

Universität Passau  
Fakultät für Mathematik und Informatik  
Lehrstuhl für Informationsmanagement  
Prof. Dr. Burkhard Freitag  
Wintersemester 2004/2005

---

Programmierpraktikum zum Thema:  
**Observed Assessment**

---

Johannes Mager  
Matrikelnummer 37975  
mager@fmi.uni-passau.de

# Inhaltsverzeichnis

<b>1 Motivation</b>	<b>4</b>
<b>2 Ausgangspunkt – Software</b>	<b>6</b>
2.1 Assessment Center . . . . .	6
2.2 Multimodeltracker . . . . .	7
2.2.1 Mimik und Mimikerkennung . . . . .	7
2.2.2 Projekt . . . . .	8
2.2.3 Anforderungen . . . . .	8
2.2.4 Einsatz . . . . .	10
<b>3 Inhaltliche Konzeption</b>	<b>12</b>
3.1 Verschiedene Umsetzungsideen . . . . .	12
3.2 Mimik des Nutzers als zusätzliche Eingabe . . . . .	12
3.3 Steuerung durch Kopfgesten . . . . .	13
3.3.1 Allgemeine Funktionsweise . . . . .	13
3.3.2 Beantworten der Aufgaben . . . . .	14
3.3.3 Beurteilung . . . . .	16
<b>4 Softwaretechnische Konzeption</b>	<b>17</b>
4.1 Kommunikation via XML-RPC . . . . .	17

4.1.1	Grundlagen von XML-RPC . . . . .	17
4.1.2	Integration von XML-RPC in das Assessment Center . . .	18
4.1.3	Merkmale der Umsetzung mit XML-RPC . . . . .	19
4.2	Integration in Swing-Oberfläche . . . . .	20
4.2.1	Verwendete Konzepte . . . . .	20
4.2.2	Klassen und Methoden . . . . .	21
4.3	Erweiterungen des Assessment Center . . . . .	22
4.3.1	Konfigurationsmöglichkeit . . . . .	22
4.3.2	Mimikanzeige . . . . .	23
4.3.3	Demonstrations-Modus . . . . .	24
4.3.4	Auswertungsbildschirm . . . . .	25
<b>5</b>	<b>Beurteilung</b>	<b>26</b>
<b>A</b>	<b>Dokumentation</b>	<b>28</b>
A.1	Bedienung . . . . .	28
A.2	Einrichten des Assessment Centers . . . . .	30
A.3	Einrichten der Mimikererkennung . . . . .	30
<b>B</b>	<b>Bibliographie</b>	<b>32</b>

# Kapitel 1

## Motivation

Man kann heute – zumindest in unserer westlichen Welt – guten Gewissens davon ausgehen, dass der Umgang mit dem PC den meisten Menschen vertraut ist. Vielfältige Aufgaben des modernen Lebens setzen Computerkenntnisse voraus, sei es im Berufsleben, beim Einkaufen oder bei der privaten Nutzung. Die Computermaus ist im Zuge dieser Entwicklung neben der Tastatur zum Standardwerkzeug für die Kommunikation zwischen Mensch und PC geworden. Ihre Allgegenwart bei der Navigation im Internet, der Arbeit in Büro-Anwendungen oder dem Steuern von Computerspielen wird vom Anwender kaum noch in Frage gestellt. Fraglich ist jedoch, ob das Problem der Befehlsübermittlung zum Computer in vielen Fällen nicht intuitiver auf andere Art und Weise gelöst werden könnte.

Die immense Leistungsfähigkeit heutiger PCs öffnet die Tür für eine Vielzahl neuer Möglichkeiten. Sowohl Ton- als auch Bilddaten können unter Zuhilfenahme leistungsfähiger Rechenverfahren in Echtzeit analysiert und die darin enthaltene Information interpretiert werden. Spracherkennungsprogramme versprechen nach einer kurzen Trainingsphase eine Erkennungsquote von bis zu 99 % und ermöglichen die Eingabe auch längerer Texte ohne Zuhilfenahme anderer Eingabegeräte.

Im Hinblick auf eine reibungsfreie Kommunikation zwischen Mensch und Computer sind diese neuen Möglichkeiten sicherlich von großer Bedeutung. Denn von der Art und Weise, in der der Mensch seit frühesten Kindertagen seine Wünsche und Bedürfnisse zu kommunizieren lernt, ist der Umgang mit Tastatur und Maus bei genauerer Betrachtung meilenweit entfernt. Untersuchungen zufolge geschieht über 80 % der zwischenmenschlichen Kommunikation auf nonverbalem Wege und ein großer Teil hiervon wird vom Sender unbewußt übermittelt. Die gezielte Unterstützung der Sprache durch Mimik, Gestik und Körperstellungen hingegen ist ein fester Bestandteil jeder menschlichen Kultur.

Die vorliegende Arbeit untersucht den Einsatz einer Mimikerkennungs-Anwendung zur Steuerung des PCs. Im Rahmen eines Assessment Centers, einer au-

tomatisierten Lern- und Prüfungsplattform, soll die Interpretation der Benutzermimik als Möglichkeit der Mensch-Maschine-Interaktion betrachtet und ihre softwaretechnische Realisierung besprochen werden. Durch die sehr einfachen Befehlsstrukturen dieser Anwendung ergibt sich eine leicht realisierbare und zugleich intuitive Steuerungsmöglichkeit der Assessments, die die Verwendung von Maus und Tastatur überflüssig machen kann.

## Kapitel 2

# Ausgangspunkt – Software

### 2.1 Assessment Center

Die *Assessment.Suite* des *Institut für Informationssysteme und Softwaretechnik IFIS* der Universität Passau dient der Verwaltung, Durchführung und Auswertung interaktiver Übungen und Prüfungen. Für den Einsatz in unterschiedlichen Anwendungsfällen bietet das Softwaresystem verschiedene Modi: Im webbasierten Lernmodus wird der Benutzer vom System durch den Prüfungsstoff begleitet und hat Zugriff auf zusätzliche Hinweise und Übungen. Der Prüfungsmodus hingegen garantiert jedem Benutzer die gleichen Chancen für die Bearbeitung der Aufgaben. Fragen und Antwortmöglichkeiten können in zufälliger Reihenfolge angezeigt werden und ein mitlaufender Timer beendet den Test nach Ablauf der Bearbeitungsdauer automatisch.

Für die vorliegende Arbeit wurde aufgrund seiner weitreichenden Interaktionsfähigkeiten der für den Prüfungsmodus entwickelte Java-Client der *Assessment.Suite* erweitert, der im folgenden als das *Assessment Center* bezeichnet werden soll.

Die Oberfläche dieses Clients wurde mit Hilfe der von *Sun Microsystems Inc.* entwickelten Grafikbibliothek *Swing* gestaltet, der Standard-Bibliothek für grafische Benutzeroberflächen in Java. Als Bestandteil der *Java Foundation Classes* stellt Swing die nötigen Oberflächenkomponenten sowie Mechanismen zu deren Kommunikation und Steuerung zur Verfügung.

Die Architektur des Assessment Center folgt den Prinzipien des *Model-View-Controller*-Entwurfsmusters. Die Hauptaufgaben der drei Komponenten sind hierbei klar definiert:

Das *Model* repräsentiert den Zustand des gerade ablaufenden Assessments und enthält die einzelnen Aufgaben. Zusätzliche Aufgaben wie das Auslesen der Dateien von der Festplatte oder das Speichern des Resultats sind ebenfalls Aufgabe des Models.

Die *View*-Komponente hingegen kümmert sich um die Darstellung der Daten des Models. Aufgabe des *Controllers* ist die Entgegennahme und Weiterleitung von Benutzereingaben sowie die Steuerung des Programmablaufs.

In der Implementierung des Assessment Centers sind View und Controller, wie in Java Swing üblich, sehr eng miteinander verwoben. Dieses auch *Model-View-Presenter* genannte Entwurfsmuster vereinfacht die Programmierung der Oberfläche, da die komplexe Kommunikation zwischen View und Controller stark reduziert werden kann.

## 2.2 Multimodeltracker

### 2.2.1 Mimik und Mimikerkennung

Als Mimik bezeichnet man die sichtbaren Bewegungen der Gesichtsoberfläche, die besonders durch die Augen und den Mund als die beweglichsten Teile des menschlichen Gesichts geprägt sind. Gesteuert wird dies durch die mimische Muskulatur, zu der man elf der insgesamt 26 Gesichtsmuskeln zählt. Ihr kompliziertes Zusammenspiel ist die Ursache für die Emotionen, die wir in einem jahrelangen Übungsprozess aus den Gesichtern anderer Menschen abzulesen lernen.

Veranschaulicht man sich, wie viele Fehler selbst erwachsene Menschen beim Einschätzen der Gefühle ihres Gegenübers machen, ist es leicht vorstellbar, wie schwer es ist, die Mimik des Menschen meßbar zu machen und die damit verbundenen Emotionen zu generalisieren. Ein großer Durchbruch war das von den amerikanischen Psychologen Paul Ekman und Wallace Friesen entwickelte *Facial Action Coding System FACS* [Ekman1978]. Ziel dieser Entwicklung war es, ein zuverlässiges und objektives Kodiersystem zur Erfassung des mimischen Ausdrucks zur Verfügung zu stellen. Grundlage von FACS sind die sogenannten *Action Units* – kleinste Bewegungseinheiten, mit denen ein mimisches Erscheinungsbild kodiert wird und die aus dem Zusammenspiel einzelner Teile der mimischen Muskulatur entstehen. Mit ihrer Hilfe können alle beobachtbaren Bewegungen im menschlichen Gesicht erfasst und genau beschrieben werden.

Die Klassifikation des Gesichtsausdrucks ist bei der automatisierten Mimikerkennung am Rechner jedoch nur einer der notwendigen Schritte. Laut Pantic und Rothkrantz [Pantic2000] lässt sich die dazu notwendige Vorverarbeitung in zwei große Schritte einteilen: Die Detektion des Kopfes und die Extraktion der Gesichtszüge. Da das menschliche Minenspiel zudem zeitabhängig ist, muß die Bewegung des Gesichts über einen kurzen Zeitraum beobachtet und analysiert werden. Diese Analyse über eine Sequenz von Einzelbildern hinweg ermöglicht zudem die Detektion von Kopfgesten wie Kopfschütteln und Nicken.

### 2.2.2 Projekt

Die Mimikerkennungs-Software *Multimodeltracker* ist ein Teil des Projekts *Sipbild: Mimik- und Gestikererkennung in Videobildfolgen* im Bayerischen Forschungsverbund für *Situierung, Individualisierung und Personalisierung in der Mensch-Maschine-Interaktion FORSIP*.

Entwickelt wird es derzeit am Lehrstuhl Informatik IX der TU München von Dipl. Inf. Matthias Wimmer im Rahmen seiner Promotion. Vollständig autonom ablaufend analysiert der Multimodeltracker das Kamerabild in Echtzeit, sucht nach einem (eventuell vorhandenen) Gesicht bzw. extrahiert und klassifiziert dessen Gesichtszüge. Die Mimikererkennung soll robust gegen Beleuchtungsänderungen und Bewegungen im Hinter- und Vordergrund ablaufen und eine genaue Detektion auch bei besonderen Gesichtsbestandteilen wie Brille, Bart etc. ermöglichen. Das Ziel des Projektes ist, dass durch diese Eigenschaften der Multimodeltracker auch außerhalb von Laborräumen im realen Betrieb eingesetzt werden kann.

Das zugrundeliegende Gesichtsmodell wird in [Wimmer2005] beschrieben. Es entstand durch die Integration mehrerer hundert unterschiedlicher Bilder von menschlichen Gesichtern verschiedener Hautfarbe, Alters und Geschlechts. 134 fest definierte Gesichtspunkte wurden in jedem dieser Bilder markiert und bilden zusammen die Grundlage für das *Point Distribution Model*.

Das Einpassen eines detektierten Gesichts in dieses Modell basiert auf der Kombination zweier verschiedener Techniken: Der Einsatz von Kantendetektion und -extraktion wurde um eine Analyse des Grauwertprofils ergänzt, um die Detektionsgenauigkeit zu erhöhen. Ein zusätzlicher Tracking-Algorithmus ermöglicht die Verfolgung des Gesichts einschließlich der bestimmten Gesichtspunkte über die Sequenz der Bilder hinweg.

Für die Klassifikation der Mimik wurde eine eigene Methode entwickelt, die nicht auf dem bereits erwähnten Facial Action Coding System basiert. Stattdessen wird aus den Informationen der Gesichtsveränderungen über mehrere Bilder hinweg mittels eines binären Entscheidungsbaums auf eine Emotion geschlossen. Bisher können neutrale, lachende und überraschte Gesichter, sowie die beiden Kopfgesten Nicken und Kopfschütteln erkannt werden.

Zur Kommunikation wurde der Multimodeltracker mit einer *RPC*-Schnittstelle ausgestattet, mittels derer die erkannte Emotion übermittelt wird. Andere Anwendungen können somit einfach und zeitnah auf die Emotionen des Benutzers reagieren und für die Interaktion von Benutzer und Computer diesen zusätzlichen Kanal nutzen.

### 2.2.3 Anforderungen

Für den effektiven Einsatz des Multimodeltracker ist der Einsatz eines leistungsstarken PCs dringend notwendig. So bewältigte der für diese Arbeit zur Verfü-

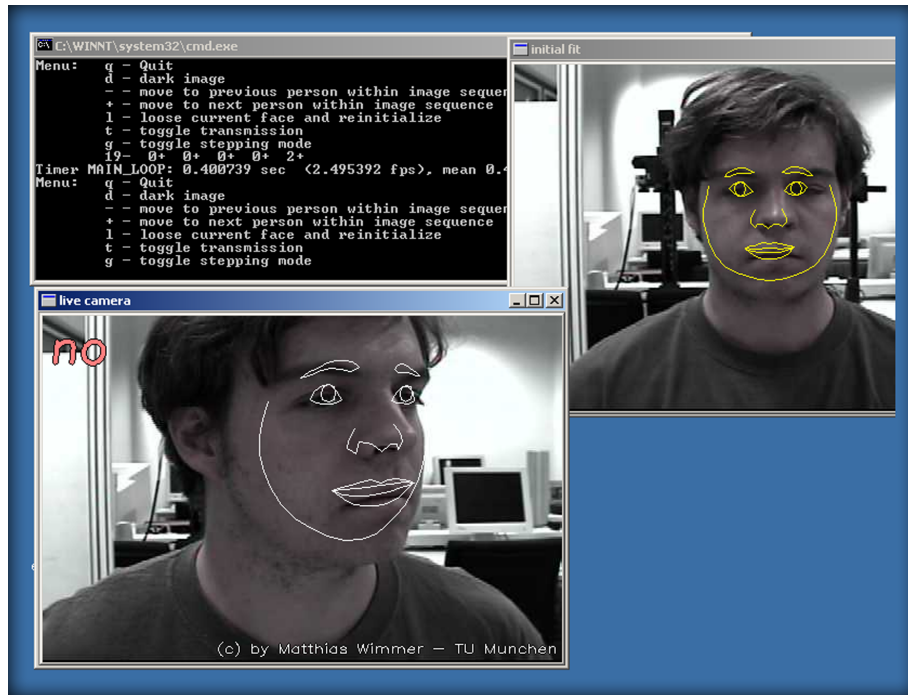


Abbildung 2.1: Die Programmoberfläche des Multimodeltracker

gung stehende PC mit Intel Pentium 4-Prozessor und 2,7 GHz einen Durchsatz zwischen 5 und 8 Bildern in der Sekunde. Folglich kann nur ein bestimmter Prozentsatz der von der Kamera gelieferten Bilder verarbeitet werden, während der restliche Teil unberücksichtigt bleibt. Die Detektion schneller Bewegungen ist somit nur begrenzt möglich. Um die Qualität dieser Detektion nicht noch weiter zu reduzieren, empfiehlt sich die Bereitstellung eines separaten PCs für die Berechnungen der Mimikererkennung. Durch den *XML-RPC-Client* können andere Programme problemlos per *Remote Procedure Call* über ein Netzwerk hinweg angesprochen werden.

Als Kamera stand für dieses Projekt eine handelsübliche Webcam zur Verfügung, die *Logitech QuickCam Pro 4000*. Mit einer Auflösung von 640 mal 480 Bildpunkten und einem Durchsatz von 15 Bildern in der Sekunde bietet sie eine ausreichende Qualität zur Verwendung in der Mimikererkennung. Eine professionelle Laborkamera, wie sie noch vor wenigen Jahren unverzichtbar gewesen wäre, ist weder im Hinblick auf die Güte der Mimikererkennungs-Ergebnisse noch für den Einsatz im echten Betrieb notwendig.

Der Multimodeltracker ist eine in der Programmiersprache *C++* geschriebene Applikation für das Betriebssystem *Microsoft Windows*. Zur Vorbearbeitung der Kamerabilder wird die Open Source Computer Vision Library *OpenCV* der Firma Intel Corporation genutzt, die auf dem PC vorhanden sein muß.

## 2.2.4 Einsatz

Der Multimodeltracker ist auf einem PC mit Microsoft Windows und der OpenCV-Bibliothek ohne Installation sofort einsatzfähig. Beim Start werden dem Programm sämtliche Optionen per Kommandozeile übergeben.

Die für die vorliegende Arbeit relevanten Parameter sind hierbei:

- source** *x* Ort der zu analysierenden Bilddaten:  
Ordner mit Bilddateien, avi-Video oder `livecam` für die direkte Verwendung der Daten einer angeschlossenen Kamera
- detectYesNo** Aktivieren der Detektion von Nicken und Kopfschütteln der sichtbaren Person
- detectEmotion** Aktivieren der Detektion und Klassifikation von Emotionen der sichtbaren Person
- initLoop** *x* Zeitaufwand für den initialen Einpass-Vorgang des erkannten Gesichts in das Modell des Multimodeltrackers
- delayFaceFound** *x* Anzahl der Frames, die zum Berechnen der Position des Kopfmodells betrachtet werden sollen
- xmlrpc** Aktivieren des integrierten XML-RPC-Clients
- host** *x* IP-Adresse des anzusprechenden XML-RPC-Servers – also der Anwendung, die die Emotions-Signale entgegennimmt
- port** *x* Port-Nummer des XML-RPC-Servers

Auf der Statuskonsole sind während des Programmablaufs weitere Eingaben möglich:

- l** Verwirft das erkannte Gesicht und reinitialisiert die Gesichtslokalisation
- d** Verdunkelt das Kamerafenster und zeigt lediglich die erkannten Konturen
- q** beendet den Multimodeltracker

Ist die `xmlrpc`-Option aktiviert, führt der Multimodeltracker bei jeder Detektion eines neuen Ereignisses einen Remote Procedure Call der Funktion `newMimicDetected` aus. Als Parameter sendet er die erkannte Mimik, kodiert als Integer-Wert. Dabei bezeichnet:

- 0** Der Benutzer schaut neutral
- 1** Der Benutzer lacht
- 2** Der Benutzer schaut überrascht

- 3 Der Benutzer hat sich von der Kamera weggedreht
- 4 Der Benutzer nickt
- 5 Der Benutzer schüttelt den Kopf

Tritt ein Benutzer in das leere Blickfeld der Kamera, wird die Funktion `newPersonDetected` aufgerufen. Verlässt er hingegen das Blickfeld, ruft der Multimodeltracker die Funktion `personWentAway` auf. Beide Funktionen besitzen keine Parameter.

## Kapitel 3

# Inhaltliche Konzeption

### 3.1 Verschiedene Umsetzungsideen

Es sind verschiedene Möglichkeiten denkbar, in einem per Computermaus zu bedienenden Programm eine Mimikererkennung zu integrieren.

Die wichtigste Frage ist zunächst, welche Rolle die Mimikererkennung spielen soll. Soll sie als alleiniges Eingabegerät die Computermaus komplett ersetzen und die Bedienung somit berührungsfrei ermöglichen? Oder ist das Ziel vielmehr, die Bedienung mittels Computermaus zu unterstützen und lediglich zusätzliche Funktionen wie Tipps oder Hinweise in Abhängigkeit von der Benutzermimik auszuführen?

Eine weitere Variable bei der Integration ist die Auswahl der Mimiksignale, die das Programm beeinflussen sollen und können. Soll das ganze zur Verfügung gestellte Spektrum an Mimiken berücksichtigt werden oder interessieren lediglich die Kopfgesten? Liegt der Fokus auf dem Erkennen von Emotionen des Benutzers, beispielsweise ob er verwundert ist oder lacht? Oder ist die einzige gewünschte Größe die Information, ob der Benutzer gerade anwesend und dem Bildschirm zugewandt ist? Für den Einsatz mit dem Assessment Center kamen im wesentlichen zwei verschiedene Nutzungsszenarien in Betracht, die im Folgenden vorgestellt werden sollen.

### 3.2 Mimik des Nutzers als zusätzliche Eingabe

Dieses Szenario entält die Steuerung des Assessment Centers auf die übliche Art und Weise mit Maus (und gegebenenfalls Tastatur). Die Mimikererkennung wird als zusätzlicher Eingabekanal interpretiert, der in bestimmten Situationen zu Programmreaktionen führen kann.

Zum Beispiel kann unmittelbar nach Präsentation einer Aufgabe auf Signale der Mimikerkennung geachtet werden. Macht der Benutzer in diesem Zeitraum ein überraschtes Gesicht, so ist davon auszugehen, dass er die Frage nicht oder nicht sicher beantworten kann. In diesem Fall kann das Programm den Benutzer mit einem zusätzlichen Hinweis unterstützen.

Vor allem im Übungsmodus, also in keiner strengen Prüfungssituation kann diese Möglichkeit durchaus sinnvoll sein, um dem Benutzer individuell zu helfen.

Nach anfänglichen Testfällen zu diesem Szenario wurde die Implementierung dieser Möglichkeit allerdings rasch verworfen. Die Ergebnisse der verwendeten Mimikerkennung erwiesen sich als zu unzuverlässig und zu wechselhaft um die verschiedenen Emotionen sicher zuordnen zu können.

### 3.3 Steuerung durch Kopfgesten

Da das Assessment Center vom Benutzer direkt vor dem Monitor durchgeführt wird, ist die wohl intuitivste Steuerungsart die Reaktion auf die beiden Kopfgesten Nicken und Kopfschütteln. Der Benutzer kann vor dem Bildschirm sitzen bzw. stehen bleiben und seine Befehle ohne große Anstrengung, und vor allem ohne Zuhilfenahme der Hände, übermitteln. Alle anderen möglicherweise erkannten Emotionen werden in diesem Modus ignoriert.

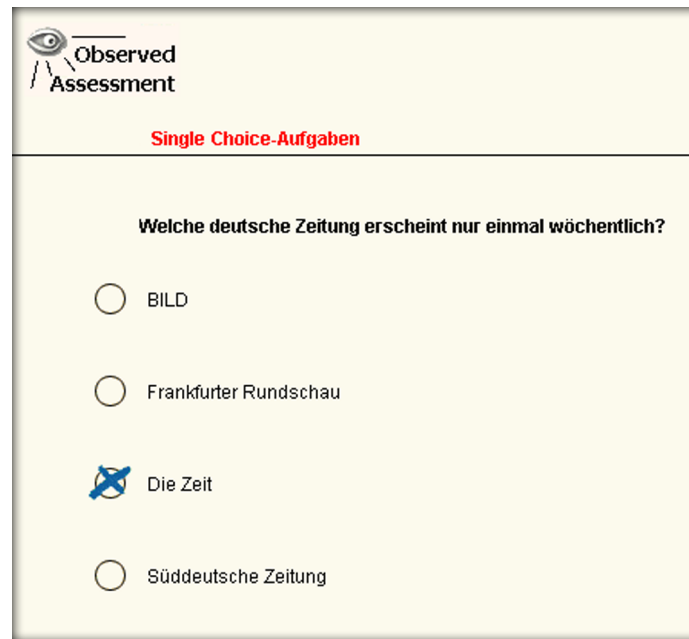
Der auf diese beiden Eingabemöglichkeiten reduzierte Bereich des normalen visuellen Kommunikationskanals ermöglicht bereits die Beantwortung der wichtigsten Aufgabentypen:

*Single- und Multiple-Choice-Aufgaben* können genauso beantwortet werden wie auch die vom Ablauf her etwas aufwändigeren *Pick-and-Place-Aufgaben*. Je nach Aufgabentyp ist hierbei für das Beantworten ein ein- oder mehrmaliger Schleifenlauf über die zur Auswahl stehenden Antworten nötig.

Die anderen von der Assessment.Suite unterstützten Aufgabentypen (*Freitext-Aufgaben*, *Graphische Markieraufgaben*, *Skalieraufgaben* und *Matrixaufgaben*) sind aufgrund ihrer komplexen Natur nicht bzw. nur sehr schwer durch Mimiksignale zu beantworten und werden aus diesem Grund im folgenden nicht weiter betrachtet.

#### 3.3.1 Allgemeine Funktionsweise

Sobald der Benutzer im Sichtfeld der Kamera erscheint, wird das Assessment gestartet und die Präsentation der Aufgaben beginnt. Verlässt der Benutzer das Sichtfeld der Kamera, wartet das Assessment Center auf das erneute Erscheinen eines Benutzers, bevor es mit der weiteren Aufgabenstellung fortfährt. Tritt erst nach einer längeren Zeit wieder ein Benutzer vor die Kamera, startet das Assessment von Neuem. Hat der Benutzer die letzte Frage des Assessment be-



The screenshot shows a user interface for 'Observed Assessment'. At the top left is the logo, which consists of an eye icon and the text 'Observed Assessment'. Below the logo, the text 'Single Choice-Aufgaben' is displayed in red. The main question is 'Welche deutsche Zeitung erscheint nur einmal wöchentlich?'. There are four radio button options: 'BILD', 'Frankfurter Rundschau', 'Die Zeit', and 'Süddeutsche Zeitung'. The 'Die Zeit' option is selected, indicated by a blue checkmark inside the radio button.

Abbildung 3.1: Beantwortung einer Single-Choice-Aufgabe

antwortet, präsentiert das Assessment Center das erzielte Ergebnis und schaltet nach einer bestimmten Zeit wieder um auf den Startbildschirm. Die verwendeten Zeitkonstanten sind hierbei frei konfigurierbar, um individuelle Einstellungen für verschiedene Assessments zu ermöglichen.

### 3.3.2 Beantworten der Aufgaben

Die Beantwortung der drei umgesetzten Aufgabentypen ist einheitlich in zwei verschiedene Phasen aufgeteilt. Zu Beginn einer jeden Aufgabe befindet sich das Programm im *Lesemodus*, kein Fokus ist gesetzt. Der Benutzer hat hier die Möglichkeit, die Frage und ihre Antwortmöglichkeiten durchzulesen.

Durch einmaliges Bestätigen (Kopfnicken) gelangt der Benutzer in den Antwortmodus, der Fokus liegt auf der ersten Antwortmöglichkeit und der Benutzer kann die Antworten nun sukzessive bestätigen bzw. ablehnen.

Diese Vorgehensweise hat sich in Versuchen als sinnvoll herausgestellt, da der Benutzer die Frage im Lesemodus in Ruhe durchgehen kann, ohne versehentlich Mimik-Signale zu senden. So neigen zum Beispiel viele Benutzer zu einem unwillkürlichen Nicken nach dem Verstehen der Frage und dem Überlegen der richtigen Antwort.

### **Single-Choice-Aufgaben**

Single-Choice-Aufgaben bestehen aus einer Frage und einer Menge von Antworten. Zur Lösung der Aufgabe muss die richtige Antwort ausgewählt werden.

Das Beantworten dieses Aufgabentyps ist auf eine sehr intuitive Art möglich: Die Frage wird dem Benutzer mitsamt den Antworten präsentiert, der Fokus – grafisch dargestellt durch farbiges Hinterlegen des Textes – liegt auf der ersten Antwort. Das Programm wartet nun auf eine Eingabe des Benutzers. Schüttelt er den Kopf, so geht der Fokus über auf die nächste Antwort und das Programm wartet auf die nächste Eingabe. Ist keine weitere Antwort mehr vorhanden, geht der Fokus wieder über zur ersten Antwort. Nickt der Benutzer hingegen, so wird die Antwort im Fokus als gewünschte Lösung selektiert und das Programm wechselt zur nächsten Aufgabe.

### **Multiple-Choice-Aufgaben**

Im Gegensatz zu einer Single-Choice-Aufgabe kann der Benutzer bei einer Multiple-Choice-Aufgabe eine beliebige Anzahl an Antworten bestätigen. Davon abgesehen sind die beiden Aufgabentypen identisch.

Dementsprechend gestaltet sich auch das Beantworten mit Hilfe der Mimikerkennung sehr ähnlich:

Einziger Unterschied ist hierbei, dass nach dem positiven Antworten auf eine Frage, also einem Nicken des Benutzers, nicht sofort zur nächsten Frage übergegangen wird. Stattdessen hat der Benutzer die Möglichkeit, neben der bereits selektierten noch weitere Antworten auszuwählen, bis er bei der letzten Antwort angekommen ist. Im Anschluß an deren Beantwortung wechselt das Assessment automatisch zur nächsten Aufgabe.

### **Pick-and-Place-Aufgaben**

Pick-and-Place-Aufgaben sind in Bezug auf die Anzahl der verschiedenen Aktionen der komplizierteste Aufgabentyp. Sie bestehen aus einem Satz oder kurzen Text mit einer bzw. mehreren Textlücken sowie einer Menge von Textstücken.

Da jede Antwort mit jeder vorhandenen Textlücke kombiniert werden kann, ist hier im Allgemeinen eine mehrmalige, schleifenähnliche Abarbeitung der Antworten nötig:

Im ersten Schritt wird die erste Antwort in die erste Textlücke gesetzt. Der Benutzer bestätigt diese Aktion durch Nicken oder veranlasst durch Kopfschütteln das zyklische Austauschen der eben ausgewählten mit der nächsten Antwort. Wurde eine Kombination bestätigt, schaltet das Assessment weiter zur nächsten Textlücke und zurück zur ersten noch nicht verbundenen Antwort. Nachdem für jede Textlücke ein solcher Antworten-Durchlauf gestartet und die

Observed Assessment

**Multiple Choice-Aufgaben**

Welche dieser Städte liegen in Bayern?

Salzburg

Passau

Ulm

Neu-Ulm

Observed Assessment

**Pick-and-Place-Aufgaben**

Wer reitet so spät durch Nacht und Wind? Es ist der Vater mit seinem Kind.

Er hat den Knaben wohl in der Kindertagesstätte abgegeben.

Er faßt ihn sicher, er hält ihn warm.

den Arm

den Händen

der Schuster

Abbildung 3.2: Beantwortung von Multiple-Choice- und Pick-and-Place-Aufgaben

letzte Textlücke mit einer geeigneten Antwort verbunden wurde, schaltet das Assessment auch hier weiter zur nächsten Aufgabe.

### 3.3.3 Beurteilung

Vom konzeptionellen Standpunkt gesehen, erwies sich die Steuerung durch Kopfgesten als durchaus praktikabler und intuitiver Steuerungsmodus des Assessment Centers. Der entscheidende Faktor für den reibungsfreien Ablauf ist jedoch ganz deutlich die Qualität des Mimiksignals. Eine weitere Verbesserung der Detektionsgüte und vor allem auch der benötigten Zeit trägt ganz wesentlich zu einer Verbesserung des Ablaufs bei. Gerade bei Pick-and-Place-Aufgaben und den dafür nötigen Schleifengängen kann die Dauer zur Beantwortung dadurch erheblich reduziert werden. Das Auftreten von schnellen Signalfolgen hingegen konnte durch eine Dämpfung der erkannten Signale hinreichend vermindert werden. Daraus resultierende Probleme für die Beantwortung von Assessments sind nur sehr selten zu erkennen.

Allerdings ist die Verwendung einer Maus oder Tastatur fast nicht zu umgehen, sollen dem Benutzer weitere Möglichkeiten eingeräumt werden. Nachträgliche Änderungen und andere zusätzliche Kommandos mit Hilfe der Mimikererkennung zu lösen, verlangt eine Vielzahl zusätzlicher Abfragen. Dies steht jedoch im Kontrast zum Ziel, eine reibungsfreie Bedienungsmöglichkeit zu schaffen. Eine andere Möglichkeit besteht im willkürlichen Festlegen einer bestimmten Mimik, mit deren Hilfe man aus dem normalen Antwortmodus in einen speziellen Kommandomodus gelangen kann. Diese Vorgehensweise ist jedoch für den Benutzer sehr unintuitiv und wurde daher nicht weiter in Betracht gezogen.

## Kapitel 4

# Softwaretechnische Konzeption

### 4.1 Kommunikation via XML-RPC

#### 4.1.1 Grundlagen von XML-RPC

*XML-RPC* ist eine Spezifikation für entfernte Funktionsaufrufe (*Remote-Procedure-Calls*). Über ein Netzwerk hinweg können somit Funktionen anderer Programme aufgerufen und Parameter übermittelt werden. Im Jahre 1998 von Dave Winer spezifiziert, existieren mittlerweile Implementierungen in den verschiedensten Programmiersprachen und für eine große Anzahl an Betriebssystemen. Eine XML-RPC Nachricht ist eine HTTP-POST Anfrage, deren Inhalt als XML-Dokument kodiert ist. Der Server führt die gewünschte Funktion mit den mitgelieferten Parametern aus und sendet den Rückgabewert, abermals in XML formatiert, zurück. Basierend auf den fest definierten, freien Internet-Standards HTTP und XML liegt der Fokus von XML-RPC auf dessen Schlichtheit, die dennoch die Übertragung komplexer strukturierter Datenstrukturen erlaubt.

Eine XML-Anfrage umfaßt neben dem Namen der aufzurufenden Methode die Übergabeparameter (sofern vorhanden). Als Datentypen stehen `boolean`, `integer`, `double` und `string` sowie ein Datumstyp zur Verfügung. Auch die Übermittlung binärer Daten in *Base64-Encoding* ist möglich. Zudem unterstützt XML-RPC *Arrays* und andere Datenstrukturen. Die ebenfalls XML-kodierte Antwort kann neben den Rückgabewerten auch spezielle Fehlermeldungen enthalten.

### 4.1.2 Integration von XML-RPC in das Assessment Center

Die im Rahmen dieser Arbeit verwendete Implementierung der XML-RPC-Spezifikation wurde von der *Apache Software Foundation* entwickelt. Die Java-Bibliothek steht unter der *Apache Software License* zur freien Verwendung zur Verfügung.

Zur Kommunikation mit dem Multimodeltracker wurde in das Assessment Center die Klasse `de.ifis.assessment.client.observed.XMLRPCServer` integriert, die als XML-RPC-Server und somit als Ansprechpartner für den in der XML-RPC-Bibliothek implementierten Webserver fungiert.

Zum Start dieses Webservers wird der in Abbildung 4.1 dargestellte Code ausgeführt. Die Daten für den Namen und Port des XML-RPC-Servers werden hierbei dynamisch aus der zentralen `observed.properties`-Konfigurationsdatei geladen. Die gleichen Daten müssen vom XML-RPC-Klienten des Multimodeltracker verwendet werden, um dessen Kommunikation mit dem Server zu ermöglichen.

Zusätzlich muß der XML-RPC-Server die innere Klasse `ServerCallback` implementiert `AsyncCallback` mit den beiden Methoden `public void handleError(Exception e, URL u, String s)` und `public void handleResult(Object o, URL u, String s)` implementieren. Für die Zwecke des Assessment Centers können diese Methoden jedoch leer bleiben.

Die restliche Implementierung der Klasse orientiert sich direkt am Einsatzzweck des Servers. Alle zum Aufruf via XML-RPC benötigten Methoden müssen mit den dazugehörigen Parametern implementiert werden. In unserem Fall sind dies die folgenden, vom Multimodeltracker vorgegebenen Signal-Methoden:

- `public void newPersonDetected()`
- `public void personWentAway()`
- `public void newMimicDetected(int newMimic)`
- `public void newEmotionDetected(int newEmotion)`

```
WebServer webservice = new WebServer(port);
webservice.addHandler(ObservedProperties.
    getString('XMLRPCServerName'), this);
webservice.start();
```

Abbildung 4.1: Einbinden der XML-RPC-Bibliothek

Sehr schnell zeigte sich bei Tests mit dem Multimodeltracker eine stark störende Eigenschaft:

Eine leicht schräge oder kreisende Kopfbewegung resultierte in einer schnellen Folge von Nicken- und Kopfschüttel-Signalen, die bei direkter Weitergabe an das Assessment Center zur sofortigen Beantwortung mehrerer Fragen führten. Die Ergänzung des XML-RPC-Server um eine Komponente, die schnelle Signaländerungen dämpfen kann, konnte dieses Problem erheblich mindern. Ihr Funktionsprinzip ist wie folgt:

Wurde ein Signal erkannt und verarbeitet, ignoriert der XML-RPC-Server für einen kurzen Zeitraum (Standard: 1 Sekunde) alle Folgesignale. Erst danach sind wieder Eingaben akzeptiert.

Implementiert wurde diese Komponente durch die Methode `protected boolean commandTime()`. Diese prüft, ob seit dem letzten per XML-RPC empfangenen Signal bereits mehr Zeit als eine Sekunde vergangen ist. Die Signal-Methoden fragen vor der Weitergabe des Aufrufs diese Information ab und verweigern andernfalls die Signalweitergabe.

Zusätzlich wurde die Interpretation des `PERSON_ENTERED`-Signals abgeändert. Dieses Signal selbst hat nun keine Programmreaktion mehr zur Folge. Die Programmreaktion (Ausblenden des "Benutzer abwesend"-Dialogs) wird stattdessen von der direkt auf das `PERSON_ENTERED`-Signal folgenden Mimik bewirkt.

Da die XML-RPC-Server-Klasse zu jedem Zeitpunkt in der Lage sein soll, Signale zu empfangen, wird sie in einem separaten Thread gestartet.

### 4.1.3 Merkmale der Umsetzung mit XML-RPC

Die Verwendung von entfernten Funktionsaufrufen ermöglicht die reibungsfreie Kommunikation des Assessment Centers mit der Mimikerkennungs-Anwendung. Darüber hinaus schafft die Implementierung jedoch eine Betriebssystems- und Programmiersprachen-unabhängige Schnittstelle zur Interaktion des Assessment Centers mit anderen Anwendungen. Zum Beispiel konnte dank dieser Interoperabilität zum Testen der neuen Steuerungsmöglichkeiten des Assessment Centers ein einfacher Java/Swing-Client genutzt werden, der über Schaltflächen per XML-RPC die genutzten Methoden aufruft.

Die Steuerung des Assessment Centers durch beliebige andere Eingabemöglichkeiten – beispielsweise Sprach- und Gestenerkennung oder spezialisierte Eingabegeräte – ist somit mit nur sehr geringem Implementierungsaufwand möglich.

Der Einsatz von XML-RPC im speziellen bietet die Vorteile der sehr einfachen und flexiblen Umsetzung sowie die Verfügbarkeit einer Vielzahl an unterschiedlichen Implementierungen.

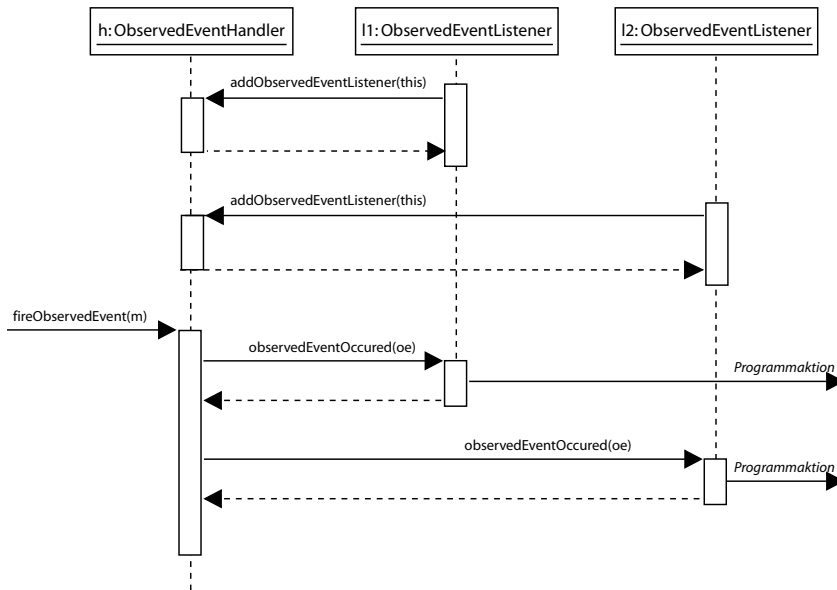


Abbildung 4.2: Umsetzung des Delegation-Model im Assessment Center

## 4.2 Integration in Swing-Oberfläche

### 4.2.1 Verwendete Konzepte

#### Singleton-Klassen

Das Grundprinzip von *Singleton-Klassen* ist die Existenz lediglich einer einzigen Objekt-Instanz, die bei allen Anforderungen verwendet wird.

Umgesetzt wird dies in erster Linie durch einen geschützten (`private`) Konstruktor, der das Erzeugen neuer Objekte von außen verhindert. Um anderen Klassen dennoch die Möglichkeit zu bieten, mit der Instanz zu arbeiten, verfügt eine Singleton-Klasse über die statische Methode `getInstance`.

Diese Methode prüft, ob bereits ein Objekt erzeugt und in der geschützten Klassenvariable vom Objekttyp gespeichert wurde. Ist dies nicht der Fall, erzeugt die Methode ein neues Objekt, weist es der Klassenvariable zu und gibt dieses neue Objekt zurück.

Auf diese Art und Weise finden alle Objektoperationen auf ein und der selben Instanz statt.

## Delegation-Model

Das Designpattern *Delegation* beschreibt die Arbeit mit *Events* und *Eventlistenern* zur Übermittlung empfangener Signale.

Im Mittelpunkt dieses Prinzips steht die Empfangsklasse, die über eine Schnittstelle bestimmte Signale erhält. Eventlistener können sich über die Methoden `addEventListener` und `removeEventListener` bei ihr an- und abmelden. Dabei sind *EventListener* beliebige Klassen, die ein bestimmtes Interface mit der darin vorgegebenen Funktion `handleEvent` implementieren.

Erhält die Empfangsklasse nun ein Signal in Form eines Aufrufs ihrer `fireEvent`-Methode, so ruft sie jeden der zu diesem Zeitpunkt registrierten Eventlistener auf und führt die `handleEvent`-Funktion aus. Als Parameter übergibt sie ein Eventobjekt mit Informationen über das empfangene Signal.

Somit delegiert sie das empfangene Signal und die damit verbundenen Aufgaben an ihre Eventlistener.

Dieses Konzept ist Grundlage der Übermittlung von Benutzersignalen in Java Swing. Zahlreiche Klassen wie z.B. `MouseEvent` / `MouseListener` und `MenuKeyEvent` / `MenuKeyListener` sind in der Swing-API bereits vordefiniert.

### 4.2.2 Klassen und Methoden

Zur Übermittlung eingehender Signale der Mimikererkennung, ruft der XML-RPC-Server – nach dem Test der Signaldämpfungs-Funktion – die zentrale `ObservedEventHandler`-Klasse auf. Diese übermittelt das zum Signal erzeugte `ObservedEvent` an alle bei ihr registrierten `ObservedEventListener` über die im Interface vorgegebene `handleObservedEvent`-Funktion. Dieser Vorgang wird in Abbildung 4.4 als UML-Programmfluss-Diagramm dargestellt.

Je nach erkannter Emotion werden verschiedene Abläufe angestoßen.

Verläßt der Benutzer das Blickfeld der Kamera wird ein `PERSON_LEFT`-Signal übermittelt und die zentrale Client-Klasse reagiert mit Öffnen eines Dialogfensters, das den Benutzer auffordert, wieder in das Blickfeld zurückzukehren. Wird im folgenden ein `PERSON_ENTERED`-Signal übermittelt, verschwindet der Dialog und das Assessment kann fortgesetzt werden.

Beim Anzeigen des Startbildschirms hingegen reagiert das Assessment Center auf das spezifizierte `startAssessment`-Signal durch Starten des Assessments.

Befindet sich der Benutzer im Fragemodus, werden erkannte Kopfgesten an die jeweilige Frage weitergereicht.

Kopfschütteln führt zum Weiterschalten und Markieren der nächsten Antwortmöglichkeit. Implementiert wird das in der jeweiligen `InteractiveQuestionGUI`-Klasse, die sich hierzu kurzzeitig als `ObservedEventListener` registriert. Als Reaktion auf eingehende `NODDING`-Events färbt und entfärbt sie die Hintergründe der entsprechenden Antwortmöglichkeiten bei Single- und Multiple-Choice-Fragen. Beim Beantworten von Pick-and-Place-Aufgaben hingegen wird

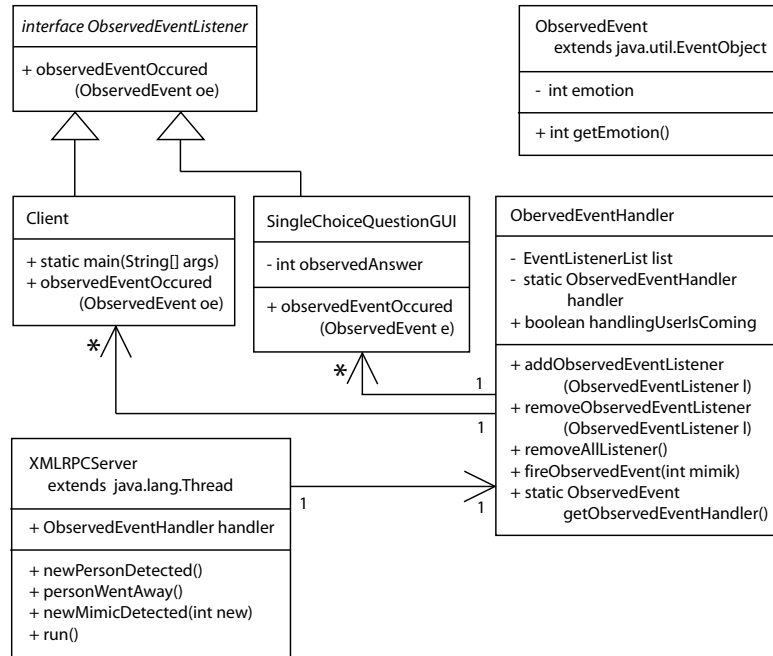


Abbildung 4.3: Klassendiagramm der Event-verarbeitenden Klassen

die aktuelle Lücke mit der aktuellen Frage *connected* bzw. wieder *disconnected*. Nicken hat das Akzeptieren der gerade angezeigten Antwortmöglichkeit zur Folge. Das `QuestionModel` speichert die gewählte Antwort, die `QuestionGUI`-Klasse meldet sich vom `ObservedEventHandler` ab und ruft die Methode `fireExerciseChange` auf, die zur nächsten Aufgabe wechselt.

## 4.3 Erweiterungen des Assessment Center

Für das Bearbeiten von Assessments mit Hilfe der Mimikererkennung und die Nutzung als Präsentationsprogramm im Dauereinsatz waren eine Reihe von Modifikationen des Assessment Centers nötig.

### 4.3.1 Konfigurationsmöglichkeit

Um die Mimikkomponente so flexibel wie möglich zu gestalten, ist die Konfiguration über ein eigenes `ResourceBundle` möglich. In der Textdatei `observed.properties` sind die Zuordnungen von Mimik zu Bedeutung, aufrufende Emotionen, Zeitkonstanten, XML-RPC-Daten und Textkomponenten für Mimikre-

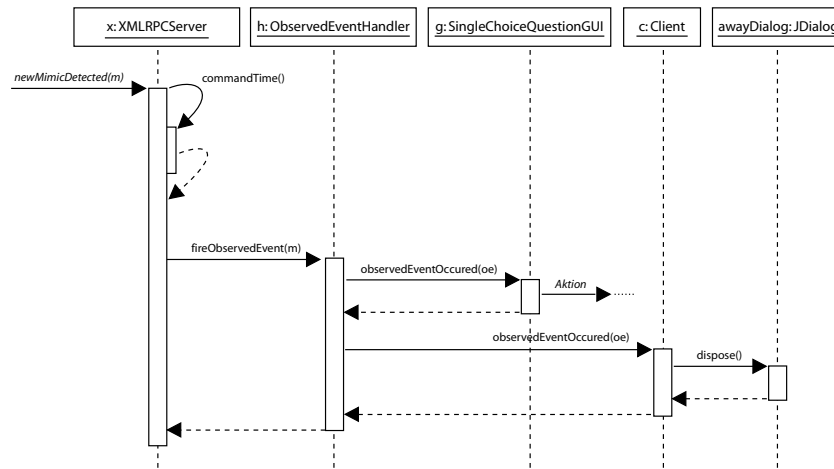


Abbildung 4.4: Programmfluss im Assessment Center bei Eingang eines Mimiksignals

levante Oberflächenteile abgelegt. Zugriff darauf bietet die Klasse `ObservedProperties` mit ihren statischen Methoden:

`getString(String s)` liest den Wert zum angegebenen Schlüssel aus und gibt ihn zurück

`getInt(String s)` liest den Wert zum angegebenen Schlüssel aus und wandelt ihn – sofern möglich – in einen Integerwert um

`getEmotionConst(String s)` gibt den Integer-Wert zum übergebenen Mimiknamen zurück

`getEmotionName(int e)` Gegenstück zu `getEmotionConst`

`getEmotionText(int e)` gibt den Text zur übergebenen Mimik zurück

`getActionConst(String s)` gibt den Integer-Wert zum übergebenen Aktionsnamen zurück

### 4.3.2 Mimikanzeige

Damit der Benutzer jederzeit Einblick in die gerade von der Mimikererkennung erkannte Emotion hat, wurde der Beantwortungsbildschirm um eine Mimikanzeige erweitert. Hierzu registriert sich die Klasse `ObservedExaminationPanel` als `ObservedEventListener` und aktualisiert bei jeder neu erkannten Emotion ein Textfeld. Konfiguriert werden kann dieses Textfeld in der `observed.properties`-Datei.

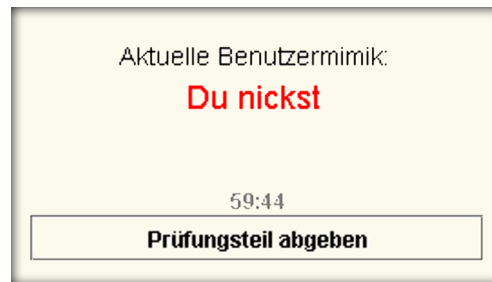


Abbildung 4.5: Anzeige der Benutzermimik

### 4.3.3 Demonstrations-Modus

Als Anforderungen eines Demonstrations-Modus, in dem das Assessment Center z.B. auf einer Messe ablaufen kann, haben sich die folgenden herausgestellt:

**Das Assessment Center soll ohne Eingriff eines Administrators mit Hilfe der Mimikererkennung gestartet und vollständig beantwortet werden können.**

Die Implementierung dieses Punktes verlangt die konsequente Verwendung der Mimikererkennung zur Steuerung aller Programmfunktionen. Während das Beantworten einer Frage wie bereits in Abschnitt 3.3 beschrieben von vorne herein auf die Steuerung mit Kopfgesten ausgelegt wurde, mußte der Prozess des Startens eines Assessments überarbeitet werden. Für diesen Zweck wurde in der Konfigurationsdatei `observed.properties` eine Konstante definiert, die einem Mimiksignal auf dem Startbildschirm die Aktion "Starten des Assessments" zuordnet.

Die GUI-Klasse `ObservedLoginPanel` meldet sich nach dem Start beziehungsweise Neustart des Assessment Centers bei der `ObservedEventHandler`-Klasse an. Auf das eingegangene Start-Signal kann somit mit dem Starten des Assessments reagiert werden. Im Anschluß an diesen Vorgang wird natürlich auch die Registrierung als `ObservedEventListener` zurückgezogen.

**Nach einer längeren Inaktivitätszeit soll ein automatischer Neustart des Assessment Center erfolgen.**

Dies wurde durch Verwenden eines Timers in der zentralen `Client`-Klasse realisiert, der bei Überschreiten einer in `observed.properties` definierten Zeit einen Neustart durchführt.

Der Timer wird – wie in Java üblich – mit einem Objekt der Klasse `javax.swing.Timer` realisiert. Dem Konstruktor wird dabei ein Zeitintervall (in unserem Fall 1000 Millisekunden) sowie ein `ActionListener` mit der Methode `actionPerformed(ActionEvent evt)` übergeben. Diese Methode wird vom Timer alle 1000 Millisekunden aufgerufen und überprüft, ob die in der Konfigurationsdatei definierte Zeit bereits abgelaufen ist. Ist der Test positiv, so wird das Assessment Center neu gestartet.

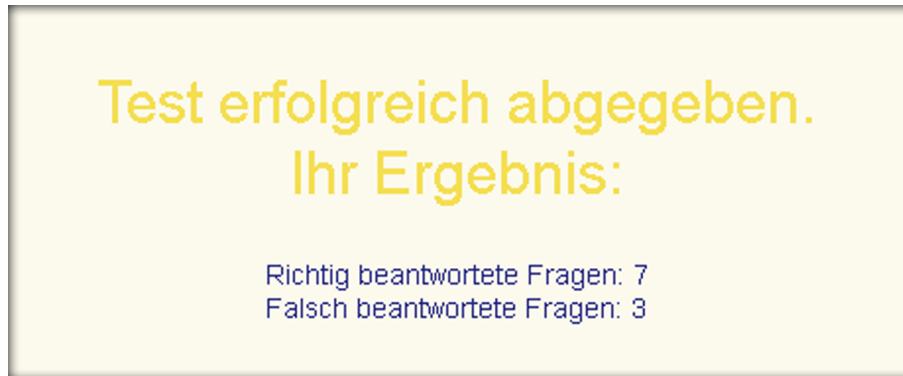


Abbildung 4.6: Der Auswertungsbildschirm

**Nach Fertigstellen eines Assessment soll ebenfalls ein Neustart erfolgen.**

Ein zweiter konfigurierbarer Timer, diesmal jedoch im `ObservedEndPanel` angesiedelt, erledigt diese Aufgabe nachdem der Auswertungsbildschirm für die in `observed.properties` definierte Zeit angezeigt wurde.

**Nach dem Neustart sollen alle zuvor gegebenen Antworten verworfen werden und das Assessment Center ohne Vorwissen neu starten.**

Für die Implementierung des Neustarts waren aufwendigere Veränderungen als für die übrigen Funktionen nötig. Da unter anderem die Klassen `SessionBean` und `AsAcServiceProvider` nach dem Singleton-Pattern implementiert sind, mußten diese um eine `restart`-Methode erweitert werden, die die gespeicherten Objekte durch neu erzeugte ersetzt. Gesteuert wird das gesamte Neustartverhalten von der Methode `restart` in der `Client`-Klasse, die von der statischen Methode `restartAssessment` aufgerufen wird.

#### 4.3.4 Auswertungsbildschirm

Als letzte Anforderung soll das Ergebnis eines abgeschlossenen Assessments sofort berechnet und auf einem Auswertungsbildschirm automatisch angezeigt werden.

Die Berechnung des Resultats in der Klasse `ObservedEndPanel` erfolgt mittels der in der `Session` gespeicherten `ExaminationSpecification`. Der für diesen Zweck initialisierter `ExaminationSpecificationHelper` erzeugt einen Iterator über die bearbeiteten `ExerciseRefs`, die den Zustand der Fragen und die erzielte Punktzahl enthalten. Durch das aufaddieren dieser Resultate entsteht das auf dem Auswertungsbildschirm präsentierte Endergebnis.

Nach Ablauf der `endPanelTime` (definiert in der Konfigurationsdatei) wird dieses Ergebnis im Rahmen der Neustart-Routine gelöscht und das Assessment startet von Neuem.

## Kapitel 5

# Beurteilung

Als zentrales Ergebnis des durchgeführten Projekts stellte sich heraus, dass eine zuverlässige Erkennung der Kopfgesten eine unbedingte Voraussetzung für den reibungslosen Ablauf eines Assessments ist. Verstärkt wird diese Notwendigkeit dadurch, dass jeder weitere Eingriff in den Programmablauf, z.B. zum nachträglichen Ändern einer fehlerhaften Antwort, wieder die Verwendung von Maus oder Tastatur nötig macht. Versuche, auch solche Befehle mit Kopfgesten zugänglich zu machen, scheiterten an dem großen zusätzlichen Aufwand, den der Benutzer durch die zahlreichen nötigen Zwischenfragen und Bestätigungen zu leisten hat.

Ein weiteres Problem liegt im relativ langen Detektionszeitraum, den die Mimikererkennung braucht, um Benutzereingaben sicher als solche zu erkennen und vor allem auch deren Ende zuverlässig festzustellen. Dieser Umstand hat zur Folge, dass zum Beispiel bei einer längeren Folge von Verneinungen zwischen den verschiedenen Kopfschüttel-Vorgängen relativ große Pausen nötig sind, um die Mimikererkennung wieder zurück in den neutralen Zustand zu bringen. Erst dann kann erneutes Kopfschütteln wieder als solches erkannt und ein entsprechendes Signal gesandt werden.

Versuche, über längere Zeiträume anhaltende Kopfgesten in mehrere verschiedene Signale aufzuteilen, haben sich leider als nicht-praktikabel erwiesen. Denn da für den Benutzer nicht exakt zu bestimmen ist, wann seine Mimik wieder als neutral erkannt wird, hat er in diesem Fall keine Kontrolle darüber, wieviele einzelne Eingaben vom System erkannt werden.

Als dritte Schwierigkeit erwies sich zunächst eine besondere Eigenheit der Mimikererkennung: Bestimmte Kopfbewegungen führten zu einer mehrmals wechselnden Interpretation der analysierten Bilddaten.

Die Ergänzung des XML-RPC-Server um die beschriebene Dämpfungskomponente konnte diese Problematik jedoch erfolgreich beseitigen.

Als letzte Eigenheit ist die Eigenschaft der Mimikererkennung zu nennen, einmal gefundene Gesichtszüge des Benutzers auch nach dessen Verschwinden noch im

Hintergrund zu suchen und dabei auch an nur sehr schwachen Konturen hängen-zubleiben. In vielen Fällen ist hier ein manuelles Reinitialisieren der Gesichtsdetektion nötig.

Trotz der beschriebenen Probleme sind die im Rahmen dieses Programmierpraktikums neu erarbeiteten Funktionalitäten des Assessment Centers jedoch keineswegs ohne Nutzen. Die Entwicklung zahlreicher neuer Funktionen wie die timergesteuerte Neustart-Routine und der Auswertungsbildschirm sind in beliebigen anderen Einsatzfällen nutzbar.

Von weit größerer Bedeutung ist allerdings die Neuorganisation des Beantwortungsablaufes der drei bearbeiteten Aufgabentypen. Im Gegensatz zur komplexen visuellen Steuerung der Programmoberfläche auf dem Bildschirm kann ein komplettes Assessment nun mit Hilfe einer einzigen binären Signalsequenz beantwortet werden. Die Integration des standardisierten XML-RPC-Protokolls zur Befehlseingabe erlaubt es, diesen Modus durch eine Vielfalt von Eingabemöglichkeiten zu steuern. So sind neben der Bedienung mittels einer Mimikerkennung auch die Interpretation von Gesten oder Tonsignalen zur Bearbeitung von Assessments möglich, und das unabhängig von deren Programmiersprache und benutzter Hardware. Zudem ist die Steuerung mittels anderer, sehr simpler Eingabeinterfaces denkbar, was neben vielen anderen Vorteilen beispielsweise auch körperbehinderten Menschen den Umgang mit dem Assessment Center erlauben könnte.

# Anhang A

## Dokumentation

### A.1 Bedienung

Die Durchführung eines Assessments über die verbundene Mimikererkennung gestaltet sich sehr intuitiv.

Nach dem Eintritt des Benutzers in das Blickfeld der Kamera registriert das Assessment Center dessen Anwesenheit und zeigt den Startbildschirm. Der Benutzer wird aufgefordert, das Assessment durch Nicken des Kopfes zu starten. Im folgenden präsentiert das Assessment Center dem Benutzer Aufgabe für Aufgabe in einer zufälligen Reihenfolge und wartet auf dessen Eingaben. Nach Beantwortung der letzten Aufgabe wird automatisch ein Auswertungsbildschirm mit den Prüfungsergebnissen angezeigt. Das Assessment ist somit beendet und startet nach einem kurzen Zeitraum von neuem.

Verläßt der Benutzer das Kamerablickfeld, wird ein Hinweisfenster eingeblendet und das Assessment Center wartet auf die Rückkehr des Benutzers.

Bei der Beantwortung von Aufgaben sind drei verschiedene Typen zu unterscheiden:

#### **Single-Choice-Aufgaben**

Single-Choice-Aufgaben, erkennbar an den runden Aufzählzeichen, erlauben die Auswahl genau einer der möglichen Antworten. Startposition für die Bearbeitung der Aufgaben ist der Lesemodus: keine der Antworten ist markiert. Durch Nicken des Benutzers wechselt die Aufgabe in den Antwortmodus und das Assessment Center markiert automatisch die erste Antwortmöglichkeit.

Der Benutzer hat nun zwei Möglichkeiten:

- Kopfnicken bestätigt die markierte Antwort und das Assessment Center geht zur nächsten Frage über.

- Kopfschütteln verneint die markierte Antwort und die Markierung geht auf die nächste Antwort über. Ist bereits die letzte Antwort erreicht, wechselt die Markierung abermals zur ersten Antwort. Dieser Vorgang wird bis zur Bestätigung einer Antwort durch den Benutzer wiederholt.

### **Multiple-Choice-Aufgaben**

Multiple-Choice-Aufgaben, erkennbar an den quadratischen Aufzählzeichen, erlauben im Gegensatz zu den Single-Choice-Aufgaben die Auswahl beliebig vieler Antworten. Auch das Verneinen sämtlicher Antworten ist möglich. Analog zum vorigen Aufgabentyp startet eine Multiple-Choice-Aufgabe im Lesemodus und wechselt durch Bestätigung des Benutzers in den Antwortmodus. Auch die Bestätigung einer markierten Antwort läuft analog zu den Single-Choice-Aufgaben, es existiert lediglich ein Unterschied: Bestätigt der Benutzer die gerade markierte Antwort, schaltet das Assessment Center nicht zur nächsten Aufgabe weiter, sondern erlaubt die zusätzliche Auswahl weiterer Antworten. Erst nach dem Bestätigen oder Verneinen der letzten Antwortmöglichkeit schaltet das Assessment Center zur nächsten Aufgabe über.

### **Pick-and-Place-Aufgaben**

Die Beantwortung von Pick-and-Place-Aufgaben gestaltet anders als die der beiden Auswahlaufgaben. Pick-and-Place-Aufgaben bestehen aus einem Lückentext und einer Menge mit Worten bzw. Satzteilen. Ziel der Aufgabe ist es, jede Lücke mit dem passenden Satzteil zu füllen.

Auch dieser Aufgabentyp startet im Lesemodus und wartet mit dem Aufruf des Antwortmodus auf eine Bestätigung des Benutzers. Im ersten Schritt wird nun in die erste Textlücke der erste Satzteil eingefügt und auf eine Reaktion des Benutzers gewartet.

- Bestätigt der Benutzer die angezeigte Kombination durch Kopfnicken, wird der Satzteil fest mit der Lücke verbunden. Im nächsten Schritt wird der erste noch nicht verbundene Satzteil in die nächste Textlücke gesetzt und auf eine abermalige Benutzerreaktion gewartet.
- Schüttelt der Benutzer hingegen mit dem Kopf, wird der Satzteil wieder aus der Textlücke entfernt und der nächste noch nicht verbundene Satzteil wird hineingesetzt. Nach dem Erreichen des letzten Satzteils wird zyklisch wieder zum ersten gewechselt.

Die Beantwortung einer Pick-and-Place-Aufgabe endet mit dem Verbinden der letzten vorhandenen Textlücke mit einem Satzteil. Danach wird automatisch zur nächsten Aufgabe übergewechselt.

## A.2 Einrichten des Assessment Centers

Die Konfiguration der Komponente für die Zusammenarbeit mit der Mimikerkennung erfolgt über die Datei `observed.properties`.

Im folgenden eine Auflistung der verschiedenen Parametergruppen:

**Emotions-Namen** Dieser Bereich enthält die Namen der vom Multimodeltracker unterstützten Emotionen sowie deren Zuordnung zu den int-Konstanten (z.B. `NEUTRAL`, `LAUGHING`, `PERSON_LEFT`).

**Emotions-Texte** Enthält einen kurzen Beschreibungstext zu jeder Emotion für die Mimikanzeige.

**Emotions-Mapping** Enthält die Zuordnungen von Emotionen zu bestimmten Aktionen des Assessment Centers (z.B. `startAssessment`).

**Zeitkonstanten** Hier können die beiden Zeitkonstanten der Anwendung frei definiert werden: `restartTime` beschreibt den Zeitraum der Inaktivität, bevor das Assessment Center von neuem gestartet wird, `endPanelTime` die Dauer, die der Auswertungsbildschirm nach Beenden eines Assessments angezeigt wird.

Alle Zeitangaben müssen in Millisekunden angegeben werden.

**XMLRPC** Beschreibt Port und Namen des integrierten XML-RPC-Server.

**Texte** Diese Gruppe enthält zahlreiche frei definierbare Texte für bestimmte Seiten des Assessment Centers, beispielsweise für die Mimikanzeige oder den Auswertungsbildschirm.

## A.3 Einrichten der Mimikerkennung

Die Konfiguration des Multimodeltracker geschieht per Kommandozeilenparameter.

Der empfohlene Aufruf lautet:

```
MultiModelTracker.exe --source livecam --detectYesNo --initLoop 10  
--delayFaceFound 5 --host <hostname> --port <portnummer> --xmlrpc
```

Hierbei bedeuten die einzelnen Parameter:

`source` Quelle der Bilddaten, `livecam` spezifiziert die angeschlossene Kamera

`detectYesNo` Aktiviert die Detektion der Kopfgesten

`initLoop` Zeitaufwand für das initiale Modell-Einpassen

`delayFaceFound` Frameanzahl für Berechnung der Position des Kopfmodells

**host** Name oder IP-Adresse des Rechners, auf dem das Assessment Center ausgeführt wird

**port** Vom XML-RPC-Server des Assessment Centers verwendeter Port

**xmlrpc** Aktiviert die XML-RPC-Unterstützung

Aus Performancegründen empfiehlt sich der Betrieb des Multimodeltracker auf einem eigenen PC. Für die Kommunikation mittels XML-RPC muss dieser Computer per Netzwerk mit dem PC des Assessment Center verbunden sein und der Multimodeltracker mit der **xmlrpc**-Option und der Adresse des Assessment Center-PCs gestartet werden. Insbesondere darf der gewählte Port nicht durch eine Firewall blockiert werden.

## Anhang B

# Bibliographie

[Ekman1978] Paul Ekman and Wallace V. Friesen and Joseph C. Hager: *Facial Action Coding System(FACS): Manual*, Palo Alto: Consulting Psychologists Press, 1978

[Pantic2000] Maja Pantic and Leon J.M. Rothkrantz: *Automatic Analysis of Facial Expressions: The State of the Art*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22, Page 1424-1445, 2002

[Wimmer2005] Matthias Wimmer: *Autonomous Emotion Detection in Real Life/Real World Condditions/Scenes*, München, 2005