6 Systemarchitektur und Implementation

6.1 Systemarchitektur

Bei der entwickelten Software zur Bestimmung der Position handelt es sich um eine eigenständige Version, die auf einer Eigenortung basiert (siehe Abschnitt 2.1.2). Das bedeutet, dass keine zusätzlichen Komponenten benötigt werden, außer einer bestehenden Wireless LAN Infrastruktur. Dabei handelt es sich um eine Softwarelösung für Handhelds (vgl. Abschnitt 4.2.2) und wird weiterführend als MaWIPS[46] bezeichnet. Diese Software stellt eine Benutzerschnittstelle zur Umgebung bereit, die es ermöglicht, Signaldaten zu erfassen, auszuwerten und Positionsdaten zu visualisieren. Dabei wurde auf die Handlichkeit und eine spätere Möglichkeit der Portierung auf weitere mobile Endgeräte geachtet.

Die nachfolgende Abbildung 6.1 zeigt die Systemarchitektur von MaWIPS, welche im Wesentlichen aus fünf Teilen besteht.

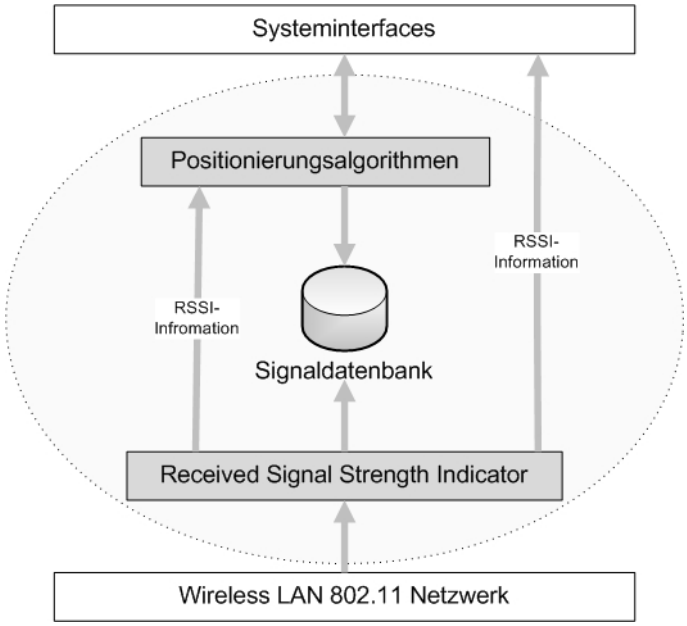

Abb. 6.1: Systemarchitektur von MaWIPS

Die unterste Schicht stellt das Wireless LAN 802.11 Netzwerk dar. Darauf aufbauend befindet sich die Schicht Received Signal Strength Indicator. Diese stellt die empfangenen Signalstärken und die Identifikationsparameter der verwendeten Access Points bereit. Hierbei wird

46. MaWIPS steht für Matschers Wireless Indoor Positioning System.

die MAC-Adresse zur eindeutigen Zuordnung verwendet. Die dritte Schicht stellt die Datenbank mit den jeweiligen Signalstärken und deren Verknüpfungen zu den Positionsdaten dar. Darüber befindet sich die Schicht der Positionierungsalgorithmen, die nach erfolgreicher Ermittlung der Position, die Daten an die Systeminterfaceschicht weiter gibt. Diese verarbeitet und visualisiert die Positionsdaten, um dem Benutzer spezifische Informationen zum Standort anzugeben. Zusätzlich wird über die Systeminterfaceschicht dem Benutzer das Erstellen einer Radio-Map über das gewünschte Operationsgebiet ermöglicht.

Die Realisierung des Systems erfolgte mit der Entwicklungsumgebung Visual Studio 2005 .NET von Microsoft mithilfe der Programmiersprache Visual C#[47]. Zusätzlich wurde das Compact Framework 2.0 Beta von OpenNETCF[48] benötigt, das speziell für die Entwicklung auf mobilen Endgeräten ausgerichtet ist. Das zur Umsetzung benutzte mobile Endgerät war ein Personal Digital Assistant X51v 624 von Dell mit einem eingebauten Wireless LAN Adapter. Das installierte Betriebssystem Windows Mobile 5.0 stammt von der Firma Microsoft und integriert neuere WLAN Schnittstellen.

6.2 Systeminterfaces

Die Positionierungssoftware MaWIPS besteht im Wesentlichen aus drei Fenstern, über die die Interaktion des Benutzers realisiert wird. Dabei gestaltet sich die Benutzeroberfläche weitestgehend selbsterklärend. Nachfolgend werden die einzelnen Fenster der Software grob umrissen.

6.2.1 Wireless LAN Spotter

Das erste Fenster stellt den Wireless LAN Spotter[49] dar, der die empfangbaren Access Points in der Umgebung anzeigt. Mithilfe dieser Komponente kann eine Überprüfung des Operationsgebietes auf Tauglichkeit für eine Positionsbestimmung festgestellt werden. Zusätzliche Identifikationsparameter der jeweiligen Access Points, wie zum Beispiel die MAC-Adresse, werden entsprechend ausgegeben. Die dazugehörige Signalstärkeinformation (*Signal*) wird hierbei in dBm angegeben, wie es die Abbildung 6.2 illustriert.

47. Visual C# ist eine objektorientierte Programmiersprache, die die Entwicklung sprachunabhängiger .NET-Komponenten unterstützt [Microsoft2006].
48. Open Source Erweiterung des Compact Frameworks von Microsoft, welches den Zugriff auf die WLAN-Komponenten ermöglicht [www.opennetcf.org].
49. Spotter - engl. bedeutet Aufklärer.

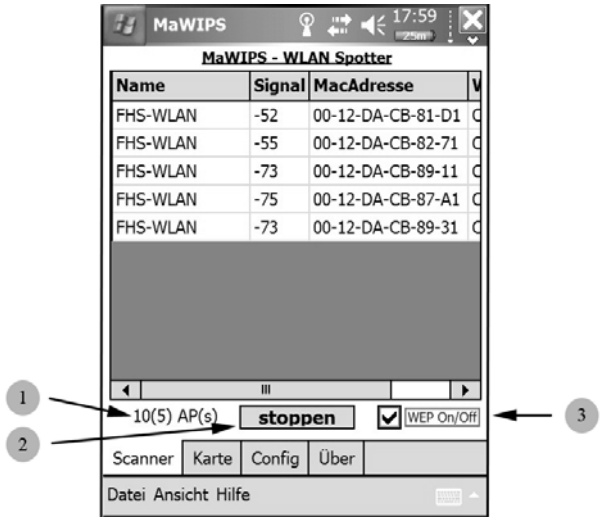

Abb. 6.2: Wireless LAN Spotter

Die Gesamtanzahl (1) der Access Points wird im unteren Teil des Spotters angezeigt. Des Weiteren kann über eine Auswahlmöglichkeit (3) das Anzeigen von unverschlüsselten Access Points erfolgen. Über den Start/Stop Button (2) wird das System aktiviert bzw. deaktiviert.

6.2.2 Konfigurationsfenster

Bevor die Software eingesetzt werden kann, sollten einige Parameter eingestellt werden. Dabei kann über das Konfigurationsfenster (siehe Abbildung 6.3) unter anderem die Anzahl der zuverwendenden Access Points (3) bestimmt sowie der zu benutzende Positionierungsalgorithmus (5) ausgewählt werden.

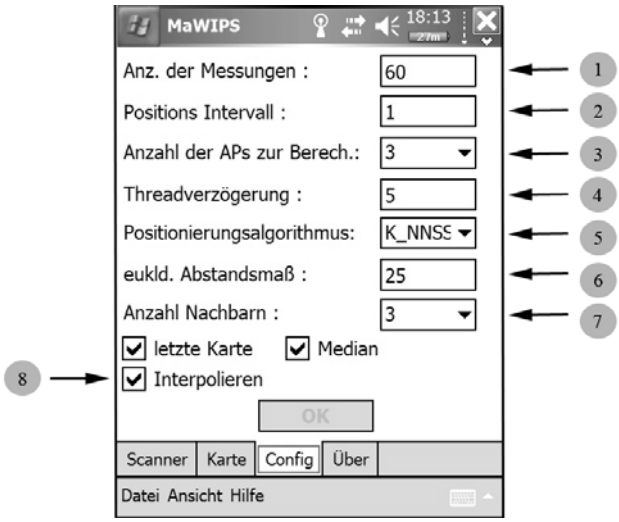

Abb. 6.3: Konfigurationsfenster

Weiterhin kann über die Einstellungsoption (6) das euklidische Abstandsmaß begrenzt werden. Dadurch können die zu weit entfernten Messpunkte im Vorfeld von der Positionsbestimmung ausgeschlossen werden. Zusätzlich besteht die Möglichkeit das Einbeziehen von Nachbarn (7) zu bestimmen. Der Wertebereich liegt hierbei zwischen eins und zehn. Mithilfe der Option (1) kann die Aufnahmezeit von Messdaten in Sekunden eingestellt und über den Positionierungsintervall (2) die Ausgabe der zu ermittelnden Position verzögert werden. Über die Option (8) ist das Aktivieren der Interpolation der Position möglich. Die Einstellmöglichkeit der Threadverzögerung (4) dient zur Anpassung an verschiedene Systeme.

Diese Einstellungen können während der Benutzung von MaWIPS verändert werden, um dadurch eine Feinjustierung der Software zu ermöglichen bzw. zu erreichen.

6.2.3 Positionierungsfenster

Die wichtigste Komponente stellt das Positionierungsfenster dar. Über dieses Fenster werden die Trainingsdaten aufgenommen, die Messpunkte visualisiert und die aktuell berechnete Position angezeigt. In der Abbildung 6.4 ist das Positionierungsfenster mit verschiedenen Messpunkten zu sehen.

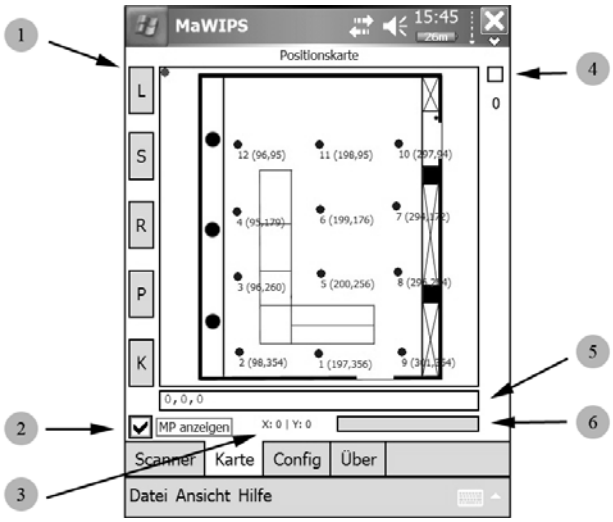

Abb. 6.4: Positionierungsfenster

Im Hintergrund wird das Layout des Multi-Media-Labor F2.06 im Gebäude F der Fachhochschule Schmalkalden dargestellt. Das austauschbare Layout hilft dabei die Messpunkte und die spätere Position in verständlicherweise visuell anzuzeigen. Dadurch kann eine Verifizierung der ermittelten Positionsdaten erfolgen und eventuelle Anpassungen der Messpunkte,

sowie Einstellungen der Software vorgenommen werden. Die auf der linken Seite befindlichen Buttons (1) stellen für das System wichtige Funktionen dar. Der Reihe nach von oben an gesehen bedeutet:

L - läd eine Signalstärketabelle,
S - speichert eine Signalstärketabelle,
R - löscht die aktuellen Signaldaten,
P - startet die Positionierungsphase und
K - läd eine bestimmte Karte (Layout eines Operationsgebietes).

Neben der Checkbox (2) für die Anzeige der Messpunkte wird die aktuell ermittelte Position (3) in Koordinaten (*x* und *y*) angezeigt. Weiterhin wird über den rechten Fortschrittsbalken (6) der Status der Aufnahme von Messdaten visuell dargestellt. Die Textbox (5) liefert Informationen über den Zustand der Software und ein Indikator (4) beschreibt die Spotter Aktivität.

6.3 Koordinaten und Datenstruktur

6.3.1 Koordinaten

Die Positionierungssoftware MaWIPS kann eine Position auf zwei Wegen beschreiben (vgl. Abschnitt 2.1). Die einfachste Beschreibung ist die Bezugnahme auf lokale Gegebenheiten, wie zum Beispiel die Raumnummer. Hierbei wird lediglich die Beschreibung als einfacher String in der Signaldatenbank abgespeichert.

Als zweite Möglichkeit der Beschreibung wird die geometrische Variante verwendet. Diese beschreibt eine Position anhand eines Bezugssystems. Im Falle von MaWIPS wird der lokale Grundriss eines Operationsgebietes verwendet, welches mit einem dreidimensionalen Koordinatensystem assoziiert wird. Jede Position besitzt dabei die Koordinaten *x*,*y* und *z*. Über die Angabe der *z*-Koordiante ist es möglich eine Unterscheidung von Etagen in einem mehrstöckigen Gebäude zu realisieren, wie es die Abbildung 6.5 zeigt.

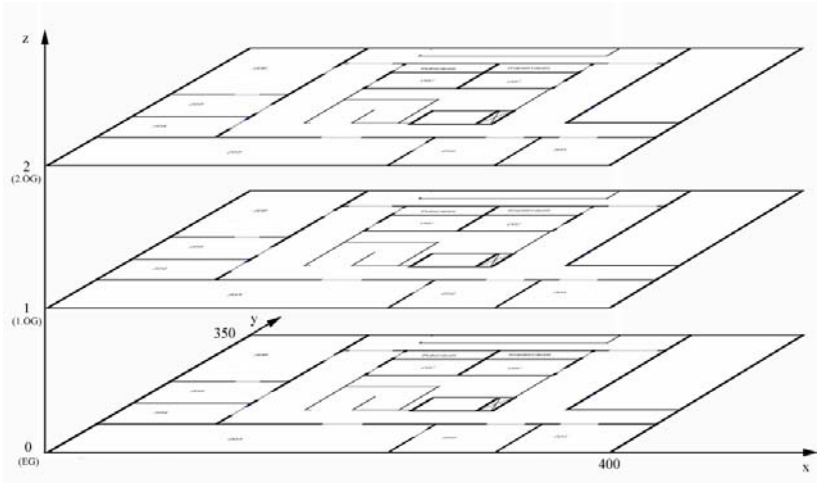

Abb. 6.5: Koordinatensystem F-Gebäude

Eine Kombination aus beiden Varianten stellt hierbei die Flexibilität und Benutzerfreundlichkeit der Software in den Vordergrund.

6.3.2 Hauptdatenstruktur

Die Hauptdatenstruktur der Positionierungssoftware ist zum einen der Messpunkt und zum anderen eine Struktur zum repräsentieren eines bestimmten Access Points.

Messpunkt

Der Messpunkt repräsentiert die genauen Positionsdaten. Es handelt sich hierbei um die Koordinaten (*x,y,z*) und die eindeutige Bezeichnung (*positionsName*). Weiterhin beinhaltet jeder Messpunkt einen Container[50] (*myApListe*) mit den jeweiligen Access Points und deren Eigenschaften. Zusätzlich können weitere Informationen angegeben werden, die die Universalität der Software unterstützen. Dabei handelt es sich um die Orientierung (*orient*), die Positions-ID (*positionsID*) und eine Liste der Messpunktnachbarn (*naechsteNachbarn*). Weiterhin besitzt der Messpunkt eine einzige Methode *messPunktAufnahme(...)*, die die Aufnahme eines neuen Messpunktes veranlasst. Jeder Messpunkt besitzt hierbei folgende Struktur:

50. Ein Container ist ein abstraktes Objekt, das Elemente des gleichen Typs beinhaltet.

```
class Messpunkt
{
    private int xKoordinate;            //X-Koordinate
    private int yKoordinate;            //Y-Koordinate
    private int zKoordinate;            //Z-Koordinate

    private Orientation orient;         //Orientierung des
                                        //Messpunktes (4 Richtungen)

    private string positionsName;       //der Positionsname
    private int positionsID;            //die PositionsID
    private ArrayList naechsteNachbarn; //Liste mit den nächsten
                                        //möglichen Nachbarn

    private MyAccessPoint myAccessPoint;//einzelnes Access Point
                                        //Objekt

    private LinkedList<MyAccessPoint> myApListe;//an dieser Position
                                                //gehörten AccessPoints

    ...
```

Access Point

Die Struktur des Access Points repräsentiert den Namen (*APName*) und die eindeutige MAC-Adresse (*MacAdresse*). Weiterhin beinhaltet die Struktur einen Signalstärkecontainer (*SignalListe*) vom Typ Double einer bestimmten Größe. Diese variiert je nach Dauer des Messzyklus und hält die Signalstärken zur späteren Analyse bereit. Dabei weist die Access Point Struktur folgende Form auf:

```
class MyAccessPoint
{
    private string APName;              //Access Point Name
    private string MacAdresse;          //Access Point MacAdresse

    private ArrayList SignalListe;      //Signalstärkeliste
                                        //des AccessPoints

    private double SignalDurchschnitt;  //Signalstärkedurchschnitt
    private double SignalMedian;        //Median der Signalstärke

    ...
```

Die beiden Variablen *SignalDurchschnitt* und *SignalMedian* werden zur Laufzeit aus dem Inhalt der *SignalListe* mit den jeweiligen Methoden *getDurschnittsSignalStaerke(...)* und *getMedianSignalStaerke(...)* bestimmt. Des Weiteren besitzt die Struktur *MyAccessPoint* eine Methode *AddSignal(...)*, die das Hinzufügen von einzelnen Signalsamples übernimmt.

6.4 Datensammlung

Eine Positionsbestimmung wird anhand von Signalstärkeinformationen der in der Umgebung empfangbaren Access Points vorgenommen. Durch die systembedingte Begrenzung des Abfrageintervalls vom Wireless LAN Adapter, ist ein Intervall kleiner einer Sekunde nicht möglich.

6.4.1 Signalstärketabelle

Die Basis der Positionierungssoftware stellt die Signalstärkedatenbank dar, die alle wichtigen Daten von den einzelnen Messpunkten enthält. Diese Daten werden in einem Container vom Typ Messpunkt (siehe Abschnitt 6.3.2) gespeichert und als Binärdatei abgelegt. Dadurch wird keine weitere Software, wie zum Beispiel ein Datenbanksystem benötigt, was den Grad der Autonomie erhöht. In der Trainingsphase (vgl. Abschnitt 6.4.2) wird für jeweils ein Operationsgebiet eine Signalstärketabelle angelegt. Die dort gewonnenen Daten können im Nachgang weiterhin bearbeitet und angepasst werden. Durch die eigens dafür entwickelte Messpunktanalysesoftware, bei der es sich um eine Desktopanwendung handelt, können die Signalstärketabellen komfortabel bearbeitet werden (siehe Anhang B).

6.4.2 Trainingsphase

Die Trainingsphase wird mithilfe des Positionierungsfensters (siehe Abschnitt 6.2.3) und mit dem angezeigten Grundriss des Operationsgebietes durchgeführt. Durch das Markieren der vorbestimmten Position im Fenster wird die Trainingsphase gestartet und die aktuellen Signalstärkeinformationen aufgezeichnet. Die nachfolgende boolesche Funktion veranlasst die Aufnahme eines Messpunktes, wobei ein Container *apListe* mit den aktuell empfangenen Access Points übergeben wird.

```
bool messPunktAufnahme(LinkedList<AccessPoint> apListe)
```

Der Rückgabewert *true* der Funktion signalisiert die korrekte Aufnahme, dementsprechend *false* das Scheitern. Der entsprechende Quellcodeausschnitt befindet sich im Anhang C.

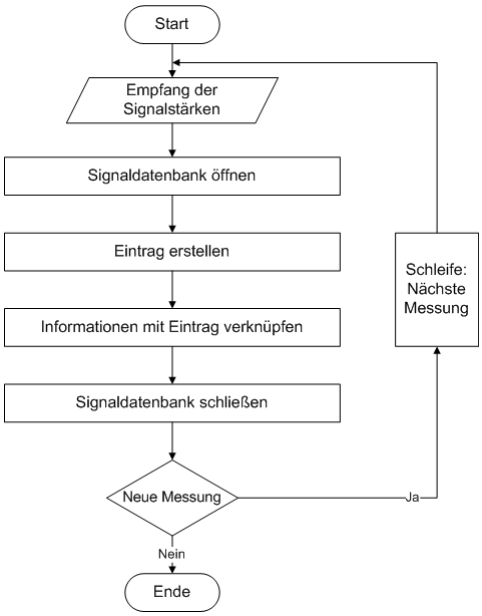

Abb. 6.6: Ablaufdiagramm Trainingsphase

Wie die Abbildung 6.6 illustriert, werden die Informationen mit dem jeweiligen Eintrag verknüpft und abschließend wird die Datenbank geschlossen. Die Trainingsphase wiederholt sich für weitere vorbestimmte Positionen, bis die Abbruchbedingung erfüllt ist. Ist dies der Fall, wird die Phase abgeschlossen und eine Benutzung der Daten kann erfolgen.

6.4.3 Positionierungsphase

Nach erfolgreicher Trainingsphase steht für die Positionierungsphase eine Signalstärketabelle zur Verfügung. Im Positionierungsfenster (siehe Abschnitt 6.2.3) kann diese über den Button *L* geladen und dem System bekannt gemacht werden. Zusätzlich kann ein Layout des Operationsgebietes mit dem Button *K* geladen und dargestellt werden. Ist dies erfolgt, wird über den Button *P* die Positionierungsphase gestartet. Die dazugehörige Funktion *PositionsBestimmung(...)* ist wie folgt deklariert:

```
Point PositionsBestimmung(LinkedList<AccessPoint> apliste,
                          PosAlgorithmus algorithm)
```

Als Übergabeparameter erhält die Funktion einen Container (*apliste*) mit den aktuell empfangenen Access Points und den zuverwendenden Positionierungsalgorithmus (*algorithm*) vom Typ Enumeration. Als Rückgabewert werden die ermittelten Koordinaten vom Typ Point[51]

51. Der Typ Point bildet die Koordinaten *x* und *y* ab.

ausgegeben. Im Anhang C befindet sich der dazugehörige Quellcodeausschnitt.

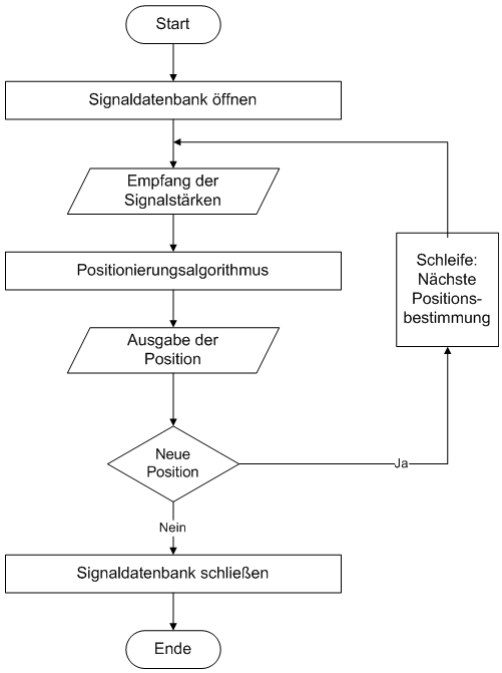

Abb. 6.7: Ablaufdiagramm Positionierungsphase

Das Diagramm aus Abbildung 6.7 gibt den Ablauf der Positionierungsphase wieder. Dabei wird nach dem Aktivieren die Signaldatenbank geöffnet und dem System übergeben. Folgend werden die aktuellen Signalstärkeinformationen aufgenommen und dem ausgewählten Positionierungsalgorithmus übergeben. Dieser berechnet die Position und gibt diese an das System zurück. Diese Abfolge der Positionsbestimmung wird so lang wiederholt, bis erneut der Button *P* betätigt wird. Dadurch stoppt die Positionierungsphase und schließt die Signaldatenbank.

6.5 Positionierungsalgorithmen

Zur Positionsbestimmung wurden in MaWIPS nachfolgend vier Positionierungsalgorithmen implementiert:

1. Nearest Neighbor in Signal Space (NNSS)
2. *k*-Nearest Neighbor in Signal Space (*k*-NNSS)
3. History Monitoring Algorithmus (HMA)
4. Multilayer Perzeptron (MLP)

6.5.1 Nearest Neighbor in Signal Space

Der Nearest Neighbor in Signal Space Algorithmus stellt den Basisalgorithmus für die folgenden Algorithmen dar. Die Abbildung 6.8 beschreibt die Implementation des NNSS Algorithmus.

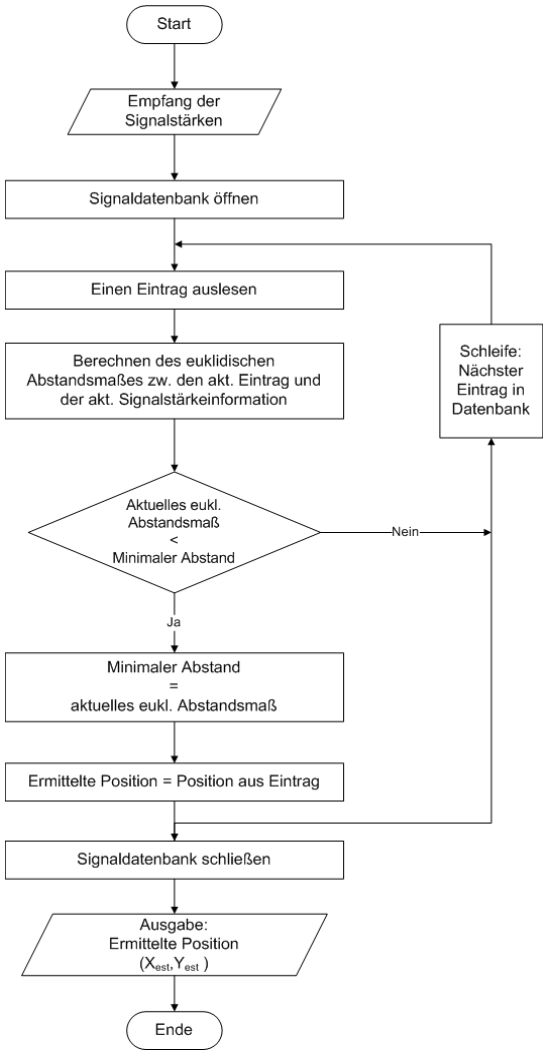

Abb. 6.8: Ablaufdiagramm Nearest Neighbor in Signal Space

Dabei stellt die Anzahl der zu durchsuchenden *n* Datentupel aus der Signaldatenbank die Komplexität des Algorithmus dar. Die folgende aufgeführte Funktion veranlasst die Berechnung der Position durch die Übergabe der aktuell empfangbaren Access Points (*apliste*).

```
Point NNSSAlgorithmus(LinkedList<MyAccessPoint> apliste)
```

Der Rückgabewert der Methode stellt die ermittelte Position vom Typ Point dar, wie es aus dem Quellcodeausschnitt im Anhang C hervorgeht.

6.5.2 *k*-Nearest Neighbor in Signal Space

Der auf den NNSS aufbauende *k*-Nearest Neighbor in Signal Space Algorithmus ist wie folgt implementiert:

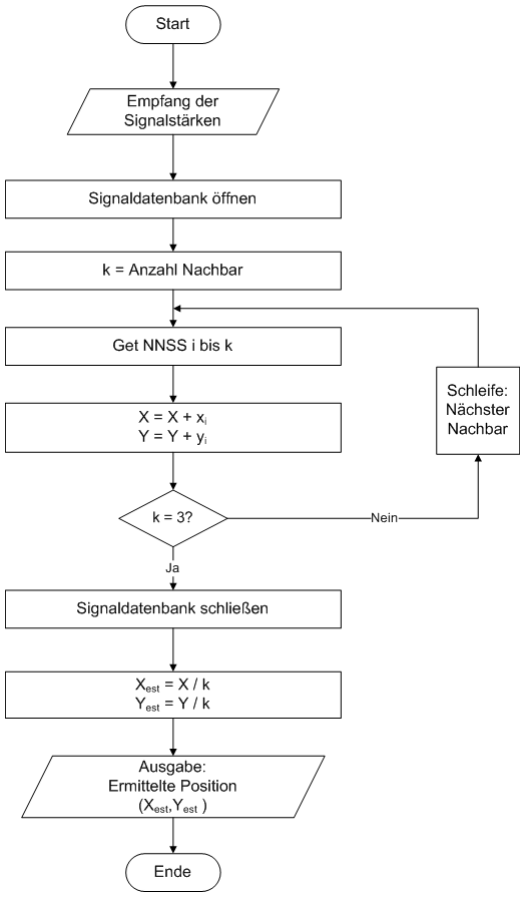

Abb. 6.9: Ablaufdiagramm *k*-Nearest Neighbor in Signal Space

Aus Abbildung 6.9 geht hervor, dass eine Abfrage (*Get NNSS i bis k*) eines nächsten Nachbarn durch den NNSS Algorithmus stattfindet. Dies wird so lang wiederholt, bis die gewünschte Anzahl von *k* nächsten Nachbarn erreicht ist. In diesem Fall wurde *k*=3 als Beispiel für drei Nachbarn gewählt. Wird die Schleife verlassen, folgt die Mittelung der Koordinaten und die anschließende Ausgabe der Position. Mit der nachstehenden Funktion wird die Berechnung (siehe hierzu Anhang C) angestoßen und die Position vom Typ Point zurückgegeben. Es wird eine Liste (*apliste*) mit den aktuell empfangbaren Access Ponts übergeben:

```
Point K_NNSSAlgorithmus(LinkedList<MyAccessPoint> apliste)
```

Die Komplexität ergibt sich aus den *k* Nachbarn und den zu durchsuchenden *n* Datentupeln aus der Signaldatenbank.

6.5.3 History Monitoring Algorithmus

Der History Monitoring Algorithmus baut wie der *k*-NNSS Algorithmus auf den Basisalgorithmus (NNSS) auf. Jedoch gestaltet sich der HMA komplexer, wie es das nachfolgende Ablaufdiagramm illustriert.

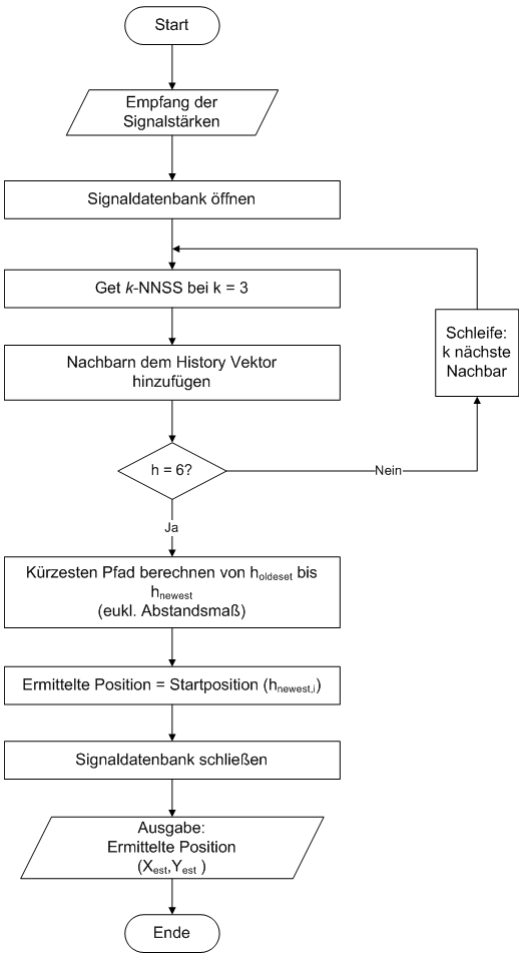

Abb. 6.10: Ablaufdiagramm History Monitoring Algorithmus

Hierbei greift der HMA auf die Abfrage (*Get k-NNSS k = 3*) der nächsten Nachbarn an der momentanen Position zurück. Diese ermittelten Nachbarn werden dem History Vektor bis zu einer Tiefe von sechs hinzugefügt. Anschließend wird der kürzeste Pfad beginnend vom ältesten bis zum neusten Eintrag berechnet. Der daraus resultierende Nachbar wird als momentane Position verstanden. Die Komplexität ergibt sich aus dem des *k*-NNSS und der Verzögerung durch die Berechnung des Pfades.

```
Point HMAlgorithmus(LinkedList<MyAccessPoint> apliste)
```

Die oben stehende Funktion startet den History Monitoring Algorithmus mit der Übergabe

der momentan empfangbaren Access Points (*apliste*) und gibt die Koordinaten vom Typ Point zurück. Im Anhang C befindet sich ein Ausschnitt der Wegberechnung als Quellcode.

6.5.4 Multilayer Perzeptron

Die Implementation des mehrlagigen Perzeptrons teilt sich in zwei Bereiche auf. Zum einen in den Trainingsteil und zum anderen in den Positionierungsteil. Das nachfolgende Ablaufdiagramm beschreibt den implementierten Trainingsprozess des MLPs:

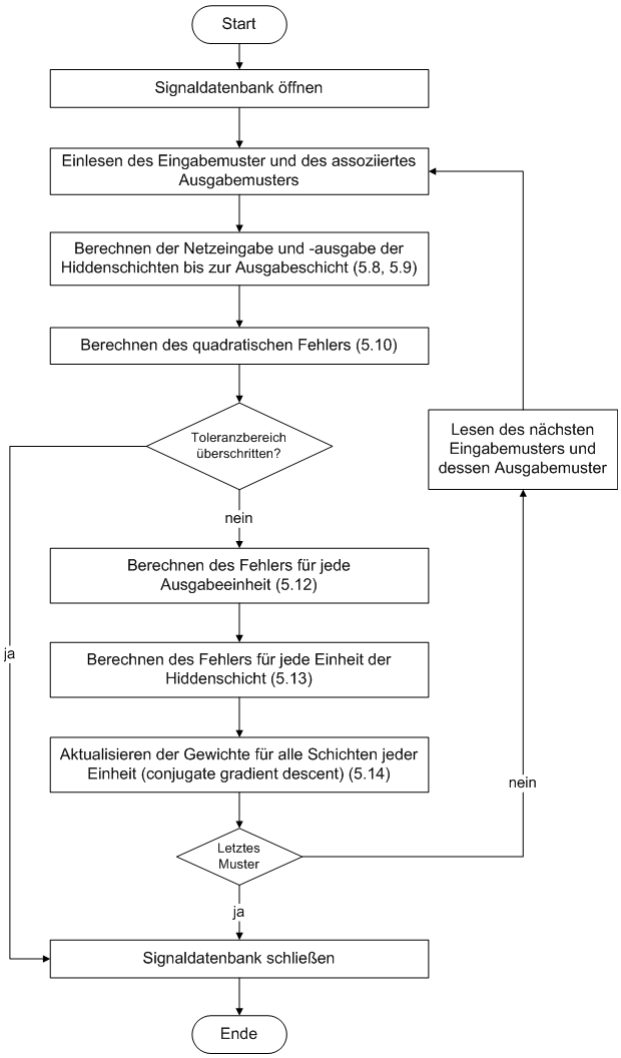

Abb. 6.11: Ablaufdiagramm Sub-Funktion Backpropagation-Algorithmus

Ist das mehrlagige Perzeptron trainiert, wird der Positionierungsprozess mit folgendem Funktionsaufruf gestartet (siehe hierzu Anhang C):

```
Point MultilayerPerzeptron(LinkedList<MyAccessPoint> apliste)
```

Hierzu wird der Funktion eine Liste (*apliste*) mit momentan empfangbaren Access Points übergeben und liefert bei Erfolg die ermittelte Position vom Typ Point.

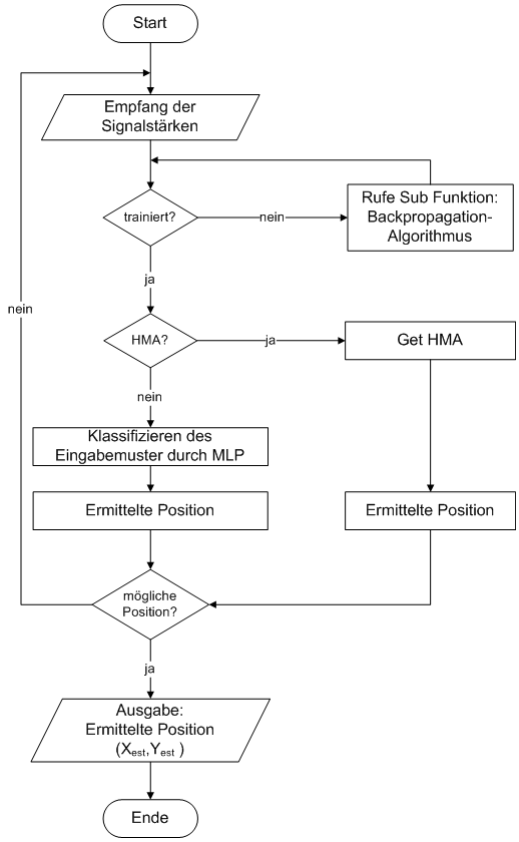

Abb. 6.12: Ablaufdiagramm mehrlagiges Perzeptron

Aus oben stehenden Ablaufdiagramm geht die Funktionsweise des mehrlagigen Perzeptrons hervor. Es kann durch das Setzen eines Flags die Kombination mit dem History Monitoring Algorithmus erfolgen. Dadurch kann eine ermittelte Position des MLP dahingehend überprüft werden, ob sich diese in der Nähe der vorangegangenen Position befindet und eine Verbindung zu dieser besteht.