Technische Universität München
Fakultät für Informatik

SEP

Smart Environment mit Spracherkennung

Bearbeiterin: Yun-Yi Lisa Wang
wangyu@in.tum.de

Aufgabensteller:	Prof. Michael Beetz, PhD
Betreuer:	Dipl. Inf. Matthias Wimmer
Abgabedatum:	

1. Einleitung

In dieser Ausarbeitung stelle ich mein System-Entwicklungsprojekt (SEP) vor. Sie ist in drei Kapitel aufgeteilt. Wir sehen im ersten Kapitel, warum dieses Projekt „sHome“ existiert und einen Überblick der Problemstellungen in meinem SEP. Im zweiten Kapitel geht es um die Grundlagen der Verwendungen in meinem SEP, was folgenden entspricht: zuerst Sprachverstehen, das den Schwerpunkt meines SEPs darstellt und zweitens Grundlagen über „sHome“ und OWL, die die Basis meines SEPs sind. Im dritten Kapitel werden zwei Teile der Implementierung, Sprachverstehen und Agenten beschrieben. Im vierten Kapitel werden die Probleme bei der Implementierung zusammengefasst. Zum Schluss wird noch auf die Erweiternsmöglichkeiten hingewiesen.

1.1 Motivation

Heute zu Tage kann man mit vielen Computer Anwendungen über Standardeingaben, wie Tastatureingaben, Mauseingaben usw. sehr bequem die notwendigen Aufgaben durchführen. Idealerweise wäre dies so möglich wie dies heute schon in einigen Filmszene vorkommt: Ein großer zentraler Rechner bietet immer die geforderten Informationen, beantwortet unsere Fragen, und führt unsere Befehle aus. Um die verschieden Geräte zu bedienen, indem man durch mündliche Sprache mit dem Rechner kommuniziert und der Rechner gleichzeitig die Umgebung überwacht. Deswegen bauen wir eine ähnliche Umgebung auf. Diese sieht so aus, im Büroalltag werden die Informationen für jede Person individuell mit Zeit- und Ortseingaben behandelt und ihr Tagesablauf wird unterstützt durch die Mithilfe des Systems.

In diesen Problemstellungen sind viele Bereiche in Informatik beteiligt, z.B. eine Wissensbank, das Sprachverstehen für mündliche Eingabe, Bilderverstehen für grafische Eingabe und viele Agenten in dem verteilten Anwendungsbereich.

1.2 Überblick

Das ganze Projekt heißt „sHome“. Eine Umgebung auf Basis der Diplomarbeit „Grid-basierte Multiagenten-Plattformen für sensorbasierte intelligente Umgebungen“ von Markus Tröscher ist durch die Verwendung einer Wissensbank, siehe Kapitel OWL, von viele Agenten und einigen anderen unterstützenden Systembestandteilen erstellt worden. Die Aufgabe meines SEPs ist die Erweiterung des Systems mit der sprachbasierten Steuerung. Dies betrifft viele verschiedene einzelne Themenbereiche, die dafür zusammengestellt werden mussten, und zwar das Sprachverstehen, die Agenten für das Empfangen und das Stellen von Fragen oder Befehlen, die Agenten für das Abholen der Informationen von der Wissensbank sowie für das Ausführen eines Befehls.

2. Grundlagen

Die betreffende Themenbereiche in meinem SEP sind „Sprachsteuerung“, „sHome“ und „OWL“. Sie werden in diesem Kapitel vorgestellt.

2.1 Sprachsteuerung

Das Steuern der natürlichen Sprache ist eine wichtige Entwicklung im Bereich Informatik. Mit mündlicher Sprache eine Eingabe zu ermöglichen, ist eine sehr bequeme Art, Texte oder Befehle einzugeben und dafür wird das Sprachverstehen verwendet. Das Sprachverstehen wird in zwei Aufgabenbereiche unterteilt, die in unterschiedlichen wissenschaftlichen Teilgebieten bearbeitet werden.

1. Das Verstehen gesprochener Sprache, also die Spracherkennung oder Automatische Spracherkennung, wird als ein Problem der Mustererkennung verstanden, wobei hier die Stimme im Vordergrund steht.
2. Das Verstehen geschriebener Sprache wird in der Künstlichen Intelligenz und in der Computerlinguistik untersucht, wobei hier die Bedeutung der Sprache und die Sprachproduktion selbst im Vordergrund stehen. Dieses wird als Sprachverstehen bezeichnet.

Diese beiden Aufgaben sind nicht unabhängig voneinander, sondern das Verstehen geschriebener Sprache baut auf das Verstehen gesprochener Sprache.

2.1.1 Automatische Spracherkennung

Die Basis der Sprachsteuerung ist die automatische Spracherkennung. Eine automatische Spracherkennung ist ein Prozess, der eine Lautsprache durch Computerbearbeitung in eine symbolische Sprache (Text) verwandelt.

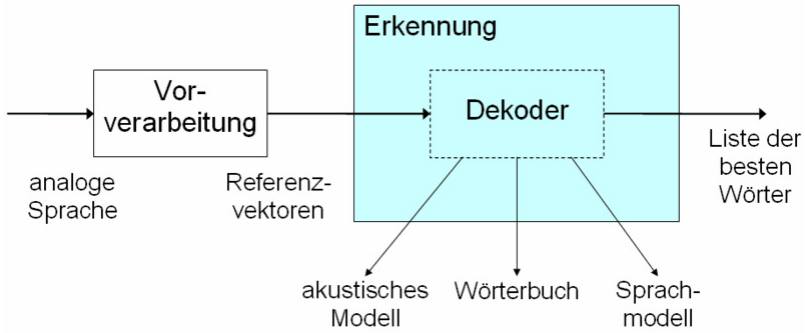

Abbildung 2.1 Aufbau eines Spracherkennungssystems[1]

Der Rechner nimmt zuerst die Akustik über das Mikrofon auf. Diese Aufnahme wird durch Tiefpass und A/D-Wandler bearbeitet. Damit werden ihre Messwerte aus den analogen Sprachsignalen gewonnen. Diese digitalisierte Aufnahme ist die Quelldaten. Siehe Abbildung2.1. Die einzelnen Messwerte wurden in gleichen zeitlichen Abständen bestimmt. Sie werden auch Abtastwerte oder Samples genannt. Nach dem Shanon-Theorem[7] sollte man darauf achten, dass die Abtastrate mindestens doppelt so groß sein muss wie die höchste vorkommende Frequenz im abgetasteten Signal. Diese Bedingung muss durch einen analogen Tiefpass vor dem A/D-Wandler gesichert sein. Ansonsten kommt es bei der Reproduktion des Signals zu Verzerrungen. Heute zu Tage verwendet man fast immer Standard-Sound-Karten im Rechner als kombinierter Tiefpass und A/D-Wandler. Das Sprachsignal ist typischerweise im Bereich von 80 – 6000 Hz angesiedelt. Deswegen ist eine Abtastfrequenz von 16 kHz für eine Aufnahme mit hoher Qualität ausreichend.

Aus den Quelldaten werden Vektoren mit verschiedenen Merkmalen extrahiert, z.B. Lautstärke (physiologisch), Energie in spektralen Bändern (physikalisch), Nulldurchgangsrate, Grundfrequenz, Formantlagen usw. Und alle 10 Millisekunden wird mit einer Fourier-Transformation ein Frequenzspektrum erzeugt. Dieses Spektrum wird dann an den Erkennungsprozess weitergegeben. Dieses Spektrum wird mit den gesamten Daten von vorgebereiteten Mustern innerhalb des Erkennungsprozess verglichen und die Symbole des ähnlichsten Referenzvektors werden als Ergebnis ausgegeben.

Die Vorbereitung des Erkennungsprozesses benötigt möglichst viele Musterdaten. Sie werden segmentiert und verschriftet d.h. mit den Phonemen[6] bzw. Wörter markiert. Dann benötigt die Spracherkennung einen Algorithmus, damit es berechnen kann, wie wahrscheinlich ist, dass eine bestimmte Folge von Merkmalsvektoren durch ein bestimmtes Wort erzeugt wurde. Es gibt drei wichtige Algorithmen: Klassische Mustererkennung mit DTW (Dynamic Time Warping), Hidden-Markov-Modelle (HMM), Künstliche neuronale Netze (ANN). In der Praxis verwendet man zurzeit meistens Hidden-Markov-Modelle. Künstliche neuronale Netze werden oftmals nur als Bestandteil von „klassischen“ Verfahren wie HMM verwendet. Der verwendete Spracherkenner in diesem Projekt ist also die Hidden-Markov-Modelle Algorithmus. Die Wahrscheinlichkeit wird für alle Wörter berechnet. Das erkannte Wort ist mit der Wahrscheinlichkeit.

Hidden-Markov-Modelle Algorithmus kann auf mehrere Ebene angewendet werden. Wenn man einen ganzen Satz erkennen will, benötigt man außer dem Wörterbuch, das Wörter und Phoneme enthält, noch eine Grammatik. Man kann zusätzlich noch ein Sprachmodell einsetzen, z.B. das Bi- / Trigramm Sprachmodell, die Wahrscheinlichkeit bestimmter Wortkombinationen zu bestimmen. Dadurch wird die falsche Hypothese ausgeschlossen und erhöht die Wahrscheinlichkeit der Wörter. Aus einem Bi- / Trigramm Sprachmodell wird die Auftrittswahrscheinlichkeit der Wortkombinationen aus zwei bzw. drei Wörtern erworben. Die Grammatik und die Sprachmodell haben gleiche Funktion, die Folge der Wörter zu bestimmen. Durch die akustisches Modell mit HMM werden Phonemen erkennen können. Dann setzen die Wörter anhand des Wörterbuchs aus Phonemen zusammen. Anschließend werden Sätze nach Sprachmodell mit einem großen HMM Automaten aufgebaut. Sieh Abbildung 2.2.

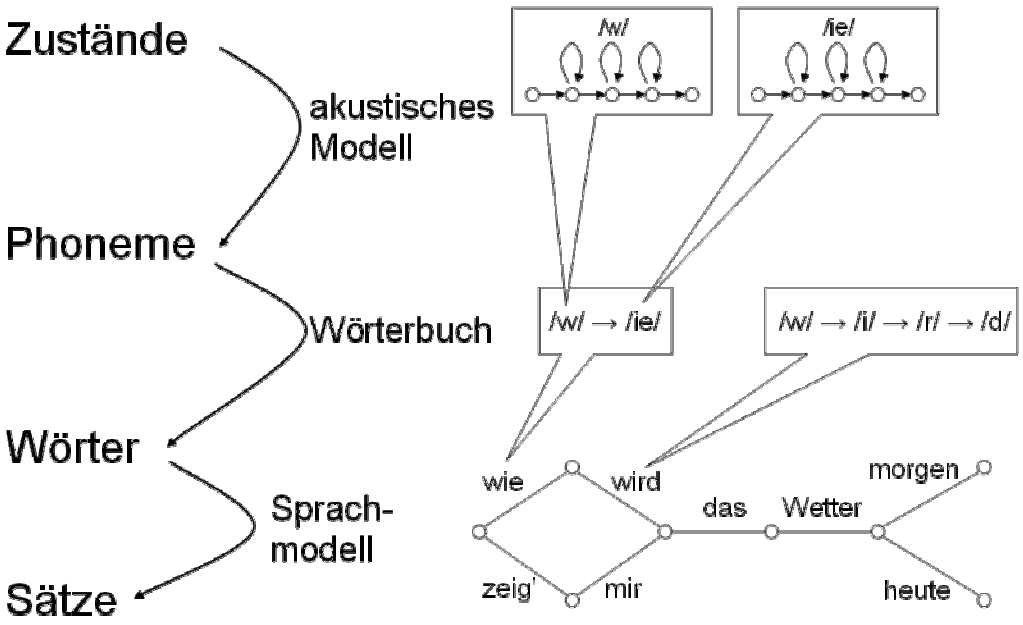

Abbildung 2.2 Modelle des Erkennungsprozesses[2]

Die Auswahl der Spracherkennung für dieses System ist Sphinx-4, die mit Java implementiert wurde. Diese Spracherkennung wurde in der Zusammenarbeit mit der Sphinx Gruppe in Carnegie Mellon Universität, Sun Mikrosystems Laboratorien, Mitsubishi Electric Research Labs (MERL) und Hewlett Packard (HP) entworfen und mit Spenden von der „Universität in California at Santa Cruz (UCSC)“ und „Massachusetts Institute of Technology (MIT)“ unterstützt[5]. Sphinx-4 ist ein sehr flexibles, modulares und ansteckbares System. Damit kann Innovationen in dem Kernforschung der Hidden-Markov-Model Sracherkennungssystem hilfen. Es lässt verschiedenartige Problemstellungen der Spracherkennung leicht realisieren. Die gleiche Basis der Programmiersprache wie in der Systemumgebung „sHome“ erhält ist ein zusätzlicher Vorteil. Sphinx-4 ist der Spracherkenner in Englisch aufgebaut worden. Und die N-Gramme, die die Spracherkennung mitliefert somit nur für die englische Sprache ausgelegt werden kann. Deswegen kann dieses System nur englische Kommandos verstehen.

2.1.2 Sprachverstehen

Für die oben beschriebene Spracherkennung ist die Bedeutung der Sprache irrelevant. Aber bei dieser Anwendung soll ein System mit dem Sprachverstehen auf den Inhalt der Äußerung reagieren können, z.B. eine Frage zu beantworten. Das heißt, das Computerprogramm soll den Sinn einer Sprache verstehen.

Im Sprachverstehen ist das Hauptproblem die Mehrdeutigkeit. Natürliche Sprache ist oft mehrdeutig. Menschen drücken nicht immer alle Details aus und sprechen vielleicht auch ironisch. Wenn man das Problem lösen möchte, würde man die Prinzipien von dem Welt- oder Situationswissen benötigen. Zum Beispiel spricht man oft mit Pronomen. Solche Wörter werden von Menschen sehr einfach verstanden, welche Person oder welches Objekt im Bezug steht, aber nicht von Rechner. Das Problem liegt nicht mehr an Spracherkennung.

In unserem System benötigt man eine ausreichende Grammatik, das die möglichen Präsentationen enthält. Danach werden mehrere Kriterien gesetzt, damit das System mit verschiedenen Äußerungen passend reagieren kann.

2.2 sHOME

„sHome“ (smart Home, Office and Meeting Environment) ist ein System, das eine intelligente Umgebung schafft und verschiedene Einsätze ermöglicht. Dies ist eine Anwendung im Bereich der Verteilten Anwendungen, Semantic Web und OWL (Web Ontology Language). Das System soll ganz individuell jeden Benutzer dienen, z.B. eine automatische Identifizierung, eine persönliche virtuelle Arbeitsumgebung, Informationen durch Sprachausgabe aus der Wissensbank wie z.B. Terminbenachrichtigungen ermöglichen und die Arbeitsgeräte durch Sprache steuern zu können. Bis auf die Sprachsteuerung ist in diesem Projekt „sHome“ am Lehrstuhl für Bildverstehen und Wissensbasierte Systeme das System schon mit vielen Funktionen entwickelt worden. Die Erweiterbarkeit des Systems ist wichtig. Ansonst wird Schwierigkeit bei dem Nachbauen der Spracherkennung geben.

In der intelligenten Umgebung können die Komponenten im System miteinander einfach kommunizieren. Diese Komponenten können als

Agenten auf beliebige Rechner verteilt werden. Zur Verwirklichung wird das Agentensystem und auf deren Plattform „CoABS Grid“ von Java implementiert. Damit ist diese Umgebung plattformunabhängig. Die zum Austausch in „sHome-ACL“ spezifisch geformten Nachrichten sind CoABS Grid Messages. Damit wird die Kommunikations-Infrastruktur des verteilten Systems auf der Plattform „CoABS Grid“ aufgebaut und es können über die vorhandene Infrastruktur Nachrichten ausgetauscht werden. Natürlich muss auf jedem verwendeten Rechner das zur Ausführung der Plattform „CoABS Grid“ erforderliche Java Runtime Environment installiert werden und das erforderliche LAN zur Verfügung stehen.

Jeder im Rechnernetz verwendete Rechner wird als Sensoren und Effektoren genutzt. Darauf befinden sich Software-Komponenten zur Steuerung spezifischer Peripheriegeräte, z.B. Licht, Lautsprecher, Mikrofon, Fernseher usw., die man mit verschiedenen Programmiersprachen realisieren könnte.

Im sHome-Agentensystem teilen sich die Agenten in zwei Gruppen auf, „Interface-Agenten“und „System-Agenten“. Durch die Interface-Agenten kann ein Datenfluss zwischen der Umwelt und der intelligenten Umgebung entstehen. Die Interface-Agenten von verschiedenen Sensoren liefern die Daten aus der Umwelt über verschiedene Peripheriegeräte zu dem System und sie bekommen auch die Daten, die über verschiedene Peripheriegeräte wie z.B. Lautsprecher, Player in die Umwelt gelangen. So wird eine Schnittstelle zwischen Benutzer und das System ermöglicht. Alle anderen Agenten werden als „System-Agenten“ realisiert. Meistens haben die System-Agenten die Funktion, Befehle in geeigneter Form zu generalisieren und eintreffende Informationen weiterzuleiten. Davon gibt es zwei Ausnahmen – die Agenten „Dispatcher“ und „Knowledgebase-AccessorAgent“ – welche auch zu den System-Agenten gehören, aber als allein stehend zu betrachten sind. Der Dispatcher ist wie eine Zentrale. Alle Nachrichten werden zu ihm versendet und über die Filter im Dispatcher wird entschieden an welchen Agenten sie weitergeleitet werden sollen. Wenn die Nachrichten an den KBAccessorAgent gesendet werden, dann werden diese Nachrichten als Wissen in der KB (Knowledge Base) abgespeichert. Gleichzeitig werden die Nachrichten weitergeleitet, indem diese in die geeignete Form generalisiert werden und somit an beziehenden Agent gesendet werden.

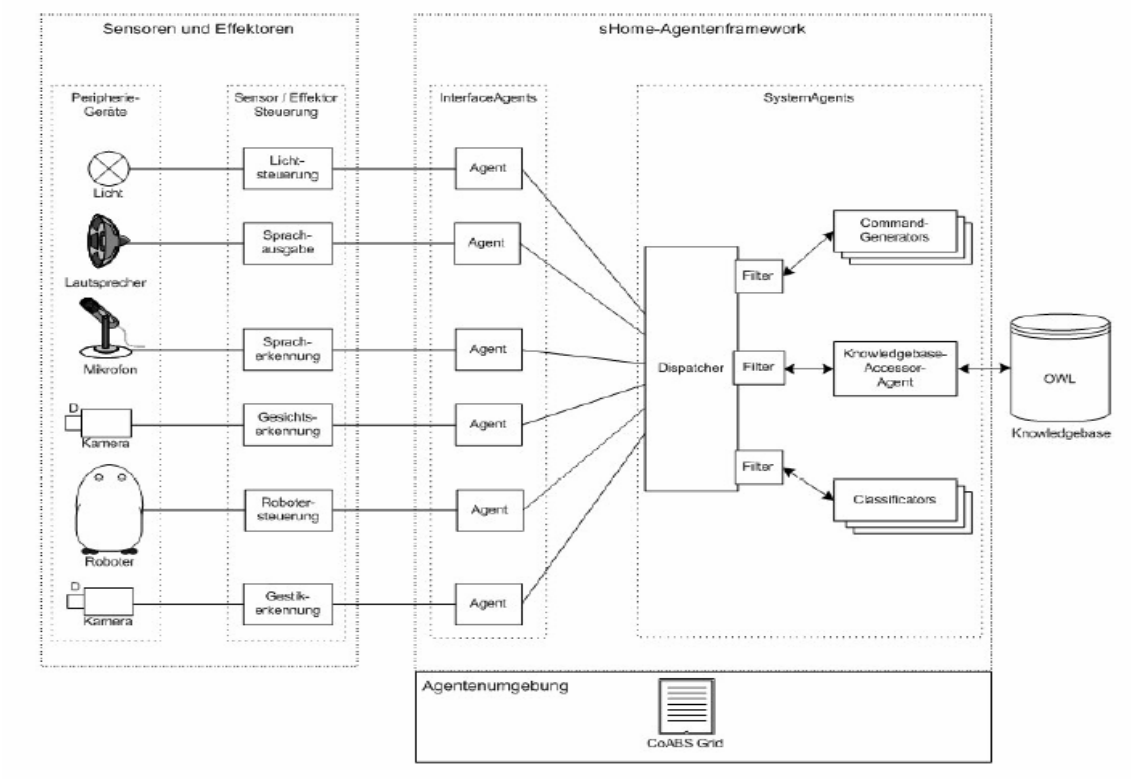

Abbildung 2.3 sHome-Agentensystem[4]

2.3 OWL

OWL ist eine Abkürzung für „Web Ontology Language“. Sie ist eine Spezifikation von W3C und besteht aus einer formalen Beschreibungssprache wie XML, um Ontologie erstellen zu können. Man will letztendlich die Objekte eines Bereichs und deren Beziehung formal beschreiben, damit die Software dieser verarbeiten kann, oder anders ausgedrückt, die Agenten diese verstehen können.

Ontologie ist die Lehre vom Sein, bzw. von den grundsätzlichen, allgemeinen, elementarsten, fundamentalen und konstitutiven Eigenschaften, den Prinzipien, den grundsätzlichsten Wesens-, Ordnungs-, und Begriffsbestimmungen des Seins. Mit dem Wissen der Realität bauen wir ein Modell der Realität, um unsere Probleme zu lösen. In der Informatik im Bereich der Wissenspräsentation versteht man unter Ontologie ein formal definiertes System von Konzepten und Relationen. Sie strukturiert in verschiedenen Bereichen und tauscht Daten aus, um die Bestandteile des Wissens zusammen zu fügen. Der bekannteste Definitionsversuch stammt

von T. Gruber. Dieser bezeichnet Ontologen als „explizite formale Spezifikation einer gemeinsamen Konzeptualisierung“[3].

Mit Ontologie kann eine Wissensbank aufgebaut werden. Somit gibt es Ähnlichkeiten zu einer Datenbank, aber doch anders als eine Datenbank. Außer Strukturen und Inhalte, die eine Datenbank bilden, bilden auch noch Regeln und Konzepte eine Ontologie. Während die Daten in der klassischen Datenbank keine Information über ihre Bedeutung besitzen, besitzen die auf Ontologie beschriebenen Daten eine formale Beschreibung sowie Regeln über ihren Zusammenhang. Durch diese Regeln können Rückschlüsse aus den vorhandenen Daten gezogen, Widersprüche in den Daten können erkannt und fehlendes Wissen selbständig aus den vorhandenen Daten ergänzt werden. Diese Rückschlüsse werden durch Inferenz, d.h. logisches Folgern abgeleitet. Unter ontologisches Lernen versteht man ein Prozess, bei dem eine Ontologie durch automatische Verfahren weiteres Wissen bekommt und dadurch weiter an Umfang und Struktur gewinnt. Dafür sind Inferenzen wichtig. Die Ontologie kann durch Relation über Relation und Regeln wegen der Komplexität in diesem Prozess selbst Wissen erzeugen. Obwohl man normalerweise das Wissen eingibt und die Möglichkeit kaum braucht, unterscheidet sich die Ontologie mit diesen Merkmalen von anderen Begriffssystemen.

Eine Ontologie besteht aus Konzepte, Instanzen, Relationen, besitzt die Eigenschaft Vererbung und erzeugt dabei Axiome. Ein Konzept ist eine Beschreibung der gemeinsamen Eigenschaften. Konzepte werden auch als Klassen bezeichnet. Konzepte/ Klassen in einer Klassenstruktur können zusammen mit Über- / Unter-klassen angeordnet werden. Instanzen werden anhand vorher definierter Konzepte erzeugt. Instanzen repräsentieren die Objekte in der Ontologie. Sie sind das Wissen. Instanzen werden auch als Individuals bezeichnet. Instanzen von gleichem Typen müssen an verschiedenen Konditionen angepasst werden. Zu diesem Zweck verwendet man Relationen. Relationen beschreiben, welche Beziehungen zwischen den Instanzen bestehen. Diese werden auch als Eigenschaften bezeichnet. In der Ontologie ist Vererbung möglich. Um Relationen und Konzepte zu vererben, werden alle Eigenschaften an das zu vererbende Element weitergegeben. Hier ist die Vererbung transitiv. Deswegen ist Mehrfachvererbung bei Konzepten grundsätzlich möglich. Dadurch können Instanzen in einer Bottom-Up-Hierachie aufgebaut werden,

welche man Delegation nennt. Axiome sind Aussagen innerhalb der Ontologie, die immer wahr sind. Diese werden normalerweise dazu verwendet, aus beziehenden Konzepten das Wissen abzuleiten, und das Wissen zu repräsentieren.

Hier ist ein Beispiel. Eine kleine Ontologie besteht aus drei Konzepten, Stadt, Land und Kontinent und beinhaltet auch die Relationen zwischen Städte, Zugverbindungen zwischen Nachbarstädten, sowie die Relation „ob zwei Länder Nachbarland sind“. Wegen der Zugehörigkeit ist das Land die Überklasse oder Oberklasse von der Stadt und Unterklasse von den Kontinenten. Umgekehrt ist die Stadt die Unterklasse von dem Land und dazugehörigem Kontinent. Und der Kontinent ist die oberste Klasse von den anderen beiden. Dann besitzt die Ontologie tausende von Städte, sehr viele Länder und fünf Kontinente als Instanzen. Durch die Vererbung bekommen wir leicht dem Axiom, „Es existiert zwischen Amerika und Europa keine Zugverbindung“.

Ontologie kann noch in zwei Typen unterteilt werden, Lightweight-Ontologien und Heavyweight-Ontologien. Lightweight-Ontologien enthalten Konzepte, Relationen zwischen Konzepten und Eigenschaften, welche diese beschreiben. Heavyweight-Ontologien erweitern die Lightweigth-Ontolotgien und fügen diesen Axiome und Einschränkungen hinzu, damit die beabsichtigte Bedeutung einzelner Aussagen innerhalb der Ontologie eindeutig festgelegt ist.

3. Implementierung

Das Ziel der Problemstellung ist, in der vorhandenen Umgebung mit der mündlichen Sprache das System „sHome“ zu steuern. Folgende Befehle und Fragen werden dem System jetzt zur Verfügung gestellt:

- What is the weather like in (Ortsname)?
- What time is it?
- When is the next meeting of (Name)?
- Turn/Switch on/off the player/TV.
- Play/Stop the song/movie
- Next/Previous chanel.
- Pause

Die ausgesprochenen Befehle sollen als Eingaben von dem Spracherkenner erkannt werden. Anschließend werden die erkannten Befehle zu Wissensbank geschickt und abgespeichert. Nach der Unterscheidung des Befehltyps werden die Informationen von der OWL-Wissensbank zurückgeliefert, wenn die Befehle als Abfragen angenommen werden, oder die Befehle werden direkt ausgeführt. Dafür sind folgende drei Teile zu implementieren:

„Sprachverstehen“, „Agenten“ und „OWL“.

3.1 Sprachverstehen

Wie in Kapitel 2 besprochen werden ein Wörterbuch und eine Sprachgrammatik benötigt. In Sphinx-4 befindet sich bereits ein Wörterbuch. Das Wörterbuch ist eine reine Text Datei. Diese Datei besteht aus zwei Spalten. Die Wörter in der linken Spalte sind unsere Sprache. Die Wörter in der rechten Spalte sind Phonemen. Folgende Tabelle 3.1 ist ein Ausschnitt aus dem Wörterbuch:

Wörter	Phoneme
at	AE T
ate	EY T
be	B IY
cheese	CH IY Z
cow	K AW
dee	D IY
eat	IY T
ed	EH D
fee	F IY
gee	JH IY
green	G R IY N
hide	HH AY D
he	HH IY
……	…….

Tabelle 3.1 Wörterbuch

Nimmt man beispielsweise das Wort „at“ aus dem Wörterbuch. Das Wort „at“ in dieser Datei besteht aus zwei Phonemen, „AE“ und „T“. Phoneme sind die kleinsten bedeutungsunterscheidenden, aber nicht bedeutungstragenden Einheiten einer Sprache[6]. Solche Phoneme gibt es in der englischen Sprache mit insgesamt 39 Stück. Siehe folgende Tabelle:

Phoneme	Beispiel	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

Tabelle 3.2 39 Phoneme aus englischen Sprachen und ihre Beispiele

In dem Wörterbuch gibt es mehr als 250 000 Wörter. Aber es müssen noch viele Wörter ergänzt werden, wie z.B. Namen und Ortsnamen.Außer diesem Wörterbuch brauchen wir noch eine Sprachgrammatik, die diese Kommandosprache beschreibt. In Sphinx-4 wird die Grammatik-Sprache „JSpeech Grammar Format (JSGF)“ verwendet.

3.2 Agenten

Es wurde zwei Agenten für das System „sHOME“ implementiert, „MicrophoneAgent“ und „MicrophoneControlAgent“.

Der Agent „MicrophoneAgent“ ist für Spracherkennung zuständig. Die Informationen von Städten und Personen müssen abgefragt werden können, deswegen muss ein Teil der Grammatik mit Städtenamen und Namen der Personen erstellt werden. Für die Vorbereitung der Spracherkennung schickt der Agent „MicrophoneAgent“ zwei Abfragen zur Wissensbank, um alle Städtenamen und alle Personnamen in der Wissensbank zu bekommen. Danach erstellt der Agent damit zwei Grammatik Dateien, um diese in die Grammatik zu importieren. In dem Grammatik File der Städtenamen sind nur die Namen der Städte enthalten. In dem Grammatik File der Personnamen scheint es komplizierter zu sein. Denn der Agent muss alle Situationen berechnen, wie eine Person gerufen werden könnte. Eine Person könnte mit oder ohne Anrede gerufen werden. Oder man nennt diese nur mit Vornamen oder sein vollständiger Name mit Anrede. Alle Möglichkeiten sind daher in der Datei zu finden. Letztendlich kann damit die Spracherkennung stattfinden. Der Agent lauscht über ein Mikrofon, um mündliche Sprache aufzufangen. Diese mündliche Sprache wird durch den Spracherkenner möglichst erkannt. Der Spracherkenner gibt einen Text als Ausgabe zurück. Diese Sprache wird später von dem Agent „MicrophoneControlAgent“ verarbeitet, indem der Agent „MicrophoneAgent“ eine Nachricht dem System schickt, die den Text beinhaltet, was gerade erkannt wird. So findet eine Übergabe der Sprache zwischen dem „MicrophoneAgent“ und „MicrophoneControlAgent“ statt. Ein Beispiel-Programmstück (siehe Anhang A) ist eine Abfrage für Namen in der Wissensbank. Dies wird auch noch später im Abschnitt OWL genauer angesprochen.

Die Aufgaben des Agents „MicrophoneControlAgent“ sind, die aufgefangenen Nachrichten zu verarbeiten, die von dem Agent „MicrophoneAgent“ erkannt wurden. Danach diese Nachrichten zu analysieren bezüglich welche Befehle oder Abfragen, die sie beinhalten, und anschließend diese Befehle auszuführen, z.B. ein Player zu starten, abzuspielen oder den Player wieder zu beenden. Wenn die Nachrichten Abfragen beinhalten, werden diese Abfragen wieder zur Wissensbank zurückgeschickt. Der „MicrophoneControlAgent“ bekommt die Informationen, die von der Wissensbank zurückgegeben wurde. Und dieser

Agent formuliert damit die ganzen Sätze für die Antworten. Manchmal wird das Werkzeug XSL für die Formulierungen benötigt. Dann werden diese Antworten der Abfragen über Lautsprecher ausgegeben.

Zum Beispiel, wenn man über das Wetter in einer Stadt fragt. Der Agent bekommt paar Informationen, „Stadtname“, „den Status des Wetters“, „die minimale Außentemperatur“ und „die maximale Außentemperatur“. Die Information über den Wetterstatus in der Wissensbank enthält nur folgende drei Zustände, „sunny“, „covered“ und „dry“. Der Agent formuliert den Satz, „In (Stadtname) the weather is dry„, wenn der Zustand in der Stadt als „dry“ bezeichnet wird. Und anhand der Temperaturen sagt der Agent auch aus, ob das Wetter kalt, sehr kalt, warm oder sehr warm ist. Das Beispiel-Programmstück siehe Anhang B.

3.3 OWL

In diesem System werden die Informationen über das Wetter und über die Meetings von den beteiligten Personen abgefragt. In dieser OWL-Wissensbank, befinden sich drei Domänen „die Organisation“, „die Calendarclock“ und „das Wetter“. In dem Bereich „Organisation“ finden wir die Daten von Namen der Professoren, der Assistenten und der an dieser OWL-Wissensbank angemeldeten Studenten, und von ihren Positionen, solche Daten, die wir brauchen, und noch mehr Daten, wie z.B. über die Lehrveranstaltungen. In dem Bereich „Calendarclock“ können wir die Meetingsinfotmationen finden, wann die Meetings stattfinden, ihre Beschreibungen und die sich an den Meetings beteiligten Personen usw. Und in den Bereich „Wetter“ finden wir die Informationen über das Wetter in verschiedenen Städten.

Folgendes ist der Struktur von einem Beispiel mit dem Domäne „Wetter“, siehe Abbildung3.1. Jeder Name eines Konzeptes steht in der ersten Zeile des Feldes. Darunter folgen die Eigenschaften des Konzeptes. Und die Pfeile zwischen zwei Konzepten bezeichnen die Eigenschaften, entsprechend ihren Relationen mit anderen Konzepten.

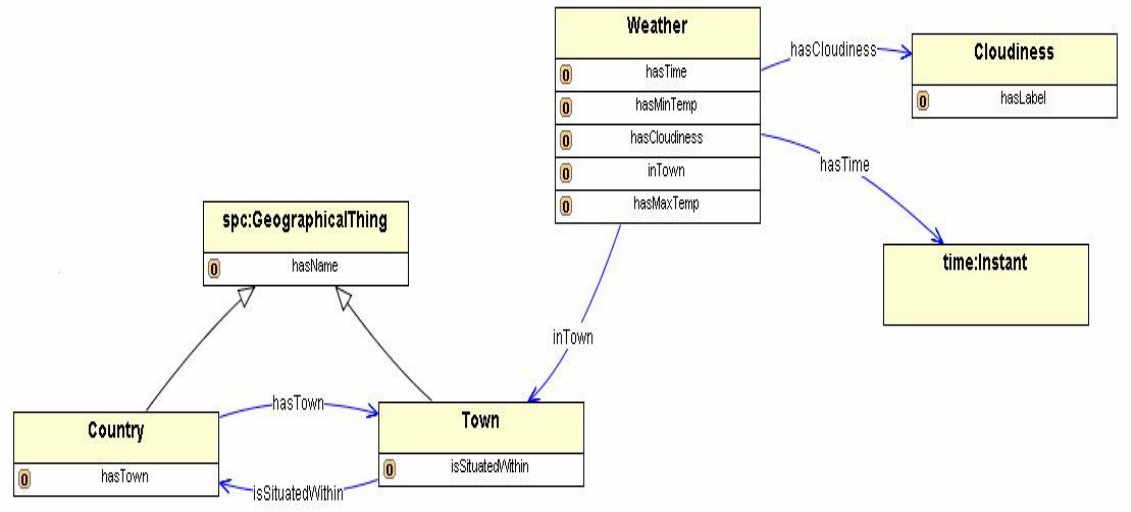

Abbildung 3.1 UML von dem Domäne „Wetter“[8]

Weil wir nur bestimmte Informationen haben wollen, benötigen wir ein Werkzeug, das die Informationen filtert, indem man diese OWL-Wissensbank mit KIF (Knowledge Interchange Format) abfragt. Bei einer Abfrage muss man zumindest „die Query“ und „die Knowledge Base URL“ angeben. Bei der KIF steht die Reinfolge der Parameter fest: „(Prädikat, Subjekt, Objekt)“. Das Anfragen wird eingegeben, indem man voraussetzende Relationen und eine Query mit den KIF-Prädikaten angibt. Wenn vor einem Parameter ein Fragezeichen in einem Prädikat steht, dann ist dieser Parameter die Variable, die die Information fordert. Das Programmstück für das Beispiel siehe Anhang A.

4. Resümee

Der Spracherkenner Sphinx-4 wurde auf Englisch aufgebaut und deswegen erkennt nur Englisch. Viele deutsche Laute gibt es nicht in Englisch. Deswegen können viele deutschen Namen nicht mit englischen Lauten beschrieben werden. Man muss die Phoneme der Namen im Wörterbuch manuell verbessern. Trotz diesen Verbesserungen werden die Phoneme oft nicht ganz korrekt sein. Das heißt, die Namen werden oft nicht oder falsch erkannt.

Was man machen könnte, ist eine deutsche Spracherkennung auf Java Plattform aufzubauen. Solcher Spracherkenner ist heute noch nicht zu finden. Oder man kann die deutsche Spracherkennung mit HTK benutzt,

ein Toolkit mit Hidden Markov Model welches in der Programmiersprache C realisiert wurde. Alternativ könnten die Benutzer auch die deutschen Namen auf Englisch aussprechen, was manchmal komisch klingt. Wenn die Namen keine englischen Namen sind, aber auf Englisch ausgesprochen werden sollen, muss man zuerst vereinbaren, wie diese deutschen Namen ausgesprochen werden müssen. Danach trägt man sie in die Liste ein. Diese Lösung ist aber unbeliebt.

In unserer Welt gibt es zwei Möglichkeiten, wie man als ein englischer Sprecher keine englischen Namen ausspricht. Meistens spricht man die Wörter nach der englisch-sprachigen Regel aus, oder man könnte auch die Namen nach der ursprünglichen Sprache aussprechen. Hier ein Beispiel mit dem Namen „Regina“. Die englische Aussprache des Namens ist anders als ihre deutsche Aussprache und ist wieder anders als ihre französische Aussprache. Da die Benutzer verschiedene Herkunft haben könnten, also verschiedene Muttersprache besitzen, könnte die Namen auch entsprechend anders ausgesprochen werden. Dann kommt es wieder zu dem Problem, wie es auch schon oben beschrieben wurde. Auf Französisch gibt es bestimmt auch wieder viele Laute, die nicht auf englischen Phonemen beschrieben werden können.

Dasselbe Problem tritt auch in die folgende Situation auf. Wenn die Agent „MicrophoneAgent“ vom System startet, werden die Daten von Wissensbank als erste neu generiert, und zwar die Namen von jedem Benutzer. Der Grund der neuen Generierung den Daten aus Wissensbank liegt an der Notwendigkeit der Grammatik, um diesen Spracherkenner starten zu können. Die Namen in der Wissensbank, die schon mal in dem Wörterbuch hinzugefügt wurden, können von dem Spracherkenner erkannt werden. Diese sind in Ordnung. Wenn die Namen in der Wissensbank sind aber noch nicht im Wörterbuch zu finden sind, dann führt dies zu dem Problem, dass das Wörterbuch, welches Basis der Grammatik ist, nicht vervollständigt werden kann. Das Problem kann also nicht optimal gelöst werden. Dies kann nur manuell gelöst werden, indem man die fehlenden Phoneme gleichzeitig in das Wörterbuch einfügt, sobald die neuen dazugehörigen Namen in die Wissensbank hinzugefügt wurden.

Bei der Spracherkennung gibt es eine bekannte Ursache, die zu einem Problem führen kann, und zwar die Auswahl des Mikrofons. Das Mikrofon nimmt unsere Stimme auf. Unsere Stimme wird danach digitalisiert. Das Mikrofon spielt in dieser Phase eine große Rolle, ob die Qualität der

Aufnahme angemessen ist. Eine Spracherkennung könnte schwierig funktionieren, trotz wir Menschen gut, richtig und klar gesprochen haben. Ein nicht angemessenes Mikrofon kann während der Aufnahme noch unnötig störende Nebengeräusche aufnehmen, oder es nimmt eine unscharfe Stimme auf. Deswegen kann man mehrere Mikrofone testen, wenn die Spracherkennung nicht richtig funktioniert. Hier ist noch eine merkwürdige Sache, und zwar die Frequenz der Stimme während der Spracherkennung. Die Frauen haben hellere Stimmen. Die Stimme einer Frau wird einfacher erkannt als die Stimme eines Mannes. Deswegen gibt es außer dem Aufbau des Systems noch mehrere Ursachen die Spracherkennung beeinflussen können.

5. Ausblick

Die Sprachsteuerung kann sich noch in eine der beiden folgenden Richtungen entwickeln: Man kann entweder die Kommando Sprache erweitern, um mehrere Funktionen zu implementieren, oder man kann einfach die Spracherkennung mit natürlicher Sprache bzw. den deutschsprachigen Spracherkenner wählen.

Das „sHome“ System kann schon viele Funktionen erfüllen. Trotzdem kann es bis jetzt viele Geräte noch nicht steuern, z.B. die Beleuchtung, die Heizung, die Klimaanlage, das Telefon, die Fax oder den Drucker usw. Des Weiteren kann man das System mit der Spracherkennung noch weiter entwickeln. Hier sind noch viele Befehle zur Steuerung der Geräte denkbar. Folgendes sind Beispiele von Befehlen zur Steuerung von weiteren Geräten, und für Anfragen von Positionen der Benutzer sowie vom Status der Geräte.

- Turn on/off the light/heater/air condition.
- Make the light darker/bright.
- I would like to call [somebody].
- I would like to dial the number [number].
- I would like to fax the [file] via the number [number]
- Can I speak with [somebody]?
- Can I leave a massage to [somebody]?
- Is the light on in [where].
- Is the heater on in [where].
- Where is [somebody]?

- Who is in [where]?
- Is [somebody] in [somewhere]?
- Do I have meetings?
- When are my meetings?
- Please print the file [Dateiname].

Um mehrere Befehle zu realisieren, muss man die Grammatik dieser Befehlsprache weiter und umfangreicher entwickeln. Die Zahlen für die Telefonnummer und die Nummer des Büros sind z.B. einzubauen. Die entsprechenden Telefone sind durch Festnetz verbunden, intern als auch extern. Dank der schnellen Entwickelung von „IP-Phone“ (VoIP) und Verwendung von kommerziellen Lösungen wie Skype, ist noch eine weitere (einfachere) Möglichkeit für das System entstanden.

Die Bedeutung jedes oben genannten Befehls, kann man immer freier ausdrücken. Irgendwann wird die Grammatik nicht mehr für Befehle gebaut, sondern für die natürliche Sprache. Man kann anstatt eines Kommando-Sprachemodells eine menschliche natürliche Spracherkennung verwenden. Es gibt Vorteile, wie z.B. die flexiblere Verwendung der Sprache, womit der Spracherkenner damit nicht nur auf die Form der Befehle beschränkt ist. Und man hätte keine weiteren Probleme und Schwierigkeiten mit der Aussprache der Namen. Falls man den deutschen Spracherkenner mit natürlicher Sprache nimmt, muss man sich weitere Gedanken machen über die Benutzer und deren zu erwartenden Dialogen. Ob es notwendig ist, Emotionen in das System zu integrieren, weil unsere Sprache menschlich ist? Man braucht wahrscheinlich ein komplexes Dialogsystem in Hintergrund und nicht nur Spracherkennung.

6. Referenzen

[1] Alexander Waibel, Aufbau eines Spracherkennungssystems (Abbildung)

http://de.wikipedia.org/wiki/Spracherkennung

[2] Modell des Erkennungsprozesses (Abbildung)

http://de.wikipedia.org/wiki/Spracherkennung

[3] Wolfgang Hesse, Informatik-Spektrum , Dez 2002

[4] Makus Tröscher , Grid-basierte Multiagenten-Plattformen für sensorbasierte intelligenten Umgebungen, Jan 2005

[5] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, Joe Woelfel, Sphinx-4: A Flexkble Open Source Framework for Speech Recognition, 2004

http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf

[6] Phil Hoole, Artikulatorische Phonetik; Exkurs zu Handout I - „Minimal-Phonologie“, Okt 2003

[7] Florian Schiel, Lehrskript von „Automatische Spracherkennung“, WS 02/03 Proseminar

[8] UML von Domäne „Wetter“.

http://wwwradig.informatik.tu-muenchen.de/ontology/weather-structure.uml.jpg

Anhang A

```
<owl-ql:query
xmlns:owl-ql=\"http://www.w3.org/2003/10/owl-ql-syntax#\"
xmlns:var=\"http://www.w3.org/2003/10/owl-ql-variables#\"
xmlns:iw=\"http://www.ksl.stanford.edu/software/IW/spec/iw.daml
#\" xmlns:tkb=\"http://www.daml.org/2001/03/daml+oil-ex#\"
xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"
xmlns:rdfs=\"http://www.w3.org/2000/01/rdf-schema#\"
xmlns:owl=\"http://www.w3.org/2002/07/owl#\"
xmlns:daml=\"http://www.daml.org/2001/03/daml+oil#\"><owl-ql:pr
emise><owl-ql:KIF>

( equals ?x ?x )
```

```
(&lt;=(rolle ?person ?roll)(|http://www.w3.org/1999/02/22-rdf-s
yntax-ns#|::|type| ?person ?roll))

</owl-ql:KIF>

</owl-ql:premise>

<owl-ql:queryPattern>

<owl-ql:KIF>

( or

  ( and

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    irstName| ?person ?firstname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    amilyName| ?person ?lastname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasG
    ender| ?person ?sex)

    ( rolle ?person ?roll)

    ( |http://www.w3.org/2000/01/rdf-schema#|::|subClassOf| ?ro
    ll
    |http://wwwradig.in.tum.de/ontology/organisation#|::|Profes
    sor|)

  )

  ( and

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    irstName| ?person ?firstname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    amilyName| ?person ?lastname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasG
    ender| ?person ?sex)

    ( rolle ?person ?roll)

    ( |http://www.w3.org/2000/01/rdf-schema#|::|subClassOf| ?ro
    ll
    |http://wwwradig.in.tum.de/ontology/organisation#|::|Assist
    ant|)

  )

  ( and
```

```
    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    irstName| ?person ?firstname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasF
    amilyName| ?person ?lastname)

    ( |http://wwwradig.in.tum.de/ontology/organisation#|::|hasG
    ender| ?person ?sex)

    ( rolle ?person ?roll)

    ( |http://www.w3.org/2000/01/rdf-schema#|::|subClassOf| ?ro
    ll
    |http://wwwradig.in.tum.de/ontology/organisation#|::|Studen
    t|)

  )

)

</owl-ql:KIF>

</owl-ql:queryPattern>

<owl-ql:mustBindVars>

<var:firstname/>

<var:lastname/>

<var:sex/>

<var:roll/>

</owl-ql:mustBindVars>

</owl-ql:query>
```

Anhang B

```
  private String weatherSentence(String city, String wStatus,
  String minT, String maxT){

      String sentence = new String();

        if (wStatus.equals("covered")){

         sentence = "In " + city + " the sky is covered.      ";

      }   else if (wStatus.equals("sunny")){

         sentence = "In " + city + " the sun is shining.      ";
```

```
   }   else if (wStatus.equals("dry")){

      sentence = "In " + city + " the weather is dry.      ";

   }

   /*if( mintemp > www ) It is hot.
   if( mintemp < xxx ) It is cold.
   if( mintemp > yyy ) It is very hot.
   if( mintemp < zzz ) It is very cold.*/


int min = new Integer(minT).intValue();

   int max = new Integer(maxT).intValue();

   if ( (min > 28)){

      sentence += "It is very hot.      ";

   }   else if ((min > 20) && (max < 30)){

      sentence += "It is hot.      ";

   }   else if ((min > 10) && (max < 20)){

      sentence += "It is cool.      ";

   }   else if ((min > 0) && (max < 10)){

      sentence += "It is cold.      ";

   }   else if ((max < 5)){

      sentence += "It is very cold.      ";

   }

   /*The temperature ranges from mintemp degrees to maxtemp
degrees centigrade.*/

   sentence += "The temperature ranges from " + min + " degrees
to " + max + " degrees centigrade";

   return sentence;

}
```