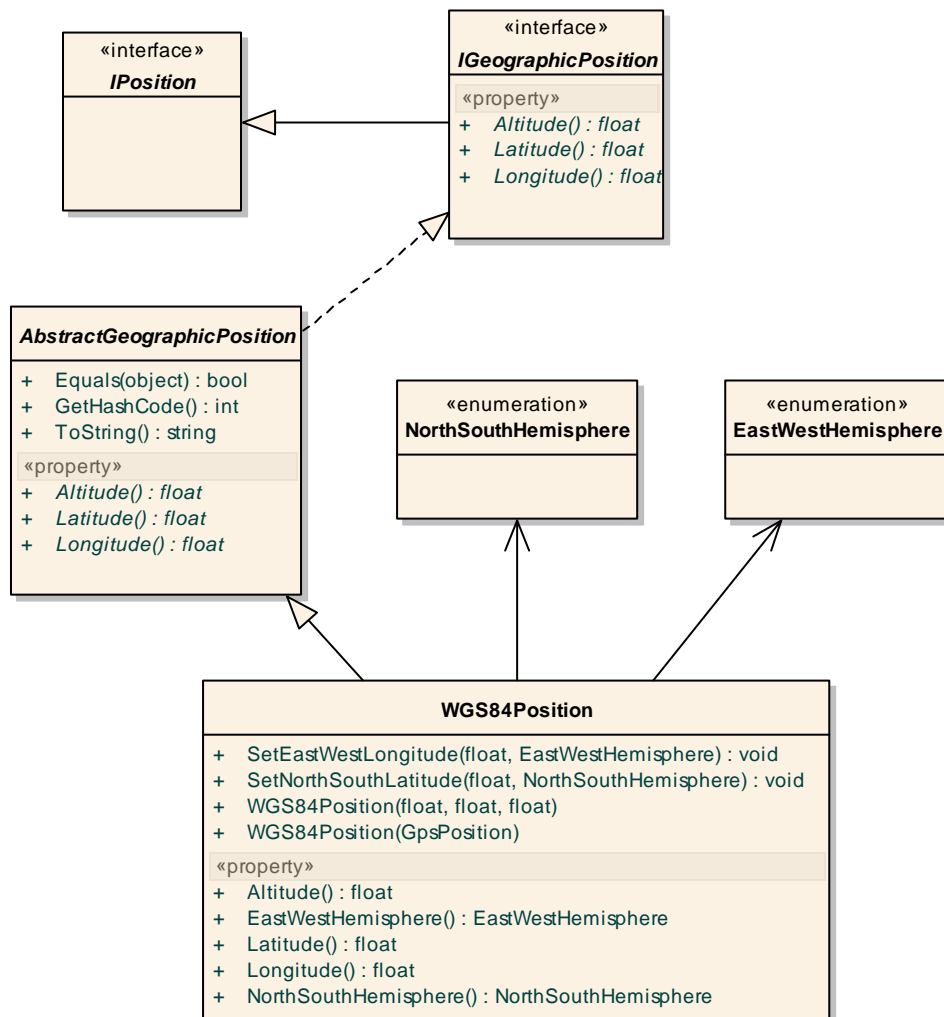


Abbildung 13: Die Klassen des Package Commons.Location mit ihren öffentlichen Schnittstellen und Verwandtschaften

### IPosition

*IPosition* ist ein Markerinterface, das eine Position auf der Erde repräsentiert. Da verschiedene Koordinatensysteme existieren (z.B. kartesische und geographische) beinhaltet die Schnittstelle keine Methoden.

### IGeographicPosition

*IGeographicPosition* repräsentiert eine geographische Position mit Höhe, Längen- und Breitengrad.

### AbstractGeographicPosition

*AbstractGeographicPosition* implementiert *IGeographicPosition* und dient dazu die `Equals`-, `GetHashCode` und `ToString`-Methoden einheitlich zu überschreiben. Die Properties bleiben abstrakt. Konkrete Implementierungen leiten dann von *AbstractGeographicPosition* ab.

### WGS84Position

*WGS84Position* leitet von *AbstractGeographicPosition* ab (und implementiert somit *GeographicPosition*). Sie repräsentiert eine geographische Position nach dem WGS84-Standard. Dieser ist ein weit verbreiteter Standard für die Positionsangabe und wird auch vom GPS-System verwendet. Zusätzlich bietet sie Methoden um die Koordinaten nur mit positiven Zahlen anzugeben und dafür die Halbkugel angeben. Z.B. müsste man für eine Position auf der Südhalbkugel eine negative Zahl für die Latitude (Breitengrad) angeben. So kann man dies aber, in dem man *NorthSouthHemisphere.South* angibt.

## Commons.WLAN

### AccessPointScanner

Die Klasse *AccessPointScanner* dient dazu das Funknetz zu scannen. Mit *ScanWLAN()* erhält man eine Liste mit Accesspoints. Man kann sich aber auch an einen Event anhängen um sich regelmässig über die Funknetz-Umgebung zu informieren.

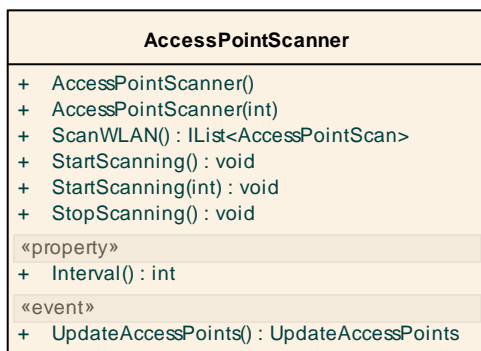


Abbildung 14: AccessPointScanner und seine öffentliche Schnittstellen

### AccessPointScan

Ein *AccessPointScan* repräsentiert eine Momentaufnahme eines Accesspoints. Dabei werden jeweils nur die für die Applikation nötigen Informationen gespeichert:

- *mac*: Die MAC-Adresse des Accesspoints. Sie besteht aus 6 Byte und ist weltweit eindeutig. Um Speicher zu sparen wandeln wir sie in ein long um (long = 8B, 6-Byte-Array = 6B + 4B (Pointer) + 4B (array.length) = 14B). Zum dem erleichtert es die weitere Verarbeitung.
- *dbm*: Die Signalstärke zum Zeitpunkt und am Ort der Messung, welche man vom AP empfängt.

Ein *AccessPointScan* ist ein Struct, weil er einmal erstellt wird und danach nur noch mit den Werten weiter gearbeitet wird. Durch das Struct erhöht sich die Performance. Die Compare-Methode ist zum Sortieren der Scans nach der MAC-Adresse.

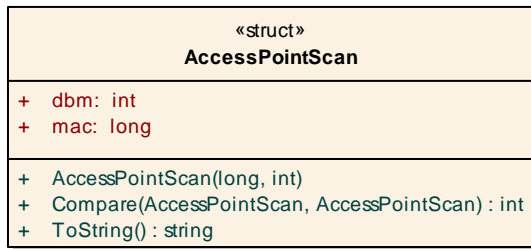


Abbildung 15: Das Struct AccessPointScan mit der öffentlichen Schnittstelle

## Datamining

Im Datamining-Package befinden sich das Interface *IDataMiner* und die Implementierung dazu. Der DataMiner erfasst *ReferencePoints*. Dazu verwendet er *AccessPointScanner* zum erfassen von *AccessPointScans* und *GPSLocalizer* zum bestimmen der Position. Über *IDataMinerPersistenceManager* (siehe Package „Persistence“) speichert er die Referenzpunkte. Über Properties kann erfahren werden, wie viele Einträge gespeichert wurden.

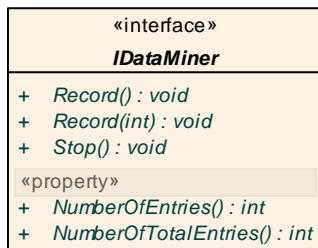


Abbildung 16: Das Interface IDataMiner

## Persistence

### IDataMinerPersistenceManager

Ein *IDataMinerPersistenceManager* ist dafür zuständig, Files, in welche vom Data-Miner gesammelte Daten geschrieben werden, zu öffnen und zu schliessen. Er bietet eine Methode zum speichern eines *ReferencePoints*.

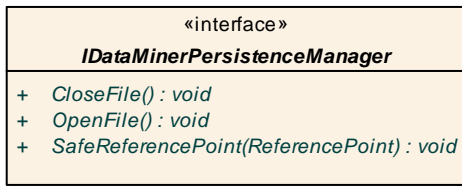


Abbildung 17: Das Interface IDataMinerPersistenceManager

## **IReferencePointDAO**

Die Schnittstelle *IReferencePointDAO* bietet Methoden um von einem Fingerprint seine Nachbarn zu erhalten. Dies können alle sein oder nur die nächsten. Zum Interface existieren verschiedene Implementierungen. Details zu diesen sind im Analyse Abschnitt zu finden.

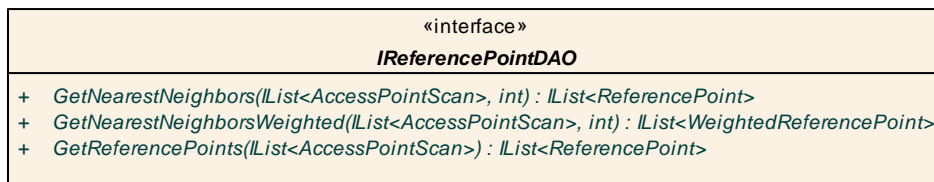


Abbildung 18: Das Interface IReferencePointDAO

## Positionsbestimmung mit WLAN

### Referenzdatenbank

Für die Positionsbestimmung mittels WLAN-Messungen wird eine Referenzdatenbank benötigt. Diese ermöglicht, aufgrund gemessener Accesspoints auf eine geographische Position zu schliessen. Die geographischen Koordinaten eines beliebigen Accesspoints sind grundsätzlich unbekannt. Es gibt verschiedene Ansätze zum Inhalt dieser Datenbank. Eine Möglichkeit ist das Erfassen der Positionen der Accesspoints. Dieser Aufbau eignet sich in Szenarien, in denen die Positionen der Accesspoints bekannt ist, oder leicht bestimmt werden kann (zum Beispiel Firmengelände).

Im Fall der Positionsbestimmung im öffentlichen Raum, zum Beispiel in Städten, wo die genauen Positionen der Accesspoints unbekannt sind, könnte mit Messungen der an verschiedenen Positionen unterschiedlich stark empfangenen Signalstärken eine Approximation der Position des APs gefunden werden.

Zur Verwendung kam eine andere Variante, bei welcher nicht mit der Position der APs, sondern die geographische Position der Messung zusammen mit einem WLAN-Fingerprint gespeichert wird. Ein Fingerprint gibt Auskunft über die an dieser Position messbaren APs, und gemessenen Signalstärken (RSS, received signal strength). Ein Paar (geographische Koordinate, WLAN-Fingerprint) wird hier als Referenzpunkt bezeichnet.

### Datenverwaltung

#### Server

Es wird ein Server benötigt, der die Referenzdaten zentral verwaltet. Dieser liefert für eine Region (zum Beispiel eine Stadt) alle vorhandenen Referenzpunkte. Die Anfrage enthält einen Regionnamen, oder eine Liste mit APs, die in einer Initial-Messung gefunden wurden. Mit diesem Input kann die Region, in der sich ein Mobilgerät befindet bestimmt werden, und ihm die in der Datenbank zu dieser Region bekannten Referenzpunkte zurückliefern.

#### Mobilgerät

##### *Speicherbedarf*

Annahme: Die Referenzdatenbank einer Region besteht aus 100'000 Referenzpunkten. Die Dichte der Referenzpunkte ist so gross, das ein durchschnittlicher WLAN-Fingerprint 10 APScans beinhaltet.

Resultat: Ein Referenzpunkt bestehend aus 3 Float-Werten (Breiten- und Längengrad, Höhe) und einem Fingerprint mit 10 APs hat folgende Grösse:

$$(3 \times 4B) + (10 \times 12B) = 112B$$

Werden alle Referenzpunkte in den Speicher geladen, wird dafür  $100'000 \times 112B \approx 10MB$  Speicher benötigt. Dies ist bereits eine Grösse, die für Mobilgeräte zum Problem werden kann. Je nach Grösse und Qualität der Referenzdatenbank könnte auch einiges mehr an Speicher benötigt werden.

Aufgrund dieser Überlegungen wurde die für die Speicherung der Referenzdatenbank auf dem Mobilgerät folgendes Vorgehen angewendet.

Die Referenzdaten einer Region werden in den externen Speicher des Mobilgerätes geladen. Dabei werden Referenzpunkte abhängig ihrer geographischen Koordinaten, in mehrere Files verteilt. Diese Struktur ermöglicht ein lazy-loading der Referenzpunkte, es müssen nicht alle Referenzpunkte der Region in den Arbeitsspeicher geladen werden.

Während der Laufzeit wird beim ersten Ortungsvorgang eine erste Messung vorgenommen. Nun wird in den Files nach einem der gemessenen APs gesucht. Existiert ein solches File, wird angenommen, dass die ungefähre Position innerhalb, oder im Umkreis des vom gefundenen File abgedeckten Abschnittes der Region liegt. Die Referenzpunkte des gefundenen Files, sowie die der acht umliegenden Files werden in den Arbeitsspeicher geladen.

Die Positionsbestimmung basiert nur auf den aktuell geladenen Referenzpunkten. Da sich die Position des Mobilgerätes ändert, müssen fortlaufend neue Files geladen werden, andere Referenzpunkte werden nicht mehr benötigt, und können aus dem Speicher gelöscht werden.

Um diese Anpassungen einfach zu gestalten, werden die geladenen Referenzpunkte von der API in einer 3x3-Grid Datenstruktur (siehe Abschnitt Grid-Datenstruktur) verwaltet. Das Grid bewegt sich über die Files der Region, wobei es seine Grösse beibehält. Die Bewegung wird durch die ermittelten Positionen initialisiert. Diese Position kann einem Node des Grids, und damit einem File zugewiesen werden. Verschiebt sich die Position vom zentralen Node in einen angrenzenden Node, wird dabei das Verschieben des Grids in dieselbe Richtung initiiert. Dieser Vorgang beinhaltet das Nachladen und Entfernen von Referenzpunkten in respektive aus dem Arbeitsspeicher.

### **Index**

Aus den geladenen Referenzpunkten muss schnell die Menge von Referenzpunkten bestimmt werden können, deren Fingerprint mindestens eine Übereinstimmung (gemeinsamer AP) mit der Ortungs-Messung hat. Diese Menge wird als die Menge der Nachbarn einer Messung bezeichnet. Die Grösse der Menge wird bestimmt durch die örtliche Dichte der Referenzpunkte der Datenbank, oder anders gesagt: sind von der Umgebung einer Ortung viele Referenzpunkte vorhanden, existieren für die Ortung tendenziell mehr Nachbarn.

Um die Nachbarn zu bestimmen müsste nun über alle Referenzpunkte iteriert, und die WLAN-Fingerprints mit der Ortungs-Messung auf identische MAC-Adressen überprüft werden. Um diesen Zugriff schneller zu gestalten, wird zusätzlich ein Index eingeführt. Der Index besteht aus einem assoziativen Array mit dem Schlüssel MAC-Adresse und einer Liste von Referenzpunkten als Wert. Der Index wird synchron mit dem Grid aktualisiert. Das heisst, werden zusätzliche Referenzpunkte geladen, wird auch der Index mitgeführt. Der Index ermöglicht einen Nachbar-Lookup in  $O(\log n)$ .

### **Ermitteln der nächsten Nachbarn**

Die Menge der Nachbarn ist typischerweise zu gross, um daraus eine gemittelte Position zu bestimmen. Es sollen nur Nachbarn in die Berechnung miteinbezogen werden, von denen aufgrund ihres WLAN-Fingerprints angenommen werden, dass sie sich in der Nähe der zu bestimmenden Position befinden könnten. Diese Nachbarn werden als nächste Nachbarn bezeichnet. Die Selektion dieser Nachbarn ist ein Kernproblem der gesamten Positionsbestimmung.

Als Grundlage für die Ermittlung dienen alleine die WLAN-Fingerprints. Ein Vergleich Nachbar – Ortungs-Messung wurde zunächst mit einer einfachen Berechnung durchgeführt. Um die Selektion zu verbessern, wurde dann die Methode angepasst. Dem WLAN-Fingerprint Vergleich werden die folgenden Charakteristiken zugeordnet:

- Anzahl gemeinsame APs  $n$
- Differenz  $s = \text{Max}(\#APs \text{ der Referenz}, \#APs \text{ der Messung}) - \#gemeinsame APs$

- Distanz  $d =$  „Distanz“ der Fingerprints

### Distanz

Die ist nicht als metrische Entfernung zu verstehen, sondern als Mass für die Ähnlichkeit der Fingerprints. Um diese zu Bestimmen, wird über die empfangenen Signalstärken gemeinsamer APs die euklische Distanz berechnet.

$x =$  Signalstärken Messung,  $y =$  Signalstärken Referenzpunkt

$$d = \|x - y\| = \sqrt{\sum_n (x_i - y_i)^2}$$

Dabei wurde zunächst nicht berücksichtigt, dass die empfangene Signalstärke in dBm gehandhabt wird. Dies führt zu einer falschen Berechnung der Distanz, da die dBm eine logarithmische Grösse ist. Um diesem Umstand gerecht zu werden, wurde in einer verbesserten Variante die empfangene Signalstärke in milliWatt umgerechnet:

$$x' = 10^{(x/10)} mW, \quad d = \|x' - y'\|$$

Nun werden die Leistungen linear verglichen. Eine weitere Anpassung versucht der Tatsache gerecht zu werden, dass im idealen Modell die Signalstärken im Quadrat zum Abstand der Quelle abnehmen:

$$P \sim P_0 / r^2 \quad \rightarrow \quad r \sim \sqrt{P} \quad x'' = \sqrt{x'}$$

Damit wird die euklidische Distanz über Werte in metrischer Einheit berechnet.

### Penalty

Aus der Distanz alleine kann noch nicht darauf geschlossen werden, dass sich ein Referenzpunkt in der Nähe der zu bestimmenden Position befindet. Je grösser seine Differenz, desto grösser die Wahrscheinlichkeit, dass er nicht in der Nähe der Position der Messung liegt. Die Distanz wird deswegen mit mit einem Penalty-Funktion manipuliert:

$$d = d + (s \times A)^B$$

A, B s d ind d dabei Faktoren, die empirisch ermittelt werden

### Positionsberechnung

Die approximierte Position wird aus den Koordinaten der nächsten Nachbarn berechnet. Dabei wird die berechnete Distanz der Referenzpunkte miteinbezogen. Da nähere Nachbarn stärker in die Berechnung einfließen sollen als weiter entfernte, wird die Distanz eines Referenzpunktes ( $d$ ) von der Distanz des am weitesten entfernten Punktes ( $d_{max}$ ) subtrahiert.

$$w_i = d_{max} - d_i$$

Die erhaltenen Gewichte ( $w$ ) sind nun gross für nähere Punkte, und klein für entferntere Punkte. Für Breiten-, Längengrad und Höhe wird jeweils das gewichtete Mittelmass berechnet (analog für Längengrad und Höhe):

$$lat_{avg} = \frac{\sum_n (w_i * lat_i)}{\sum_n (w_i)}$$

(*n*: Anzahl Referenzpunkte, *w*: Gewichte und *lat*: Breitengrade der Referenzpunkte)

Die approximierte Position ist somit ( $lat_{avg}$ ,  $lon_{avg}$ ,  $height_{avg}$ ).

### Datenstruktur Grid

Das Grid ist die zweidimensionale Variante der Liste. Grafik und Tabelle veranschaulichen die Gemeinsamkeiten und Unterschiede. Im Unterschied zur Liste wird dem Grid nicht ein einzelner Node angefügt, sondern jeweils eine Liste von Nodes, deren Länge der momentanen Breite respektive Höhe des Grids gleich sein muss. Dasselbe gilt für die Methoden zum Entfernen von Nodes.

Datenstruktur	Liste	Grid
Referenzen eines Nodes	Left, right	North, East, South, West
Dimension	Length	Width, Height
Hinzufügen	AddFirst(Node), AddLast(Node)	AddNorth(Node[]), AddEast(Node[]), AddSouth(Node[]), AddWest(Node[])

## Analyse der WLAN Lokalisierung

### Test-Aufbau

Die verwendete Referenzdatenbank besteht aus einer Sammlung von Referenzpunkten, die beim mehrmaligen Ablaufen einer bestimmten Strecke (ca. 700 Meter, im Zentrum von Brugg) zu verschiedenen Tageszeiten, und in beide Richtungen gesammelt wurden.

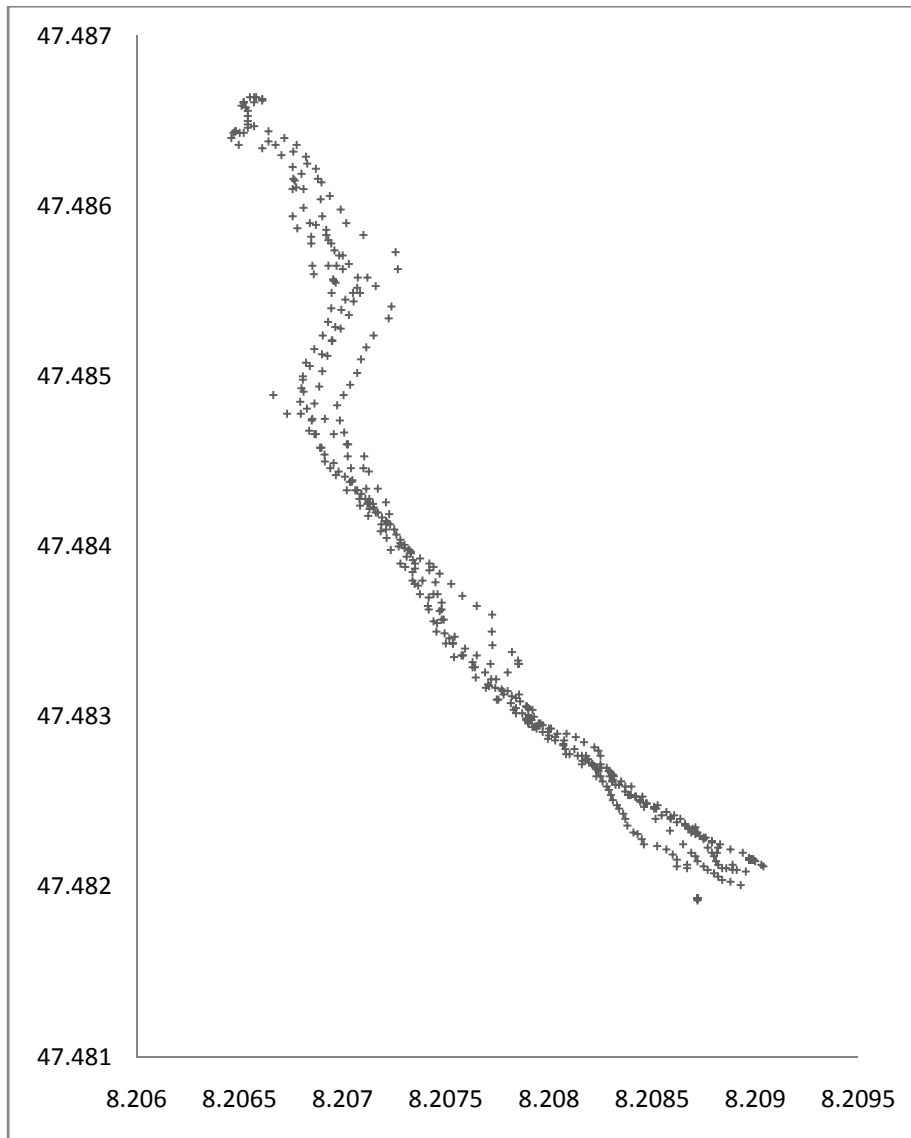


Abbildung 19: Koordinaten erfasster Referenzpunkte

Das Diagramm zeigt die Koordinaten der erfassten Referenzpunkte. Dass nicht immer die genau gleiche Strecke abgelaufen wurde, und die GPS-Messung aufgrund der zum Teil hohen Gebäude nicht exakt ist, erklärt die Streuung der Punkte, die sonst alle auf eine Linie liegen würden.

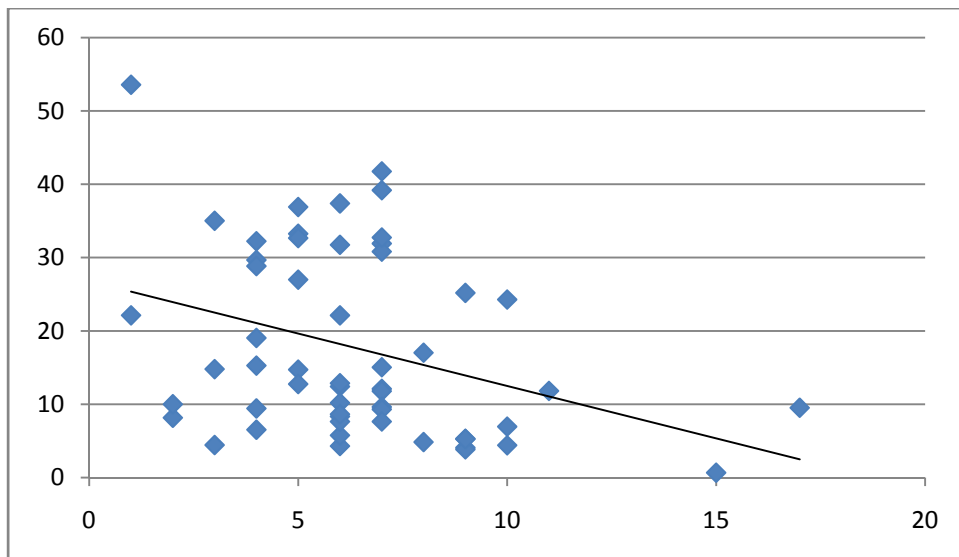


Abbildung 20 Distanz zwischen ermittelten und wirklichen Koordinaten

Nun wurde eine weitere Messung derselben Strecke als Input für die Positionsbestimmung verwendet. Für jeden Referenzpunkt wurden dem API der WLAN-Fingerprint übergeben, und die damit ermittelte Position den gemessenen GPS-Koordinaten gegenübergestellt.

Zunächst kann festgestellt werden, dass der Fehler der Position im Zusammenhang mit der Anzahl gemessener APs steht. Mit einem WLAN-Fingerprint mit vielen APs kann die Position genauer berechnet werden.

### Vergleich der verschiedenen Nearest Neighbors Algorithmen

Es wurden vier verschiedene Algorithmen angewendet, die sich jeweils in der Berechnung der nächsten Nachbarn unterscheiden

Simple Variante, ohne dBm Umrechnung, ohne Penalty

- a) wie a), mit Penalty
- b) dBm Umrechnung in mWatt, Penalty
- c) dBm Umrechnung in Meter, Penalty

Alle vier Varianten wurden mit derselben Testmessung auf die Referenzdatenbank getestet. Dabei wurde die Anzahl nächster Nachbarn, über welche die Position approximiert wird variiert. In der Tabelle sind die Ergebnisse dargestellt.

n	a)	b)	c)	d)
3	15.02	18.74	18.62	18.51
4	14.60	17.40	17.06	17.39
5	14.95	16.68	16.16	16.69
8	15.94	16.54	16.09	17.03
12	16.90	16.35	16.32	16.75
18	16.52	15.37	15.36	15.12
25	17.69	14.36	14.69	14.43

Abbildung 21 Mittelwerte der Abweichung der berechneten Distanz gegenüber der GPS-Koordinaten aller Messungen

## Ergebnisse

Es wird ersichtlich, dass die beste Abweichung für alle vier Varianten etwa dieselbe ist. Ausser bei Variante a) nimmt die Genauigkeit mit der Zahl nächster Nachbarn fast stetig zu. Die Variante, welche ohne Penalty-Funktion arbeitet, verliert mit zunehmender Zahl Nachbarn an Genauigkeit. Jedoch erreicht auch sie einen ähnlichen Grad an Präzision.

Ein detaillierter Vergleich der Varianten ist schwierig. Die verwendete Referenzdatenbank beinhaltet zwar ein einigermaßen dichtes Netz von Referenzpunkten, jedoch sind die mit GPS ermittelten Koordinaten zu ungenau. Häufig weichen die Punkte im Bereich von mehreren Metern von ihrer tatsächlichen Position ab. Dies wird umso mehr zum Problem, wenn mehrere Messungen der gleichen Strecke vorhanden sind, bei denen die Abweichung der Gps Koordinaten variiert. Wenn also aus Messungen an derselben Position zu zwei Zeitpunkten zwei Referenzpunkte resultieren, die unterschiedliche Koordinaten aufweisen. Um die Analyse und daraus die Präzision der Positionsbestimmung zu verbessern würde eine exaktere Referenzdatenbank benötigt.

## Visualisierung

Die folgende Abbildung illustriert die Referenzdatenbank, die Punkte stellen jeweils einen Referenzpunkt dar.



Abbildung 22: Referenzpunkte mit Google Earth dargestellt

Folgendes Bild zeigt die Ergebnisse eines Testdurchlaufes. Gelbe Pfeile stellen die mit GPS gemessenen Koordinaten dar, weiße Punkte die von der API berechneten Koordinaten.



Abbildung 23: Referenzpunkte (blau), wirkliche Position (grün) und ermittelte Position (weiss)

Folgendes Bild stellt eine detailliertere Ansicht dar, die grünen Linien zeigen auf, über welche Referenzpunkte die Position ermittelt wurde.

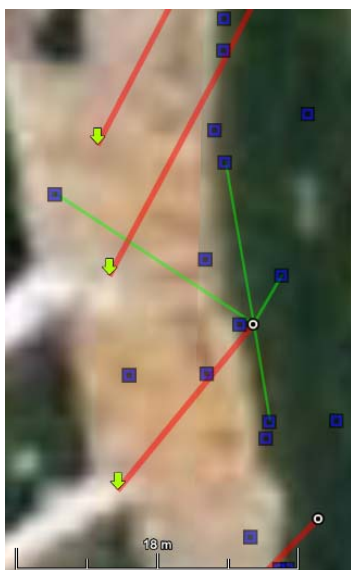


Abbildung 24: Benutzte Referenzpunkte zur Ermittlung der Position



Abbildung 1: Komponenten und ihre Beziehungen .....	3
Abbildung 2: Packageübersicht und die Enthaltenen Interfaces und Klassen .....	4
Abbildung 3: Klassenhierarchie der Localizer.....	5
Abbildung 4: Das Interface ILocalizer .....	5
Abbildung 5: MobileLocalizer mit den öffentlichen Schnittstellen .....	6
Abbildung 6: MobileLocalizer als Komposition von verschiedenen Localizern.....	7
Abbildung 7: Das Interface IGPSTLocalizer.....	7
Abbildung 8: Das Interface IWLANLocalizer .....	8
Abbildung 9: Das Interface IWLANLocationCalculator .....	8
Abbildung 10: Die abstrakte Fabrik mit ihrer öffentlichen Schnittstellen.....	9
Abbildung 11: Das Struct ReferencePoint mit seiner öffentlichen Schnittstelle.....	9
Abbildung 12: Das Struct WeightedReferencePoint mit seiner öffentlichen Schnittstelle.....	10
Abbildung 13: Die Klassen des Package Commons.Location mit ihren öffentlichen Schnittstellen und Verwandtschaften .....	11
Abbildung 14: AccessPointScanner und seine öffentliche Schnittstellen .....	12
Abbildung 15: Das Struct AccessPointScan mit der öffentlichen Schnittstelle .....	13
Abbildung 16: Das Interface IDataMiner.....	13
Abbildung 17: Das Interface IDataMinerPersistenceManager.....	14
Abbildung 18: Das Interface IReferencePointDAO.....	14
Abbildung 19: Koordinaten erfasster Referenzpunkte .....	19
Abbildung 20 Distanz zwischen ermittelten und wirklichen Koordinaten .....	20
Abbildung 21 Mittelwerte der Abweichung der berechneten Distanz gegenüber der GPS-Koordinaten aller Messungen .....	20
Abbildung 22: Referenzpunkte mit Google Earth dargestellt .....	21
Abbildung 23: Referenzpunkte (blau), wirkliche Position (grün) und ermittelte Position (weiss).....	22
Abbildung 24: Benutzte Referenzpunkte zur Ermittlung der Position .....	22

## Glossar

VS Visual Studio

GPS Global Positioning System

WLAN Wireless Local Area Network (Drahtloses Netzwerk)

AP AccessPoint (Hotspot, WLAN-Antenne)

WLAN-Fingerprint Wird hier verstanden als die Gesamtheit der an einer bestimmten Position zu einem Zeitpunkt gemessenen Signalstärken der gefundenen Accesspoints