Universität Leipzig
Fakultät für Mathematik und Informatik
(Institut für Informatik)

Spezifikation und Implementierung einer Übungssoftware deutsch-arabisch

Diplomarbeit

Leipzig, Juni 1998

vorgelegt von
Pauli, Dirk

Vorwort

Die Rechenleistung moderner Personalcomputer ermöglicht inzwischen vielfältige Anwendungsmöglichkeiten jenseits der naturwissenschaftlich-technischen Bereiche. Gerade bei der Unterstützung und Erweiterung des Fremdsprachenunterrichts stellt der Computer ein mannigfaltig einsetzbares Werkzeug dar. Mit der vorliegenden Arbeit soll ein spezieller Einsatzbereich untersucht werden, der bisher ein Schattendasein fristet: Der Unterstützung der Übungsmöglichkeiten im Arabischunterricht.

An dieser Stelle möchte ich Herrn Prof. Dr. Heyer für die fortwährende Unterstützung und Betreuung während der Entwicklung dieses Projekts danken. Weiterer Dank geht an Herrn Prof. Dr. Schulz für die Einführung in die Probleme des computerunterstützten Arabischunterrichts und den Entwicklungsauftrag.
Danke auch an Herrn Rothe vom Universitätsrechenzentrum, der mir mit der Demonstration seines Lernprogramms "Deutsch" Inspirationen für die Entwicklung des eigenen Systems mit auf den Weg gab und an Matthias Kluge, der mir unermüdlich mit der Zuarbeit der Übungseinheiten half.
Schließlich möchte ich meiner gesamten Familie danken, ohne deren Unterstützung diese Arbeit wohl nicht zustande gekommen wäre.

Inhaltsverzeichnis

1. Pädagogische Betrachtungen

1.1. Der Lernprozeß

1.1.1. Einführung

In der Literatur sind zum Thema Modelle des Lernprozesses die verschiedensten Varianten zu finden. Die einzelnen Modelle betrachten entweder den allgemeinen Lernprozeß, der auch das Lernen von z.B. Verhaltensweisen mit einschließt, oder beziehen sich stark auf das Lernen von fakten- und regeldominierten Wissensbereichen wie z.B. den naturwissenschaftlichen Fächern. Die Modelle der zweiten Gruppe lassen sich jedoch auf das Lernen von Fremdsprachen adaptieren. Hierbei fallen wiederum die verschiedenen Ausformungen auf, welche auf unterschiedlichen lernpsychologischen Theorien basieren. (Vgl. [Uth 78], [Sch 92], [Sei 93])

Im folgenden soll ein Modell beschrieben werden, daß sich ausschließlich dem Fremdsprachenlernprozeß widmet. Dabei werden die Auswirkungen auf den Unterrichtsaufbau im allgemeinen mit beschrieben, die die im Modell verfolgten Ziele unterstützen.

1.1.2. Ein Fremdsprachenlernmodell

1.1.2.1. Allgemeines

Das hier beschriebene Modell betrachtet den Fremdsprachenlernprozeß als eine Abfolge von 4 Ebenen, die jedoch nicht linear aufeinander folgen müssen und teilweise sogar iteriert werden sollen. Bei den 4 Ebenen handelt es sich um das Erfassen, Einprägen, Einüben und Anwenden. Dabei wird beachtet, daß einzelne Phasen einander bedingen und aus bestimmten Blickwinkeln Überlappungen möglich sind. So sind zum Beispiel in jeder Handlung das Entstehen neuer Abbilder, die Festigung und Präzisierung vorhandenen Wissens und die Neuverknüpfung der Gedächtnisinhalte zu beobachten. Die Phasen unterscheiden sich demnach nur in ihrer Schwerpunktsetzung innerhalb des Lernprozesses.

Handlungen, die schwerpunktmäßig der Erzeugung neuer Abbildungen von sprachlichen Erscheinungen im Gedächtnis des Lernenden dienen, werden im folgenden Erfassungshandlungen genannt. Als Nebeneffekte dieser Handlungen sind die Festigung vorhandener Abbilder und die Entwicklung von allgemeinen kognitiven Fähigkeiten zu nennen. Demgegenüber dienen

die Einprägehandlungen vorrangig der Festigung der sprachlichen Abbilder. Dabei werden aber auch neue Bedeutungen bereits bekannter Erscheinungen erschlossen. Die dritte Gruppe, die sogenannten Einübungshandlungen, hat ihren Schwerpunkt in der Ausbildung von sprachlichen Operationen, also der Automatisierung von Teilhandlungen, die zur fremdsprachigen Kommunikation nötig sind. Dadurch werden aber auch Ziele der Einprägungshandlungen mit bedient. Die letzte Gruppe von Handlungen, die Anwendungshandlungen, dient der Bewältigung von Kommunikationssituationen. Dabei können die Schwerpunkte der ersten drei Handlungsgruppen, wie z.B. das Erfassen neuer Erscheinungen, am Rande mit bedient werden.
Wie aus dieser Einteilung ersichtlich wird, sind diese Gruppen weder unabhängig voneinander noch lassen sie sich in strikt linearer Reihenfolge erfolgreich zum Lernen von Fremdsprachen einsetzen. Erst der Verbund dieser Handlungen ermöglicht ein effektives Aneignen von Fremdsprachen.

1.1.2.2. Erfassungshandlungen

Bei den Erfassungshandlungen steht die Vermittlung von neuen Wörtern, grammatischen Strukturen etc. im Vordergrund. Dabei muß der Lernende als erstes die Schriftgestalt abbilden, diese mit der vermittelten Lautfolge verknüpfen (Lesen und Hören) und selbst diese beiden Gestalten aktiv gebrauchen lernen (Schreiben und Sprechen). Damit ist die syntaktische Seite der Neuvermittlung abgeschlossen. Weitaus wichtiger ist jedoch nun die Zuordnung der fremdsprachigen Gestalt zu der muttersprachigen Bedeutung (semantische Aspekte). Dabei ist darauf zu achten, daß diese Verknüpfung nicht in Form von sogenannten Vokabelgleichungen erfolgt, da in den wenigsten Fällen eine wirkliche Äquivalenz zwischen dem fremdsprachigen Abbild und der muttersprachigen Bedeutung in allen Bedeutungsvarianten zu finden ist. Eine Eingliederung der Bedeutung in bestimmte Situationen ist somit sinnvoll und vermeidet falsche Übertragungen der Bedeutung von der Muttersprache auf die Fremdsprache. In dieser Phase ist es weiterhin sinnvoll, den Lernenden auf bestimmte sprachtheoretische Regelmäßigkeiten hinzuweisen. Dadurch wird die Einordnung der neuen Vokabeln in Wortfamilien und die Ableitung von verwandten Begriffen erleichtert.
Die allgemein akzeptierte Auffassung ist, daß es bezüglich der Behaltensleistung vorteilhaft ist, wenn bei der Vorstellung der neuen

Wörter von mehreren Rezeptionskanälen (optisch, akustisch) Gebrauch gemacht wird.
Besonders effektives Erfassen neuer Begriffe wird ermöglicht, wenn die Erfassungshandlungen in Aufgaben eingebettet werden, da dadurch eine höhere geistige Aktivität entsteht als bei der bloßen Vorstellung der neuen Begriffe. Dies setzt jedoch eine gewisse Motivierung des Lernenden vorraus, die ihn aktiv nach neuem anzueignendem Wissen suchen läßt.

1.1.2.3. Einprägehandlungen

Die Einprägehandlungen dienen dazu, über wiederholte Aufnahme und Wiedergabe von sprachlichen Äußerungen diese in ihrer Verwendung zu festigen. Einprägehandlungen werden immer kurz nach dem Erfassen neuer Begriffe, bei der Systematisierung dieser unter verschiedenen Aspekten und bei der Wiederholung von bereits bekanntem Sprachmaterial angewandt. Dies kann durch gezieltes Hin- und Herübersetzen, durch Füllen von Lückentexten oder als Reaktion auf z.B. bildliche Impulse erfolgen. Bei dem Einprägen von Sprachmaterial ist darauf zu achten, daß behaltensfördernde Zusammenhänge zwischen den Wörtern hergestellt werden. Dies kann über semantische Beziehungen wie Gleichheit oder Hierarchiebeziehungen erfolgen. Dadurch wird eine Systematisierung der Begriffe herbeigeführt, die das Behalten der Wörter erleichtert. Als einzuprägende Begriffe sollten vorrangig einzelne Wörter ausgewählt werden, da dies dazu führt, daß die Kombinationsfähigkeit der Lernenden gefördert wird. Dadurch sind die Lernenden mit einem geringen Wortschatz trotzdem in der Lage, vielfältige Aussagen zu generieren. Das Einprägen von komplexeren Strukturen macht nur bei feststehenden Floskeln Sinn. Auch mag es sinnvoll sein, im Anfangsunterricht zu Beginn kleinere Strukturen einprägen zu lassen, diese aber in dem Moment, wo alle beteiligten Wörter und grammatischen Erscheinungen, die die Phrase beinhaltet, zugunsten des Einprägens der Einzelwörter und der Kombinationsregeln zu vernachlässigen, da sie sonst nur den Artikulationsspielraum des Lernenden unnötig einengen und sein Gedächtnis belasten.
Um die Quote der richtig behaltenen Wörter zu optimieren, ist es von Nutzen, die erste Wiederholung möglichst kurz nach dem Erfassen anzusiedeln (da in der ersten Zeit nach dem Erfassen die Vergessensrate besonders hoch ist) und die Wiederholungen über den gesamten Unterrichtszeitraum zu verteilen. Außerdem ist bei einem kurzen Zeitraum

zwischen Erfassen und Wiederholung die Rate der vergessenen Wörter noch gering und bei einem größeren Abstand zwischen beiden müssen viele Wörter schon wie neu gelernt werden, was die Effizienz der Wiederholung senkt. Weiterhin ist es vorteilhaft, die Art der Wiederholung ständig zu variieren, so daß sich zum Beispiel mündliche und schriftliche Aufgaben auf produktivem und rezeptiven Gebiet abwechseln. Durch diese Variation werden die unterschiedlichen Verarbeitungszentren im Gehirn aktiviert und die Behaltensleistung gefördert.

1.1.2.4. Einübungshandlungen

Die Einübungshandlungen haben zum Ziel, die für die Kommunikation notwendigen Operationen und Teilhandlungen soweit zu automatisieren, daß sich der Lernende bei der Anwendung seines Wissens auf inhaltliche Aspekte konzentrieren kann und nicht alle dafür notwendigen Operationen Schritt für Schritt durchgehen muß.
Operative Ziele sind somit eine Korrektur von Fertigkeiten, die aus der Muttersprache her bekannt sind, um sie für die Fremdsprache anwendbar zu machen und die Ausbildung von Fertigkeiten, die in dieser Art in der Muttersprache inexistent sind. Solche zu korrigierenden oder neuzubildenden Fertigkeiten können sowohl sichtbarer (graphomotorisch, artikulatorisch etc.) als auch interner Natur (grammatische Fertigkeiten etc.) sein. Die Ausbildung derartiger Fertigkeiten ist oftmals mit einem komplexen Umbau der Handlungen verbunden, bevor sie ihr Optimum erreichen (Vgl. [Uth 78]).
Um diese Ziele zu erreichen, ist eine kontinuierliche Übung mit steigendem Schwierigkeitsgrad unabdingbar, der die Aufmerksamkeit des Lernenden von den Teilkomponenten einer Handlung hin zu der Komplexhandlung lenkt. Dabei ist es wichtig, daß besonders zu Beginn der Übungen darauf geachtet wird, daß die Handlungen fehlerfrei durchgeführt werden. Durch ein solches bewußtes Training werden dauerhafte Fertigkeiten ausgebildet und widrige Einflüsse von Fertigkeiten der Muttersprache nachhaltig zurückgedrängt.
Wie bei den Einprägehandlungen ist es auch hier vorteilhaft, wenn sich verschiedene Übungsarten in ständigem Wechsel ablösen, da sich so über gegenseitige Beeinflussungen der Teilbereiche Hören, Lesen, Sprechen und Schreiben positive Effekte einstellen und die Gesamtaktivität erhöht werden kann.

1.1.2.5. Anwendungshandlungen

Anwendungshandlungen sind die komplexeste Art der Lernhandlungen und können erst nach Erwerb eines gewissen Stammes an Wissen durchgeführt werden. Sie sollen den Lernenden auf kommunikative Situationen vorbereiten und die hierfür benötigten Fähigkeiten und Fertigkeiten ausbilden.

In dieser Phase werden vorhandene Kenntnisse, Fähigkeiten und Fertigkeiten zu den jeweiligen Bedingungen der aktuellen Situation immer neu verknüpft, so daß dadurch ein Netz von Operationen entsteht, das für spätere kommunikative Aufgaben bereitsteht.

Um diese Vernetzung zu optimieren, ist es wichtig, den Lernenden möglichst lebensnahe Aufgaben in definierten Situationen zu stellen. Somit sind originale Texte oder Radio- und Videoaufzeichnungen denen vorzuziehen, die als reine Lehrbuchartikel vorliegen, soweit das die Kenntnisse der Lernenden zulassen. Natürlich sorgt auch hier wieder die Abwechslung von akustischen mit optischen Materialien und der Wechsel von rezeptiven zu produktiven Aufgaben für eine förderliche Wirkung auf den Lernprozeß. Bei der Auswahl der Themen für die Anwendungshandlungen ist es wichtig, daß den Lernenden die Sachverhalte bekannt sind, da sonst auf Grund fehlender Sachkenntnisse keine Kommunikation zustande kommen kann bzw. das Verstehen behindert wird.

Wegen der relativ hohen Voraussetzungen für die Anwendungshandlungen werden sie im Anfangsunterricht nur wenig einsetzbar sein.

1.1.3. Alternativ-Modell

In [Sei 93] wird ein Modell vorgestellt, das sich über sogenannte Lernzielstufen definiert:

Lernzielstufe	*Methode der Leistungsmessung*
Wissen	Freies Erinnern
Verstehen	Zusammenfassen
Anwenden	Logisches Schließen
Analysieren	(Neu-) Strukturieren
Synthese	Neuordnen
Beurteilen	Urteil

(entnommen aus [Sei 93], S.152)

Die dort angegebenen 6 Stufen lassen sich jedoch mit den 4 Phasen des oben beschriebenen Modells von [Uth 78] in Korrelation bringen. So entsprechen die Lernzielstufen Wissen und Verstehen der Phase des Erfassens, die Lernzielstufen Anwenden und Analysieren entsprechen der Phase Einprägen und die Stufen Synthese und Beurteilen lassen sich mit der Phase des Anwendens vergleichen. Bei diesem Modell liegt der Schwerpunkt auf dem Erlernen von Faktenwissen, insofern wird zwischen Fakten und Fähigkeiten beim Lernprozeß nicht explizit unterschieden. Die Stufen Synthese ("...Informationen zu funktionellen Einheiten zusammenfassen...", [Sei 93], S.152, also eigene Aussagen mit dem bekannten Wortschatz produzieren) und Beurteilen gehen über das Ziel der Phase Anwenden im obigen Modell hinaus. Da jedoch der Schwerpunkt dieser Arbeit auf dem Anfangsunterricht liegt, ist gerade das Beurteilen ein zu weit gehendes Ziel und somit das erste Modell ausreichend.

1.2. Computerunterstützter Unterricht

1.2.1. Vorteile und Grenzen von CBT

1.2.1.1. Motivation für den Computereinsatz im Unterricht

Der konventionelle Unterricht an den Schulen und Hochschulen ist meist als Frontalunterricht in Gruppen ausgelegt. Diese Unterrichtsgestaltung ist aufgrund der relativ großen Schülerzahlen pro Gruppe wenig individuell. Dadurch werden einerseits die begabteren Schüler oft unterfordert, andererseits die schwächeren Schüler mit ihren Problemen allein gelassen. Diese Situation wird noch verstärkt durch den Zwang, einen bestimmten Lehrplan einhalten zu müssen. Der konventionelle Unterricht ist also aus Sicht der Lernenden mehrheitlich fremdgesteuert.

Seit Beginn der 80er Jahre sind bei der Unterrichtspräsentation Veränderungen zu beobachten. Der Trend geht weg von dem einfachen Tafel-Kreide-Unterricht hin zu der verstärkten Nutzung von Medien wie (Stand-)Bildern, Ton- und Videoaufzeichnungen und der Nutzung von Projektionsfolien. Natürlich ist auch mit einfachsten Mitteln ein abwechslungsreicher Unterricht möglich, durch den verstärkten Medien-Einsatz ist aber eine größere Abwechslung bei der Vermittlung von Wissen einfacher zu erreichen, was zu einer stärkeren Motivation bei den Lernenden und somit zu besseren Lernergebnissen führen sollte (Vgl. [Ehn

86]). Der Hauptnachteil des Unterrichts, die mangelnde Individualisierung läßt sich nur durch eine starke Verkleinerung der Gruppen oder die Nutzung neuer Möglichkeiten bei der Unterrichtsgestaltung vermeiden.
Eine solche Möglichkeit ist der Einsatz von Computern mit speziellen Lernprogrammen. Dabei soll aber der Computer nicht, wie von den Pionieren des CBT in den 60er Jahren gefordert, der Computer nach und nach den Lehrer vollständig ersetzen. Dieser Versuch wurde bei mehreren Untersuchungen als unwirksam bis schädlich für die Lernergebnisse befunden. Der Hauptgrund dafür dürfte die Tatsache sein, daß bei einem alleinigen Einsatz von CBT die sozialen Aspekte des Lernens stark vernachlässigt wurden und der Unterricht wieder in eine neue Einseitigkeit geführt wurde.
Eine Beiordnung von computerunterstützten Methoden zu den bisher genutzten Mitteln verbreitert jedoch die Angebotspalette und kann insbesondere motivational positiv auf den Unterricht wirken.

1.2.1.2. Vorteile

Bei richtiger Anwendung dieses Mediums läßt sich der Grad der Individualisierung des Unterrichts erhöhen. Gerade beim Einsatz von Computern in der Übung und Festigung gibt ein Programm objektivere Beurteilungen der Leistungen und die Abwesenheit des Gruppendrucks fördert die Leistungen der Schwächeren, die oftmals mißerfolgsorientiert sind, besser. Dabei ist jedoch darauf zu achten, daß die Ergebnisse einer solchen Übungssitzung als vertrauliche Daten behandelt werden und nur der Lernende selbst zu ihnen Zugang hat, da sonst leicht eine Aversion gegen das Medium entstehen könnte.
Liegt weiterhin die Steuerung des Programms in den Händen des Lernenden, ist es möglich, daß er gezielt die Lerninhalte abruft, die ihn interessieren oder bei denen er glaubt, daß er dort Übungsbedarf hat. Bei dieser Möglichkeit ist zu beobachten, daß es meist die Leistungsstärkeren sind, die aus einer solchen freien Steuerbarkeit Vorteile ziehen, während die schwächeren Lernenden oft besser mit einem vorgegebenen Kurs durch die Übungen zurechtkommen. (Vgl. [Sei 93], S.68)
Ein weiterer, nicht zu unterschätzender Vorteil von CBT ist die Möglichkeit der Visualisierung von komplexen Vorgängen über Simulationen. Gerade in der Berufsausbildung und in den naturwissenschaftlichen Fächern lassen sich so komplexe Vorgänge eingängig darstellen.

Durch die Entlastung des Lehrers von Routinearbeiten wie Übungseinheiten steigt die Qualität des Gesamtunterrichts, da dieser dann eingehender auf neue Inhalte im Unterricht eingehen kann. Dies impliziert natürlich, daß die Übungen via CBT außerhalb der regulären Unterrichtszeit stattzufinden haben. Dadurch entstehen für die Lernenden neue Freiheiten in bezug auf die Verwaltung ihrer Zeit.
Obwohl die Erstellung einer professionell geplanten CBT-Lektion oder Übungseinheit kostspielig ist, sind bei intensiver Nutzung dieser Möglichkeit Kostenspareffekte zu beobachten, die insbesondere darauf beruhen, daß die Kurszeiten gekürzt und zum Teil (z.B. bei beruflicher Weiterbildung) die Kosten für auswärtige Unterbringung reduziert werden können.

1.2.1.3. Nachteile

Beim momentanen Stand der Technik ist CBT aber auch mit einigen Nachteilen behaftet. Die Hauptargumente der CBT-Gegner sind der hohe Entwicklungsaufwand von CBT-Programmen und die fehlende bzw. mangelhafte Adaption an die Vorkenntnisse des Nutzers.
Der hohe Aufwand für die Erstellung eines CBT-Programms ist oft dafür ausschlaggebend, daß gerade bei kleineren Anwendergruppen das Preis-/ Leistungsverhältnis zu Ungunsten des CBT ausfällt. Sinnvoll wird der Einsatz also erst bei einer hinreichend großen Nutzerzahl, die am ehesten bei allgemeinen Themen wie z.B. Fremdsprachenunterricht erreicht wird oder bei einem sich über einen größeren Zeitraum nicht oder wenig änderndem Lernstoff bei einer fest umrissenen Nutzergruppe, wie es bei Schullernprogrammen oft der Fall ist.
Die mangelnde Adaptabilität der Programme beeinträchtigt die Qualität insofern, als daß auf den spezifischen Lernhintergrund eines einzelnen Lernenden nicht eingegangen werden kann. Als positiv sind schon solche Programme zu beachten, die sich auf eine relativ kleine, fest umrissene und in sich homogene Nutzergruppe abzielen, da die Eigenschaften der Lernenden dann untereinander sehr ähnlich sind.
Eine Anzahl der ich auf dem Markt befindlichen Programme sind noch strikt linear organisiert und beinhalten eine sehr geringe aktive Anpassungsfähigkeit an die Fähigkeiten der Nutzer. Das wird vor allem auf leistungsstärkere Lernende negativ wirken, da sie sich subjektiv betrachtet zu lange mit Übungen zu bereits bekannten und gefestigten Themen

beschäftigen müssen.
Eine andere Art von Programmen, die vornehmlich zur Wissensvermittlung eingesetzt werden, bauen auf der Hypertext-Technik auf. Dadurch wird der strikt lineare Charakter aufgehoben und die Nutzer können über Querverweise zu verwandten Themenbereichen gelangen. Eine Gefahr hierbei ist jedoch das Problem, daß nach relativ kurzer Zeit die Orientierung verloren gehen kann, wo im System man sich befindet ("lost in hyperspace"). Diese Gefahr läßt sich durch eine Markierung des beschrittenen Lernpfades und der Möglichkeit, diesen wieder zurückzuverfolgen, vermindern.
Eine oftmals angebrachte Kritik am CBT ist auch die eingeschränkte Interaktionsfähigkeit zwischen den Lernenden und dem Computer. In manchen Bereichen mag z.B. der Zwang, Texteingaben über die Tastatur zu machen, sogar hinderlich für den Lernerfolg sein (z.B. Schreiblernprogramme; Vgl. [Sei 93], S.119). Auch die Nutzung von Maus und Joystick kann bei manchen Problemstellungen künstlich anmuten. Ein anderes großes Problem stellt das Unvermögen der Programme dar, nahezu richtige Lösungen zu erkennen und diese als solche zu kennzeichnen. Um diesen Effekt hervorzurufen, müßten alle möglichen Antworten des Lernenden in einer bestimmten Situation vorher, d.h. während der Entwicklung der CBT-Einheit, antizipiert werden, damit man entsprechend auf sie reagieren kann. Dies läßt sich aber nur teilweise verwirklichen, z.B. in Programmen für den Fremdsprachenunterricht bei häufig auftretenden Verwechslungen oder bei einfachen Rechtschreibfehlern (1 Buchstabe vergessen etc.).
Schließlich ist es mit CBT allein nicht möglich, ein entsprechend hohes Ausbildungsniveau zu erreichen. Über eine Einbindung in die traditionellen Lehrmöglichkeiten lassen sich jedoch Vorteile erzielen.

1.2.2. Schlußfolgerungen für das Projekt

1.2.2.1. Zielsetzung des Projekts

Das zu erstellende Programm hat das Ziel, den Unterricht der arabischen Sprache zu ergänzen. Es soll eine weitere Möglichkeit für Studenten der Arabistik schaffen, das Gelernte in mehreren Übungen zu festigen und sich selbst möglichst effizient zu kontrollieren.
Als Voraussetzung zur Nutzung des Programms werden nur grundlegende Kenntnisse der Computerbedienung sowie die Kenntnis der arabischen Schriftzeichen gefordert. Die typische Nutzergruppe werden somit Studenten der Arabistik in den ersten Semestern sein.
Das Programm stellt nur eine Möglichkeit dar, im Unterricht behandelte Themen zu üben und zu festigen. Es sollen keine neuen Lerninhalte vermittelt werden. Da im allgemeinen davon ausgegangen werden kann, daß zu Beginn des Arabischunterrichts keinerlei Vorkenntnisse vorhanden sind (sog. Nullsprache), ist eine ständige Anwendung und Wiederholung des Lernstoffes besonders wichtig.
Die möglicherweise anzubringende Kritik, daß mit dem Programm nur die Schriftsprache geübt und gefestigt werden kann, ist nur bedingt berechtigt. Die mehrheitliche Meinung geht dahin, daß der Lernprozeß von der im gesamten arabischen Raum einheitlichen ("Bis auf bestimmte literarische Genres (Dialogpartien in Erzählungen, Theater, Volkspoesie, Witz) sind die Dialekte Verschriftungsversuchen gegenüber bisher resistent.", [Bau 95], S.343) Schriftsprache zu den regional stark unterschiedlichen mündlichen Dialekten sich einfacher gestaltet als der umgekehrte Weg. Daher wird in den meisten Bildungseinrichtungen der Schwerpunkt auf die Schriftsprache gelegt. (Vgl. [Bau 95], S 343f.)

1.2.2.2. Spezifische Probleme des Arabischunterrichts

Die arabische Sprache gehört nicht wie die meisten europäischen Sprachen zu den indogermanischen. Durch ihre Abstammung von den semitischen Sprachen kann sich der Lernende nicht auf die aus seiner Muttersprache bekannten Strukturen stützen. Auch die Anzahl der Lehnübersetzungen und Internationalismen ist im Arabischen vergleichsweise gering, so daß sich der Lernende auch hierauf nicht stützen kann. Weiterhin ist das Schriftbild des Arabischen von dem europäischer Sprachen vollständig verschieden, so daß zu Beginn erst das neue Schriftsystem erlernt werden

muß. Erst, wenn die arabische Schrift beim Lesen und Schreiben sicher beherrscht wird, ist ein Einsatz dieses Programms sinnvoll. Vorher könnte es sogar nachteilige Wirkung auf das Erlernen der Schriftsprache haben, da bei der Tastatureingabe weder die unterschiedlichen Grapheme für die einzelnen Phoneme beachtet werden müssen (diese Aufgabe erledigt der Computer selbst) noch die für die Schreibfertigkeiten notwendigen graphomotorischen Fertigkeiten geschult werden können.
Ist diese Fertigkeit aber erst einmal ausgebildet, kann das vereinfachte Schreiben über Tastatur sogar hilfreich wirken, da im Arabischen die meisten Wörter auf ihre (meist 3) sogenannten Wurzelkonsonanten (Radikale), die eine Grundbedeutung definieren, zurückführen lassen und die Feinunterscheidung der Wörter einer solchen Familie über Prä-, In- oder Suffixe bzw. unterschiedliche Vokalisierung und Konsonantenverdopplung hergestellt werden. (Vgl. [Bau 95], S 343 f.)

1.2.2.3. Übungstypen

Zur Festigung der Vokabeln und Wortstrukturen werden mehrere verschiedene Typen von Übungen angeboten. Diese sollen auf unterschiedliche Weise die Vokabeln festigen und vor eventuellen Verwechslungen warnen.

1.2.2.3.1. Multiple-Choice-Aufgaben

Bei der Gegenüberstellung von mehreren Antworten zu einer Fragestellung können leicht eventuell entstandene Verwechslungen zwischen verschiedenen, dem ungeübten Hörer ähnlich klingenden Worten (die sich auch im Schriftbild ähneln) getestet und gegebenenfalls verbessert werden.

1.2.2.3.2. Vokabelkontrollen

Die direkte Kontrolle von Vokabeln stellt das klassische Mittel der Übung von Wörtern und die Förderung deren Einprägung dar. Die Kontrolle kann sowohl in Richtung deutsch-arabisch als auch in Richtung arabisch-deutsch erfolgen. Das führt mittelfristig zu einer verbesserten Fähigkeit, arabische Wörter entschlüsseln zu können und sie mit ihren deutschen Äquivalenten zu verbinden. Dabei muß aber darauf geachtet werden, daß durch zu exzessive Nutzung dieses Übungstypus bei dem Lernenden nicht der Eindruck von "Vokabelgleichungen" entsteht. Auf eventuelle Mehrdeutigkeiten in der Übersetzung muß daher hingewiesen werden.

1.2.2.3.3. Vokalisierungsübungen

In der arabischen Schriftsprache werden, wie in vielen semitischen Sprachen, von wenigen Ausnahmen wie einigen Eigennamen und dem Qur'an abgesehen, die Vokale nicht mitgeschrieben, wenn sie nur kurz gesprochen werden. Die für diese Vokale existierenden Zeichen sowie die Zeichen für eine Konsonantenverdoppelung, Vokallosigkeit und grammatische Endungen sind nur fakultativ. Dadurch können Mehrdeutigkeiten in der Bedeutung von unvokalisierten Worten entstehen. Bei diesem Übungstyp wird nun speziell getestet, ob solche Unterschiede bekannt sind und die Fähigkeit der richtigen Aussprache gefördert.

1.2.2.3.4. Datumsangaben

Die Angabe des Datums im Arabischen (in ausgeschriebener Form) erfolgt nach einer anderen Regelung als es in den indogermanischen Sprachen üblich ist. Die Angabe z.B. des Datums 1. Mai erfolgt in der Form "am 1. Tage des Monats Mai". In Verbindung mit der Angabe des Jahres erweitert sich dieses Konstrukt nochmals.

1.2.2.3.5. Lückentexte

Haben sich die obigen Übungstypen hauptsächlich mit der Festigung der Zuordnung von deutschen zu den arabischen (und umgekehrt) Vokabeln beschäftigt, liegt der Fokus dieses Übungstyps auf der Stärkung der Fähigkeit, den Sinn eines Satzes auch dann noch zu erkennen, wenn ein Wort fehlt. Weiterhin ist es bei dieser Übung besser möglich, die Konjugation der Verben am Beispiel zu üben. Auch einfache Dialoge können so dargestellt werden. Außerdem können die Vokabeln in bestimmten Kontexten geübt werden, was der Forderung nach situationsbezogenem Lernen Rechnung trägt.

2. Technische Betrachtungen

2.1. Ersterfassung der Anforderungen zur Übungssoftware Deutsch<->Arabisch

Die zu erstellende Software sollte folgende Anforderungen erfüllen:

1) Allgemeines

- lauffähig auf IBM-kompatiblen Rechnern mit Windows 95
- Emulation der arabischen Tastaturbelegung ohne Nutzung von Systemzusätzen

2) Autorenspezifisches

- leichte Eingabe- / Änderungsmöglichkeit der Übungen
- Bereitstellung folgender Übungstypen:
  - Vokabelübungen, Pluralbildung
  - Vokalisierungsübungen
  - Datumsangaben
  - Lückentexte
  - Multiple-Choice-Aufgaben
  - interaktive Schautafeln

3) Nutzerspezifisches

- leicht verständliche Oberfläche
- intuitive Benutzerführung
- Möglichkeit der Wiederholung von früher fehlerhaft bearbeiteten Aufgaben
- Einblendung der arabischen Tastaturbelegung permanent möglich
- Verwaltung mehrerer Benutzer

2.2. Spezifikation

2.2.1. Zielbestimmung / Motivation

Im Zuge der Reform der universitären Ausbildung, die auch vor der Sprachausbildung nicht Halt macht, wird es immer wichtiger, daß sich die Lehrkräfte auf die Vermittlung von neuen Lerninhalten konzentrieren können. Da jedoch gerade bei dem Sprachstudium Übungen eine nicht zu unterschätzende Komponente darstellen, soll mit diesem Programm die Übungsbetreuung übernommen oder mindestens erleichtert werden.
Daher soll für das Orientalische Institut der Universität Leipzig ein System erstellt werden, das dieser Aufgabe gerecht wird.

2.2.2.Produkteinsatz

2.2.2.1. Anwendungsbereiche

Das System soll zur Unterstützung des Lernens der arabischen Sprache eingesetzt werden. Es soll sich sowohl für den inneruniversitären Einsatz als auch für den Einsatz zu Hause beim Selbststudium eignen.

2.2.2.2. Zielgruppen

Die typische Benutzergruppe sind Studenten der arabischen Sprache im Haupt- und Nebenfach ab dem ersten Semester. Die Kenntnis der arabischen Schriftzeichen wird allerdings vorausgesetzt, so daß der effektive Einsatz ab etwa Mitte des ersten Semesters möglich ist. Ebenfalls vorausgesetzt werden Grundkenntnisse im Umgang mit Computern (Tastaturbedienung, Programmstart etc.).

2.2.3. Produkt-Umgebung

2.2.3.1. Software

Als Zielsystem wird Windows 95® vorausgesetzt, welches in der westeuropäischen Ausführung vorzuliegen hat. Die erste Version erfordert weiterhin, daß ein deutsches Tastaturlayout vorliegt.

2.2.3.2. Hardware

Grundsätzlich soll jeder IBM-kompatible Rechner, auf dem Windows 95® lauffähig ist, unterstützt werden. Für einen optimalen Betrieb werden

jedoch mindestens ein 80486 DX4@100MHz und 16MB Hauptspeicher vorausgesetzt.
Außerdem sollte die Grafikkarte mindestens 265 Farben sowie 640 x 400 Bildpunkte darstellen können.

2.2.4. Funktionsbeschreibung

2.2.4.1. Basismodus

2.2.4.1.1. Anmelden des Benutzers

Jeder Benutzer erhält seine eigene Arbeitsumgebung, in der der aktuelle Stand der Übungen und die gemachten Fehler abgelegt werden. Die Authentifizierung des Benutzers erfolgt mittels Paßwort.

2.2.4.1.2. Wiederholung

Die bei früheren Übungssitzungen fehlerhaft bearbeiteten Aufgaben werden zur Wiederholung angeboten. Dabei wechselt das System in den Übungsmodus.

2.2.4.1.3. Lektionsauswahl

Nach der Wiederholung kann die als nächstes zu bearbeitende Übungseinheit ausgewählt werden, mit der das Programm dann im Übungsmodus fortsetzt.

2.2.4.1.4. Abmelden des Benutzers

Bei der Beendigung einer Übungssitzung werden der neue Status des Benutzers sowie die Fehlerliste abgelegt. Außerdem wird der Druck der fehlerhaft bearbeiteten Aufgaben mit den richtigen Lösungen angeboten.

2.2.4.2. Übungsmodus

2.2.4.2.1. Vokabelübungen

Hier werden Übersetzungsübungen in beiden Richtungen sowie Übungen der arabischen Grammatik wie z.B. Pluralbildung angeboten. Es erfolgt eine binäre Auswertung der Einzelübungen (richtig / falsch).

2.2.4.2.2. Vokalisierungsübungen

Hier wird die Fähigkeit der richtigen Vokalisierung, d.h. der Vervollständigung von arabischen Worten mit Vokal-Hilfszeichen überprüft. Die Auswertung erfolgt in einem ternären System. (richtig / unvollständig / falsch)

2.2.4.2.3. Datumsangaben

Die richtige Angabe eines Datums in ausgeschriebener Form in der arabischen Sprache wird hier überprüft. Die aus mehreren Worten bestehende Benutzereingabe wird binär bewertet (richtig / falsch).

2.2.4.2.4. Lückentexte

Nachdem der vollständige arabische Text angezeigt wurde, wird der Benutzer aufgefordert, die Lücken im danach angezeigten Text zu füllen. Die Auswertung erfolgt für jede Lücke getrennt binär (richtig / falsch).

2.2.4.2.5. Multiple-Choice-Aufgaben

Auf eine Frage zu einer beliebigen Thematik werden mehrere mögliche Antworten vorgegeben, von denen nur eine richtig ist. Die Auswertung erfolgt binär (richtig / falsch).

2.2.4.2.6. Interaktive Schautafeln

In Tabellenform werden z.B. grammatikalische Probleme oder kurze Dialoge dargestellt. Aufgabe des Benutzers ist es, offene Einträge zu ergänzen. Die Auswertung erfolgt für jeden Eintrag getrennt binär (richtig / falsch).

2.2.5. Produktdaten

Zu jedem eingetragenen Benutzer wird eine Datei mit den Daten über bereits abgearbeitete Übungseinheiten und fehlerhaft bearbeitete Aufgaben abgelegt.

2.2.6. Leistungsanforderungen

Das System soll als Single-User-System ausgeführt werden. Die Auswertung einer Einzelübung soll maximal 1 Sekunde Bearbeitungszeit benötigen. Die Anmeldeprozedur soll nach Angabe von Namen und Paßwort im korrekten Fall maximal 5 Sekunden benötigen.

2.2.7. Schnittstellen

2.2.7.1 Schnittstelle zum Benutzer

Das User-Interface soll aus einem dreigeteilten Hauptfenster bestehen.
Die 3 Teile des Hauptfensters dienen

- der Darstellung von Statusinformationen

- der Navigation innerhalb der Übungseinheiten
- als Arbeitsbereich für die Einzelübungen.

2.2.7.2 Schnittstelle zu den Übungen

Zur einfachen Erstellung von Übungen für das System sollen die Übungseinheiten in getrennten Dateien eines Verzeichnisses in Textform vorliegen.

2.2.8. Entwicklungsumgebung

2.2.8.1 Software

Zur Entwicklung des Systems wird die Verfügbarkeit folgender Produkte vorausgesetzt:

- Microsoft Windows 95®
- Microsoft Visual C++ V 4.2 oder höher
- ein TrueType-Font®-Editor; aus Gründen der Verfügbarkeit wird hier TypeSmith 2.5b von Soft-Logik unter Amiga-DOS® 3.1 verwendet. Grundsätzlich kann auch jeder andere TrueType-Font®-Editor verwendet werden.

2.2.8.2 Hardware

Für den Betrieb der Entwicklungswerkzeuge werden benötigt:

- IBM-kompatibler PC mit Pentium®-Prozessor oder kompatiblem Prozessor anderer Hersteller bei mindestens 150MHz
- 32MB Hauptspeicher

Wird TypeSmith® eingesetzt, wird weiterhin benötigt:

- Ein Amiga-Computer mit mindestens Motorola 68020, besser 68040-Prozessor (z.B. Amiga 4000) und
- mindestens 10 MB Hauptspeicher

2.3. Ablauforganisation

2.3.1. Überblick über gebräuchliche Organisationsvarianten in der Software-Entwicklung

2.3.1.1. Wasserfallmodell

Das erste überhaupt strukturierte Prozeß-Modell nach der sogenannten Softwarekrise war das Wasserfallmodell. Die Neuerung in der Software-Entwicklung war die logische Gliederung des Entwicklungsprozesses in verschiedene Phasen mit definiertem Inhalt und Ergebnis. Ein einfaches Wasserfallmodell, wie in [You 93] vorgestellt, besteht aus den 4 Phasen Analyse, Design, Implementierung und Test. In seiner ursprünglichen Form war das System so ausgelegt, daß das Ergebnis einer Phase in die nächste Phase "hinüberschwappt", was eine strenge Serialisierung zur Folge hatte. Außerdem war eine Rückkehr in vorherige Phasen nicht vorgesehen. Dadurch konnte es zu Problemen kommen. In späteren Verfeinerungen wurden die Phasen in mehrere Teilphasen aufgespalten und die Rückkehr zur direkt vorgelagerten Phase zwecks Fehlerbehebung ermöglicht. Das machte das System schon wesentlich flexibler und ist bis heute in dieser Form durchaus noch gebräuchlich. Das Ausmaß der Serialisierung der Phasen ist in der modernen Wissenschaft nicht eindeutig geklärt. Es finden sich sowohl Anhänger der strengen Serialisierung als auch Vertreter des anderen Extremes, der zeitparallelen Ausführung aller Phasen. (Vgl. dazu [You 93] und [Bal 98]).

Die mehrheitliche Meinung ist jedoch, daß dieses Modell einen entscheidenden Nachteil hat: es ist dokumentengetrieben, d.h. ohne die am Ende einer jeden Phase anzufertigenden Dokumente ist das System nicht überprüfbar. Dadurch entsteht die Gefahr, daß das Hauptaugenmerk weg von dem eigentlichen Softwareprodukt hin zu den Dokumenten gelenkt wird. Außerdem ist das Modell sehr unflexibel bei der Berücksichtigung der Nutzerwünsche. Einmal festgelegt, können die Anforderungen kaum noch geändert werden. Außerdem wird davon ausgegangen, daß zu Beginn der Entwicklung alle korrekten Anforderungen vorliegen und diese nicht mehr geändert werden, was oft mit der Praxis kollidieren dürfte. Weiterhin liegen für den Auftraggeber erst relativ spät sichtbare Ergebnisse vor. Dadurch steigt die Gefahr einer Fehlentwicklung.

Diese gravierenden Nachteile dieses Modells lassen es als nicht sinnvoll für das vorliegende Projekt einsetzbar erscheinen. Vor allem die starre Handhabung des Entwicklungsablaufes und der unterschätzte Einfluß der sich möglicherweise ändernden bzw. konkretisierenden Nutzerwünsche machen das Wasserfallmodell zu einem schlechten Kandidaten für die Entwicklungsorganisation dieses Projekts.

2.3.1.2. V-Modell

Das V-Modell ist strenggenommen nur eine Erweiterung des Wasserfall-Modells um die Qualitätssicherung. Ein auf diesem Modell basierendes System wird heute noch in der Softwareentwicklung der Bundesbehörden angewendet. Durch die integrierte Qualitätssicherung mutiert das V-Modell zu einem Modell, das einen wesentlich ganzheitlicheren Ansatz verfolgt (siehe [Bal 98], S.109: "... erhebt den Anspruch auf Allgemeingültigkeit ...") und auch auf die Entwicklung von Hardware ausgedehnt werden kann. Dadurch jedoch wird es für die Verwendung bei kleineren Projekten schnell zu unhandlich und schwerfällig, da hier meist auf den Einsatz von CASE-Tools verzichtet wird.
Bei der Betrachtung der Tauglichkeit für die Organisation des vorliegenden Projekts fallen die gleichen Kritikpunkte wie beim Wasserfallmodell auf. Außerdem wird durch den verfolgten ganzheitlicheren Ansatz beim V-Modell der Verwaltungsaufwand stark aufgestockt. Somit ist dieses Modell für die Entwicklung eines vergleichsweise kleinen Projekts zu schwerfällig.

2.3.1.3. Prototypen-Modell

Eine andere Weiterentwicklung des Wasserfallmodells stellt das sogenannte Prototypen-Modell dar. Das Prototypen-Modell ist eigentlich eine Gruppe von Modellen. Es bricht bezüglich der Grundidee mit dem Wasserfallmodell, erst alle Anforderungen zu sammeln und dann das Erzeugnis zu erstellen. Hier geht man davon aus, daß es eine legitime Art und Weise der Softwareentwicklung darstellt, die Benutzer-Anforderungen Schritt für Schritt zu ermitteln.
Das Hauptproblem bei dem verfeinerten Wasserfallmodell war die mangelnde Einbeziehung des Auftraggebers bzw. Endbenutzers in den Entwicklungsprozeß. Oftmals ist es nicht möglich, zu Beginn der

Entwicklung eine vollständige Anforderungsliste an das Produkt zu erhalten. Da jedoch beim Wasserfallmodell der Dialog mit dem Auftraggeber spätestens am Ende der Analysephase eingefroren wird, besteht die Gefahr, am Bedarf vorbei zu entwickeln. Genau an dieser Stelle greift das Prototypen-Modell ein. Es gestattet den Dialog zwischen Anwender und Entwickler über alle Bereiche der Entwicklung hinweg und ermöglicht auch eine Art Alternativentest bei bestimmten Problemen.
Die Gruppe des Prototypen-Modells umfaßt mehrere Untertypen, von denen die wichtigsten Repräsentanten hier kurz vorgestellt werden sollen:

Demonstrationsprototyp

Er dient meist dazu, einen Auftrag zu erhalten. Da er wenig Kosten verursachen soll, wird er oft unter Vernachlässigung aller Standards entwickelt, dient er doch nur zur prinzipiellen Darstellung des Produktes. Nach der Demonstration wird er weggeworfen.

Prototyp im engeren Sinne

Er dient der Veranschaulichung bestimmter Probleme während der Modellierung. Er ist ein ablauffähiges Provisorium.

Labormuster

Dieser Prototyp dient nicht der Entwickler-Auftraggeber-Kommunikation, sondern wird aufgebaut, um Alternativen bei der Konstruktion besser einschätzen zu können.

Pilotsystem

Ein Prototyp der etwas anderen Art ist das Pilotsystem. Wurden alle anderen Prototypen nach Gebrauch weggeworfen, fließt ein Pilotsystem mit in das Endprodukt ein. Das hat natürlich Konsequenzen bezüglich der Entwicklung eines solchen Systems, da hier die gültigen Standards der Softwareentwicklung beachtet werden müssen. Der Aufbau eines Pilotsystems ermöglicht die ständige Kontrolle der Anforderungen durch den Auftraggeber, da er am Entwicklungsprozeß durch Anpassung der Vorgaben mit teilnehmen kann. Die Abgrenzung von einem Pilotsystem zu der evolutionären Entwicklung von Software ist nicht klar definiert.

Ein etwas anderer Blickwinkel auf die Problematik der Prototypen läßt eine alternative Unterscheidung der Prototypen in horizontale und vertikale zu.

Horizontaler Prototyp

Ein Softwareprodukt ist meist in mehrere Ebenen gegliedert oder zumindest gliederbar. Wird nun nur eine solche Ebene (z.B. die Benutzeroberfläche) zu Demonstrationszwecken aufgebaut, spricht man von einem horizontalen Prototyp.

Vertikaler Prototyp

Im Gegensatz zu dem horizontalen Prototypen wird bei einem vertikalen nur eine bestimmte (Anwender-)Funktionsgruppe aufgebaut und die anderen Funktionen bleiben unberücksichtigt. Die Implementierung dieser Funktion erfolgt jedoch durch alle Ebenen hindurch.
Die Vorteile des Prototypen-Modells sind in der flexibleren Handhabung der Reihenfolge der Entwicklungsphasen und der verbesserten Einflußnahme des Auftraggebers auf den Entwicklungsprozeß zu sehen. Durch den permanenten Informationsfluß zwischen Entwickler und Auftraggeber wird die Gefahr einer Fehlentwicklung stark eingeschränkt.

2.3.1.4. Das evolutionäre Modell

Einen anderen Weg, den Auftraggeber in die Entwicklung mit einzubeziehen, geht das evolutionäre Modell. Nachdem die grundlegenden Kernanforderungen definiert wurden, wird das so festgelegte Produkt straight-forward entwickelt und an den Auftraggeber als Nullversion ausgeliefert. Dieser ermittelt weitere Anforderungen an das Produkt, die dann in die nächste Version mit einfließen usw.
Diese Verfahrensweise eignet sich besonders dann, wenn dem Auftraggeber selbst die Anforderungen an das Produkt nicht vollständig klar sind. Insofern bestehen Parallelen zum Pilotsystem.
Allerdings hat diese Flexibilität den Preis, daß unter Umständen nicht erkannte Kernanforderungen zu einer vollständigen Überarbeitung der Systemarchitektur führen können. Daher ist eine vollständig offene Architektur hier angebracht, was aber zu Verzögerungen der Implementierung und weniger effizientem Code führen kann.
Seine Stärken kann das evolutionäre Modell hauptsächlich bei der Entwicklung für einen anonymen Markt ausspielen. Bei der Entwicklung für einen speziellen Kunden oder Kundenkreis, zu dem ständiger Kontakt besteht, hat das evolutionäre Modell leichte Nachteile zu dem Prototypen-Modell.

2.3.1.5. Das inkrementelle Modell

Auch das inkrementelle Modell verfolgt den Ansatz, den Benutzer mit in die Entwicklung mit einzubeziehen, versucht jedoch die Nachteile des evolutionären Modells zu vermeiden. Dies soll durch eine möglichst vollständige Aufnahme der Kernanforderungen erfolgen. Trotzdem wird zuerst nur ein Teil der Anforderungen implementiert und der Benutzer kann die ersten Erfahrungen mit dem System frühzeitig sammeln. Durch die vollständige Analyse zu Beginn der Entwicklung wird vermieden, daß die Ziele mit der Architektur kollidieren können. Damit kann man das inkrementelle Modell als eine Art Kaskade von mehreren Wasserfallmodellen ansehen.

Dieses Modell eignet sich auch zur Entwicklung von Produkten für den anonymen Markt, da der Feedback der Nutzer in den nächsten Versionen mit berücksichtigt werden kann. Auch ist die Gefahr von aufgeblähtem Code durch die starke Bemühung, ein offenes System zu erhalten, stark eingeschränkt.

Das inkrementelle Modell hat wie das evolutionäre Modell seine Stärken bei der Entwicklung von Produkten, die eine breite Anwendung finden sollen. Bei der Entwicklung von spezielleren Programmen hat das Modell ebenfalls leichte Nachteile zum Prototypen-Modell, wenn auch diese weniger stark ausfallen.

Die offensichtliche Nähe des inkrementellen zum evolutionären Modell führt in der Literatur oft dazu, daß hier keine Unterscheidung zwischen diesen beiden vorgenommen wird.

2.3.1.6. Das objektorientierte Modell

Das objektorientierte Modell legt sein Hauptaugenmerk auf die Mehrfachnutzung von Code, so daß die Entwicklungskosten gesenkt werden können. Außerdem ist auch eine höhere Güte dadurch erreichbar, daß man bereits in anderen Anwendungen vorhandenen Code, der dort bereits getestet wurde, einsetzt. Dadurch werden nicht nur Entwicklungskosten und -zeit gespart, sondern auch Kosten der Wartung. Die Wiederverwendung von Code kann auf mehreren Ebenen ansetzen. So ist es möglich, ganze Subsysteme zu übernehmen, was Wiederverwendung in der Analysephase entspricht. Der Wiederverwendung auf der Ebene des Designs entspricht die Übernahme technischer Entwürfe. Schließlich sind

auch noch während der Implementierung über Klassenbibliotheken erhebliche Zeitersparnisse möglich.
Obwohl es den Anschein hat, das objektorientierte Modell sei erst in der jüngeren Vergangenheit entstanden, finden sich die ersten Wurzeln dazu bereits in den 70er Jahren. Jedoch fehlten damals noch die entsprechenden Programmiersprachen, die dieses Modell unterstützten. Mit Aufkommen entsprechender neuer oder erweiterter Sprachen wie Smalltalk und C++ wurde das Interesse an diesem Modell wieder verstärkt.
Das System der Objektorientierung wurde nacheinander in die verschiedenen Phasen der Softwareentwicklung eingeführt. Dabei ist es interessant, daß zuerst die nachgelagerten Phasen diese Idee übernahmen, bevor die Definitionsphase auch dieses Modell benutzte. Als erstes wurden die Programmiersprachen bekannt, welche die Objektorientierung unterstützten (OOP), dann bediente sich auch das Design dieser Idee (OOD) und schließlich wurde die Objektorientierung sogar in die Analyse eingeführt (OOA). In dieser Zeit entstanden die unterschiedlichsten Interpretationen der Grundidee (Vgl. auch [Bal 96],[Bal 98],[Boo 94]) und die verschiedensten Arten der Notationen während der Analyse- und Designphase. Zur Zeit wird daran gearbeitet, eine einheitliche Notation festzulegen, die die Vorteile der wichtigsten Strömungen konsistent miteinander verbindet.
Durch seinen Schwerpunkt auf der Wiederverwendbarkeit von Programmteilen eignet sich das objektorientierte Modell für alle Arten von Programmen. Da bei dem objektorientierten Modell keinerlei Aussagen über die Einbindung des Auftraggebers in den Entwicklungsprozeß und den Entwicklungsablauf gemacht werden, läßt sich dieses Modell gut mit einem anderen kombinieren, um eine für das spezielle Projekt maßgeschneiderte Lösung zu erhalten.

2.3.1.7. Das nebenläufige Modell

Wie der Name schon andeutet, wird beim nebenläufigen Modell versucht, die einzelnen Phasen der Entwicklung, die eigentlich mehr oder weniger sequentiell ablaufen sollen, zu parallelisieren. Man verspricht sich so eine Verkürzung der Entwicklungszeit. Dies soll dadurch zustande kommen, daß von Anfang an alle Parteien, die mit dem Erzeugnis in Berührung kommen, mit einbezogen werden. Diese Technik ist schon seit längerem aus der Fertigungsindustrie bekannt und soll nun gewissermaßen auf die

Softwareentwicklung portiert werden. Ein stärkerer Zielbezug sowie vielfältige arbeitsorganisatorische Eigenschaften charakterisieren dieses Modell. Obwohl Parallelitäten zu dem evolutionären und inkrementellen Modell sichtbar werden, ist es bei diesem Modell von vornherein das erklärte Ziel, ein vollständiges Produkt auszuliefern.

2.3.2. Die eingesetzte Methode

Da mit Visual C++ ein Vertreter der Programmiersprachen gewählt wurde, der die objektorientierte Methode auf der Ebene der Implementierung unterstützt und bei Verwendung der MFC-Bibliothek einem nahezu unmöglich macht, eine andere Technik zu verwenden, liegt es nahe, die objektorientierten Methode in allen Ebenen zu nutzen. Nur so ist eine konsistente Entwicklung möglich. Nutzt man während der Analyse und des Designs eine andere Methode, ist es unumgänglich, die Ergebnisse auf die OOP-Technik zu portieren. Dadurch entsteht in den meisten Fällen ein Strukturbruch, der nur durch die konsequente Verwendung von OOA und OOD vermieden werden kann.
Da die OO-Methode relativ offen ausgelegt werden kann, liegt es nahe, die Vorteile dieser Methode mit den Vorteilen des Prototypen-Ansatzes zu verbinden. Bei dem vorliegenden Projekt erscheint die Entwicklung eines Pilotsystems vorteilhaft, da damit die Stärken der Objektorientierung mit der Flexibilität der Benutzer-Entwickler-Interaktion verbunden werden können.

2.4. Entwurf

2.4.1. Allgemeines

2.4.1.1. Wahl der zu nutzenden Programmiersprache

Aus der Vielzahl der heutzutage zur Verfügung stehenden Programmiersprachen und Programmiersystemen gilt es die optimale herauszufinden. Als erstes Auswahlkriterium wäre die Verfügbarkeit für Windows 95 bzw. der Codegenerierung für diese Plattform anzuführen. Weiterhin ist es vorteilhaft, wenn bereits einige Vorkenntnisse in der Nutzung der Sprache vorliegen. Da in der heutigen Zeit der objektorientierte Ansatz bei der Softwareentwicklung state of the art zu sein scheint, beschränkt sich die Auswahl also auf OOP-Pascal und C++ bzw. deren Derivate verschiedener Hersteller.
Weiterhin sind in den letzten Jahren visuell unterstützte Entwicklungsumgebungen auf den Markt gekommen. Diese erleichtern zumindest den Aufbau der Programmoberfläche, was zu einer Verkürzung der Entwicklungszeit führen sollte.
Zur Auswahl bleiben meinem Wissen nach im wesentlichen nur 2 Alternativen: Borland Delphi und Microsoft Visual C++. Da Delphi auf Pascal basiert, wurde Visual C++ der Vorrang gegeben, da diese Programmiersprache im allgemeinen den kompakteren und effizienteren Code generiert.

2.4.1.2. Arabischer Zeichensatz

Da ein in Europa erhältliches Windows nicht über einen Zeichensatz der arabischen Schrift verfügt, muß mit dem Programm ein solcher mitgeliefert werden. Dabei sind 2 Alternativen möglich. Entweder findet ein Zeichensatz eines Drittherstellers oder ein selbst generierter Verwendung. Die erste Alternative hat das Problem, daß die meisten Zeichensätze entweder kommerziell erstellt wurden und somit erst lizenziert werden müßten oder auf Shareware-Basis vorliegen und oft von minderer Qualität sind. Die zweite Variante zieht zwar einen großen Arbeitsaufwand nach sich, umgeht aber die obigen Probleme. Daher wird in diesem Projekt dieser Weg beschritten.
Weiterhin ist zu überlegen, wie die Kodierung des arabischen Textes im Programm erfolgen soll. Hier sind 3 Möglichkeiten denkbar: Als erstes besteht die auf den ersten Blick sehr einfache Möglichkeit, eine eigene

Kodierung zu entwickeln und zu nutzen. Diese Idee scheitert jedoch nicht nur an den Richtlinien guter Programmierung, da bereits Standards existieren. Sie ist auch problematisch im Hinblick auf die Erstellung der Lektionsdateien. Variante 2 ist die Nutzung der Windows-Codepage für Arabisch, wie sie in den arabischen Windows-Versionen verwendet wird. Die Verwendung dieser Variante hätte den Vorteil, die Lektionsdateien mit einem handelsüblichen Editor auf einem arabischen Windows-System generieren zu können. Als Nachteil muß angeführt werden, daß die Darstellung der Schrift dem Programm überlassen wird, insbesondere die Logik der Zeichenposition im Wort, da in der arabischen Schriftsprache ein Buchstabe in verschiedene Grapheme zerfällt. Weiterhin bliebe es dem Anzeigeprogramm überlassen, die im Arabischen weit verbreiteten Ligaturen zu generieren. Diese sind jedoch bis auf die lam-alif-Ligaturen fakultativ.

Als Variante 3 ist die Verwendung von Unicode zu überprüfen. Dieses Kodiersystem ist mittlerweile standardisiert (ISO 10646) und umfaßt 448 Ligaturen sowie alle Formen der Einzelbuchstaben und Hilfszeichen. Damit wären die meisten Probleme der Variante 2 abgedeckt. Dieser Vorteil wird jedoch von 2 nicht zu vernachlässigenden Nachteilen wieder zunichte genacht:

Erstens ist Unicode ein 2-Byte-Code, was den Einsatz eines einfachen Editors zur Lektionsdateierstellung unmöglich macht (mir ist zumindest kein Texteditor bekannt, der Unicode erzeugt) und zweitens wird Unicode von Windows 95 nicht unterstützt. Windows NT unterstützt zwar Unicode, sieht sich aber (zumindest in der westeuropäischen Variante) nicht in der Lage, die arabischen Pages anzubieten.

Da als Zielplattform Windows 95 fixiert wurde, kommt nur der Einsatz der Variante 2 in Frage. Dadurch ist das Problem der Ligaturen weiterhin existent.

Es existieren zwar Unmengen von fakultativen Ligaturen in der modernen arabischen Schriftsprache, jedoch sind viele kaum von der normalen Hintereinanderschreibung der meist 2 Buchstaben zu unterscheiden oder aber aus so vielen Buchstaben bestehend und so komplex, daß sie ohne eingehende Sprachkenntnisse nicht mehr entziffert werden können. Die weite Verbreitung der Kalligraphie verschärft diesen Umstand noch. Gerade im Anfangsstadium des Lernens der arabischen Sprache wirkt die Fülle der verschiedenen Ligaturen auf den Lernenden nur verwirrend.

Für das vorliegende Projekt ist es somit vorteilhafter, nur die obligatorische lam-alif-Ligatur mit den verschieden Hilfszeichen sowie einige Ligaturen von Hilfszeichen zu verwenden, die die Lesbarkeit des Textes zumindest nicht beeinträchtigen, meist sogar etwas verbessern.
Für weitergehende Betrachtungen zu Ligaturen in der arabischen Schriftsprache vgl. [Bol 94].

2.4.2. Identifizierung der Programm-Module

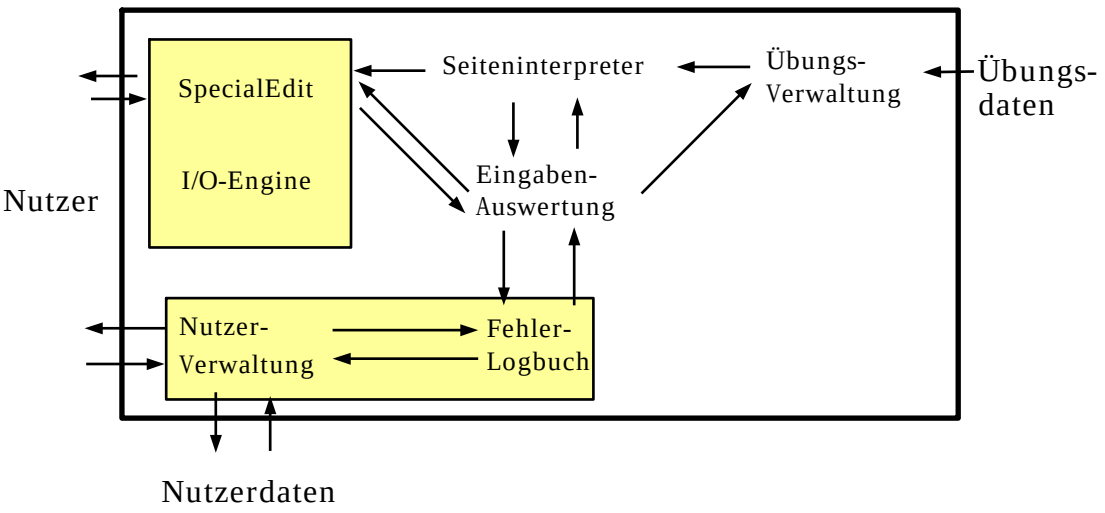
Zur effektiven Bearbeitung der dem Programm gestellten Aufgaben ist es unerläßlich, das Programm in Module zu zerlegen. Diese Zerlegung ist auch bei späteren Erweiterungen und eventuellen Fehlerkorrekturen hilfreich, da der Ort der Änderung leicht nach Aufgabenkomplexen zu identifizieren ist.

2.4.2.1. Modularisierung des Programms

Folgende Funktionsblöcke sind zur spezifikationsgetreuen Arbeit des Programms nötig:

- Verwaltung der Nutzerdaten
- Verwaltung der Übungen
- Kommunikation mit dem Nutzer
- Bewertung der Nutzereingaben zu den Fragestellungen
- Zerlegung der Übungen in Aufgaben
- Verarbeitung der fehlerhaft bearbeiteten Aufgaben

Damit ergibt sich folgende Gliederung in Module und deren Kommunikationswege:



In der Nutzerverwaltung werden alle Daten eines Nutzers wie sein Name und bereits bearbeitete Aufgaben (richtige und fehlerhafte) gesammelt und archiviert. Der Seiteninterpreter ist für die Zerlegung einer Übungseinheit in Aufgaben und die Identifizierung des jeweiligen Aufgabentyps zuständig. Die Daten erhält er von der Übungsverwaltung, die die Zuordnung der Aufgabenblöcke zu den jeweiligen Dateien als Aufgabe hat. Die so vorbereiteten Daten werden der I/O-Engine übergeben, die für die Darstellung und die Entgegennahme der Nutzer-Eingaben zuständig ist. Auf Grund technischer Zwänge muß das Eingabecontrol für arabischen Text als faktisch eigenes Modul implementiert werden. Es steht jedoch nur mit der restlichen I/O-Engine in engem Kontakt. Somit bilden diese beiden Module eine logische Einheit. Das Fehlerlogbuch ist für die Verwaltung der fehlerhaft beantworteten Aufgaben zuständig und leitet unter Umständen Wiederholungen von Aufgaben ein. Da man die Übungsfehler als Eigenschaft des Benutzers betrachten kann, besteht zwischen Fehlerlogbuch und Nutzerverwaltung eine sehr enge Bindung.

2.4.2.2. Aufgaben der internen Modulschnittstellen

2.4.2.2.1. Nutzerverwaltung <-> Fehlerlogbuch

Bei einem bereits vorhandenem Nutzer werden von der Nutzerverwaltung der Name, das Datum der letzten Übungssitzung und die fehlerhaft bearbeiteten Aufgaben an das Fehlerlogbuch übergeben. Im Falle der Neuanlage eines Nutzers werden diese Werte initialisiert. Bei der Beendigung einer Übungssitzung durch einen Nutzer werden die neuen Daten zwecks Archivierung an die Nutzerverwaltung übergeben.

2.4.2.2.2. I/O-Engine <-> Eingaben-Auswertung

Sowohl die Aufgabenlösungen, die vom Nutzer eingegeben wurden, als auch eventuelle Navigationseingaben (Wechsel der Lektion etc.) werden von der I/O-Engine an die Eingaben-Auswertung weitergereicht, die ihrerseits Rückmeldungen über die Richtigkeit der Lösungen abgibt.

2.4.2.2.3. Eingaben-Auswertung <-> Fehlerlogbuch

Zur statistischen Erfassung und eventuellen späteren Wiederholung werden fehlerhaft gelöste Aufgaben an das Fehlerlogbuch gemeldet. Bei der Wiederholung fehlerhaft gelöster Aufgaben werden die Nummern der

Aufgaben der Eingaben-Auswertung zur Einleitung der notwendigen Aktionen übergeben.

2.4.2.2.4. Eingaben-Auswertung -> Übungsverwaltung

Im Falle eines Lektionswechsels, der vom Nutzer eingeleitet wird, wird diese Information an die Übungsverwaltung weitergeleitet.

2.4.2.2.5. Eingaben-Auswertung <->Seiten-Interpreter

Dem Seiten-Interpreter werden Kommandos über die Navigation innerhalb einer Lektion mitgeteilt, so daß dieser die entsprechende Aufgabe vorbereiten kann. Die Navigationskommandos können sowohl relativer (nächste / vorherige Aufgabe) als auch absoluter Natur (Aufgabennummer) sein. Es erfolgen Rückmeldungen über den Erfolg der Aktionen. Außerdem erhält die Eingaben-Auswertung die richtigen Lösungen der Aufgabe.

2.4.2.2.6. Übungsverwaltung -> Seiten-Interpreter

Dem Seiten-Interpreter wird die aktuelle Lektion zur Bearbeitung übergeben.

2.4.2.2.7. Seiten-Interpreter -> I/O-Engine

Vom Seiteninterpreter werden die Daten der aktuellen Aufgabe zur Darstellung an die I/O-Engine gesendet. Die Daten beinhalten sowohl den Typ der Aufgabe als auch die Aufgabentexte.

2.4.3. Aufgaben und Funktion der Module

2.4.3.1. Nutzer-Verwaltung

Die Aufgabe der Nutzerverwaltung ist es, alle Benutzer des Programms und den Status ihrer Arbeit zu erfassen. Weiterhin ist die Nutzerverwaltung dafür zuständig, den Status beim Start einer neuen Übungssitzung auf den Status bei Ende der letzten Sitzung zu restaurieren.

Ein Benutzer wird über seinen Namen identifiziert. Jeder Benutzer erhält seine eigene Datei. Der Dateiname wird über eine Hashfunktion des Nutzernamens ermittelt und besteht aus einer 6-stelligen Buchstabenfolge. Da diese direkt aus dem 24-Bit-Hashwert erzeugt wird, sollte genügend Raum für die Aufnahme von Nutzer vorhanden sein.

Die Statusdaten werden mittels eines vom Benutzer wählbaren Paßwortes geschützt. Als Verschlüsselungsalgorithmus wird CAST-128 wie in [Ada 97] beschrieben verwendet. Dabei arbeitet die Verschlüsselung im ECB-Modus. Die Wahl fiel auf diesen Algorithmus, da er erstens zur freien Verwendung sowohl in kommerziellen als auch in nicht-kommerziellen Anwendungen zugelassen wurde und zweitens einen hohen Datendurchsatz besitzt (nach Angaben des Autors ca. 3,3 MByte/s auf einem 150MHz Pentium Prozessor; vgl. [Ada 97])).

Der "Magic Cookie" am Anfang der zu verschlüsselten Daten ist ein konstanter Wert. Dies ermöglicht zwar einen known- plaintext- Angriff, jedoch ist der Algorithmus stark genug, daß die Kosten für einen Angriff um Dimensionen höher liegen sollten als der Wert der gesicherten Daten.

Die Authentifizierung des Benutzers erfolgt durch Entschlüsselung des ersten Blocks der Statusdaten. Wurde der "Magic Cookie" wiederhergestellt, ist das Paßwort korrekt (zumindest sollte die Wahrscheinlichkeit, den korrekten "Magic Cookie" bei falschem Paßwort zu erhalten, bei einem starken Verschlüsselungsverfahren, wie CAST-128 eines ist, infinitesimal klein sein) und der Benutzer zur Programmnutzung berechtigt. Die Daten werden dann an das Fehlerlogbuch übergeben, um den alten Zustand des Systems wiederherzustellen und unter Umständen eine Wiederholung von fehlerhaft bearbeiteten Aufgaben einzuleiten.

Die Nutzerverwaltung besitzt ihre eigene Oberfläche zur Kommunikation mit dem Nutzer in Form eines Dialoges, der bei Bedarf eingeblendet wird.

2.4.3.2. Übungsverwaltung

Die Aufgabe der Übungsverwaltung ist es, die Zuordnung zwischen den logischen Lektionsnummern und den physischen Lektionsdateien herzustellen. Jede Lektion ist in einer separaten Datei beliebigen Namens abgelegt. Die Zuordnung wird beim Programmstart aus einer speziellen Verzeichnisdatei gelesen (bei Bedarf wird diese erst erzeugt) und in einem Suchbaum gespeichert.

Bei der Anfrage zu einer Lektion wird der Dateiname ermittelt und die Datei geöffnet und dem Seiteninterpreter übergeben. Beim Wechsel der Lektion bzw. Beenden des Programms wird die Lektionsdatei automatisch geschlossen.

Der Suchbaum wird als AVL-Baum implementiert, da sich diese Art des binären Baums durch einen hohen Grad an Ausgeglichenheit auszeichnet. Das sollte günstige Auswirkungen auf die Suche haben. Ein einfacher binärer Baum könnte entarten und wäre immer höher als (oder gleichhoch wie) ein AVL-Baum. Der zusätzliche Strukturierungsaufwand fällt nur bei der Erstellung an, so daß zu einem späteren Zeitpunkt kein weiterer Overhead anfällt. Die Alternative der Nutzung eines B-Baumes scheidet auf Grund der relativ geringen zu verwaltenden Datenmengen wegen Unverhältnismäßigkeit aus.

Statt der Indizierung der Lektionsdateien in einer speziellen Verzeichnisdatei käme auch eine ähnliche Variante wie bei der Nutzerverwaltung in Betracht: Der Dateiname ist das Ergebnis einer Hash-Funktion, die als Argument die Lektionsnummer hätte, oder die Lektionsnummer selbst als Dateinamen. Beide Verfahren wären prinzipiell möglich und vielleicht sogar etwas schneller als die Baumsuche, jedoch erscheint es mir günstiger, "intelligente" Dateinamen zuzulassen und diese zu indizieren, da dadurch die Datenpflege erleichtert wird. ("War der Schreibfehler in 21.dat oder 22.dat?" ist für die meisten Menschen schwerer zu behalten als die Antwort auf die Frage "War der Schreibfehler in Begrüßungen.dat oder Landeskunde.dat?")

In den AVL-Suchbaum werden als Suchschlüssel die logischen Lektionsnummern und als Information der entsprechende Dateiname mit abgelegt. Da in einem solchen ausbalancierten Baum eine Suche maximal $\log_2(n)+1$ Suchschritte bei n Einträgen benötigt, ist die Suchzeit im Vergleich zu der Zeit des Datentransfers vom externen Massenspeicher vernachlässigbar gering.

2.4.3.3. Seiteninterpreter

Der Seiteninterpreter ist für die Aufbereitung und Aufteilung einer Lektion in einzelne Übungen zuständig. Dabei wird die von der Übungsverwaltung übergebene Datei nach bestimmten Kommandosequenzen durchsucht. Das Dateiformat der Übungsdateien wurde bewußt "menschenlesbar" gelassen. Die im Dateikopf enthaltenen Texte werden zur Weiterverwendung als Feedback an die I/O-Engine weitergegeben. Im folgenden werden die einzelnen Aufgaben auf den Aufgabentypus hin untersucht, die richtigen Antworten werden der Eingaben-Auswertung übergeben und der Aufgabentext und -typ an die I/O-Engine. Danach wird auf ein Navigationskommando der Eingabe-Auswertung gewartet, um mit der entsprechend nächsten Aufgabe fortzufahren. Nach Abarbeiten der letzten Aufgabe einer Lektion wird ein Ende-Signal an die Eingabe-Auswertung gesandt.
Die "menschenlesbare" Form der Übungsdateien ermöglicht eine leichtere Erstellung und Bearbeitung der Übungen, da nicht erst spezielle Editoren benötigt werden und auch schon vorhandene Texte leicht eingebaut werden können. Ein binäres, also nur maschinenlesbares Format hätte diesen Vorteil nicht, würde jedoch eine kompaktere Speicherung ermöglichen. Die für die Übungen anfallende Datenmenge ist jedoch nicht so immens, als daß eine komprimiertere Darstellung von Nöten wäre.

2.4.3.4. Eingaben-Auswertung

Die Eingaben-Auswertung ist das eigentliche Kernstück des Programms. Sie ist dafür zuständig, daß alle vom Nutzer eingegebenen Kommandos und Antworten auf die Fragen in Abhängigkeit vom Kontext richtig ausgewertet werden und zur entsprechenden Programmreaktion führen. Die Nutzer-Eingaben werden dazu von der I/O-Engine entgegengenommen und unverzüglich an die Eingaben-Auswertung weitergereicht. Dort werden dann z.B. die Antworten auf ihre Richtigkeit hin überprüft. Sollte eine Frage falsch beantwortet worden sein, so wird ein Kommando an die I/O-Engine zur Darstellung eines entsprechenden Hinweises gesendet und ein Eintrag im Fehlerlogbuch veranlaßt. Aber auch Navigationskommandos werden hier ausgewertet. So muß z.B. bei einem Wechsel der Lektion die Übungsverwaltung angewiesen werden, die entsprechende Datei zu suchen und nutzbar zu machen. Bei dem Beenden des Programms wird von der

Eingaben-Auswertung geprüft, ob noch ein Nutzer angemeldet ist und in diesem Fall die Nutzer-Verwaltung angewiesen, den Nutzer-Status zu aktualisieren, bevor das Programm beendet wird.

2.4.3.5. Fehlerlogbuch

Das Fehlerlogbuch ist die einfachste Einheit des Programms. Es ist ausschließlich dafür zuständig, die von einem Nutzer gemachten Fehler bei der Bearbeitung der Aufgaben zu verwalten und der Eingaben-Auswertung und Nutzer-Verwaltung zugänglich zu machen.

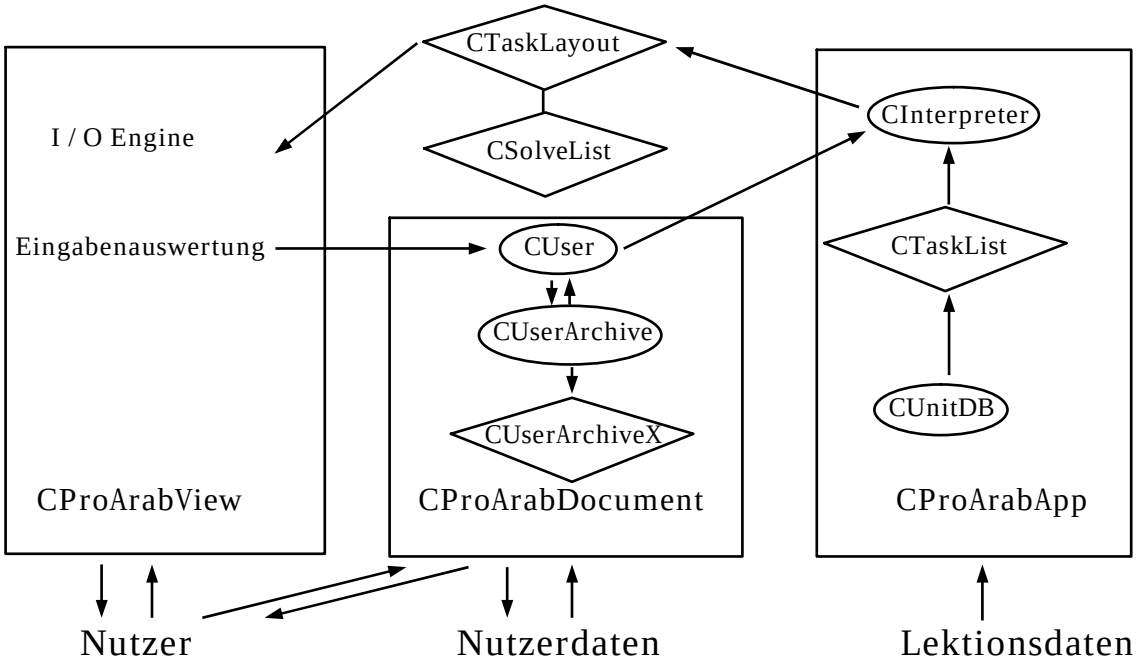
2.4.3.6. I/O-Engine

Die I/O-Engine ist die Schnittstelle zu dem Nutzer. Sie ist in 3 Zonen aufgeteilt: den Aufgabenbereich, den Statusbereich und den Navigationsblock.
Im Aufgabenbereich werden die aktuelle Aufgabe angezeigt und die Antworten des Nutzers entgegengenommen. Der Statusbereich informiert z.B. über die aktuelle Aufgabe. Der Navigationsbereich besteht aus mehreren Schaltelementen, mit denen Aufgaben übersprungen werden können oder zu vorherigen Aufgaben zurückgekehrt werden kann bzw. andere Änderungen des Übungsverlaufes vorgenommen werden können.
Alle Eingaben werden an die Eingaben-Auswertung weitergereicht und von dort werden Kommandos zu bestimmten Sachverhalten zur Visualisierung entgegengenommen. Die Texte der einzelnen Aufgaben erhält die I/O-Engine vom Seiteninterpreter in bereits aufbereiteter Form.
Da die Art des Programms Eingaben in arabischer Sprache voraussetzt und das vom System zur Verfügung gestellte Edit-Control keinen linksläufigen Text unterstützt, wird dazu ein eigenes Control via OLE eingebunden. Die Prozeduren zur Codepage-Umsetzung und der Zeichen-Logik zur Identifizierung der Position eines Buchstabens im Wort befinden sich in einer separaten DLL, da diese Routinen sowohl vom Programm direkt, als auch indirekt über das OLE-Control benötigt werden und so nicht doppelt implementiert werden müssen, was unter Umständen zu Inkonsistenzen führen würde.
Außerdem ist es durch die Ausgliederung dieses Codes leichter möglich, das Programm später an andere (Nutzer-)Sprachen anzupassen, da dann lediglich die DLL getauscht werden muß.

2.4.4. Identifizierung der Klassen

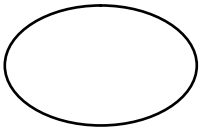
2.4.4.1. Allgemeines

Die identifizierten Module kommen Subsystemen gleich, die nun im einzelnen näher untersucht und detaillierter beschrieben werden müssen. Die Relation Modul = Subsystem stimmt in allen Fällen bis auf die I/O-Engine, die mit der Einheit des SpecialEdit getauften Arabisch-Eingabecontrols verschmolzen wird. Weiterhin wird die Nutzerverwaltung und das Fehlerlogbuch als eine Einheit angesehen.
Die Notation der Klassen erfolgt in Textform nach gültiger C++-Notation.
Da in MS Visual C++ das Document-View-Konzept verwendet wird, verdeutlicht die folgende Darstellung die Verteilung der Funktionen in diesem Konzept.

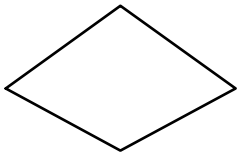


Legende:

Klassen des Document-View-Konzeptes



datenverarbeitende Klassen



Datentransport-Klassen

2.4.4.2. Nutzerverwaltung / Fehlerlogbuch

2.4.4.2.1. Klasse CUser - Eigenschaften eines Benutzers

Die Klasse CUser ist die zentrale Verwaltungsstelle für alle Eigenschaften des Nutzers. Zu einem bestimmten Zeitpunkt kann von Ihr maximal 1 Objekt existieren.

```
class CUser
{
        friend class CUserArchive;
        // attributes
        public:
                CString name;
                CTime date;
        private:
                CString password;
                CTagList errors;
                CTagList oks;
                CDoneList done;
                CTimeSpan lasttime;
                CTime starttime;
        // operations
        public:
                CUser(CString Name, CString pw);
                ~CUser();
                int TagError(UWORD unit, UWORD nr);
                int TagOK(UWORD unit, UWORD nr);
                int AddDone(UWORD unit);
                int RemDone(UWORD unit);
                void CheckComplete(UWORD unit, ULONG count);
                CTagNode *GetFirstErr();
                CTagNode *GetNextErr();
};
```

Beschreibung der Member-Variablen

Name der Variablen	*Beschreibung*	*Typ*
name	Enthält den Namen des Benutzers	CString; ein Zeichenkettenvariablentyp von der MFC-Bibliothek
password	Enthält das Paßwort, mit dem die persönlichen Daten in der Nutzerdatei verschlüsselt werden	Siehe oben
errors	Liste mit den fehlerhaft bearbeiteten Aufgaben des Nutzers	CTagList; eine Liste speziell für diese Anwendung
done	Liste mit allen vollständig bearbeiteten Lektionen	CDoneList

Name der Variablen	*Beschreibung*	*Typ*
oks	Liste der richtig bearbeiteten Aufgaben	CTagList
date	Datum der letzten Sitzung	CTime; MFC-Bibliothek
lasttime	Zeitspanne der Sitzungen bisher gesamt	CTimeSpan; MFC-Bibliothek
starttime	Datum und Zeitpunkt des heutigen Sitzungsbeginns	CTime; MFC-Bibliothek

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUser	Konstruktor der Klasse	Erhält den Namen und das Paßwort des neu anzulegenden Nutzers
TagError	Fügt der Liste der Fehler einen weiteren hinzu. Der Rückgabewert zeigt den Erfolg der Funktion an	Unit: Nummer der Lektion Nr: Nummer der Übung Beide Parameter sind vom Typ unsigned short int
TagOK	Entfernt Eintrag aus der error-Liste und fügt ihn der oks-liste hinzu	Unit: Nr. der Lektion Nr: Nr. der Übung
AddDone	Fügt der Liste eine weitere abgeschlossene Lektion hinzu. Rückgabewert = Erfolgsmeldung	Unit: Nummer der Lektion unsigned short int
RemDone	Entfernt Eintrag aus der Done-Liste	Unit: Nummer der Lektion unsigned short int
CheckComplete	Testet, ob alle Übungen einer Lektion in der Oks-Liste sind	Unit: Nummer der Lektion Count: Anzahl der Einträge

2.4.4.2.2. Die Klasse CUserArchive - die Schnittstelle zur Permanentspeicherung

Die Klasse CUserArchive stellt alle Funktionen zum regulären Laden und Speichern der Benutzerinformationen und zum Im- und Exportieren dieser bereit. Zu diesem Zweck ist sie auch als Freund der Klasse CUser deklariert, d.h. sie kann auch auf alle nicht-public deklarierten Member zugreifen.

```
class CUserArchive
{     // attributes
      private:
            CFile file;
            CString fname;
      // operations
      public:
            CUserArchive();
            CUserArchive(CUser *user,int mode=out);
            CUserArchive(CString fname,int mode=out);
            ~CUserArchive();

            int Open(CUser *user, int mode=out);
            int Open(CString fname,int mode=out);
            int Close();

            CUserArchive operator>>(CUser &);
            CUserArchive operator<<(CUser &);
      private:
            CString GenerateFName(CUser &);
};
```

Beschreibung der Member-Variablen

Name der Variablen	*Beschreibung*	*Typ*
open_mode	Richtungsangabe des Datentransfers (Laden oder Speichern)	Aufzählung [in,out]
file	Die zu dem entsprechenden Nutzer korrespondierende Datei als Handle	CFile, MFC-Bibliothek, Filehandle-Klasse
filename	Bei regulärem Laden oder Speichern wird ein eindeutiger Dateiname aus dem Nutzernamen generiert, der in dieser Variablen abgelegt wird	CString, MFC-Bibliothek, Zeichenkettentyp

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUserArchive	Standard-Konstruktor, generiert ein Objekt dieser Klasse, das mit keinem Nutzer assoziiert ist	N/A
CUserArchive	Konstruktor, der das Objekt gleich mit einem bestimmten Nutzer verbindet; kann eine Exception bei auftretenden Fehlern auswerfen	user: Zeiger auf assoziiertes Nutzer-Objekt mode: Transferrichtung, etweder in oder out
CUserArchive	Konstruktor zum Im- und Export von Nutzerdaten; kann eine Exception bei auftretenden Fehlern auswerfen	file: Dateiname als CString; mode: Transferrichtung
Open	Verbindet einen Nutzer mit seinem Datensatz	user: Zeiger auf assoziiertes Nutzer-Objekt mode: Transferrichtung, etweder in oder out
Open	Verbindet einen Nutzer mit einem Datensatz über Im- oder Export	file: Dateiname als CString; mode: Transferrichtung
Close	Trennt die Verbindung zwischen Datensatz und Nutzerobjekt	N/A
operator>>	Importoperator, lädt die Datensätze in das Nutzerobjekt	user: Referenz auf zu füllendes Nutzerobjekt
operator<<	Exportoperator,speichert die Daten eines Nutzerobjekts in Datei	user: Referenz auf zu exportierendes Nutzerobjekt
GenerateFName	Generiert aus dem Namen des Nutzers den Standard-Dateinamen (=Rückgabewert)	user: Referenz auf Nutzerobjekt

2.4.4.2.3.Die Klasse CUserArchiveX - die Exception der Nutzerverwaltung

Bei der Speicherung und beim Laden eines Datensatzes können nicht nur Dateifehler auftreten sondern auch andere Fehler wie Doppelbelegung von Namen etc. In diesem Fall kann der genaue Grund des Scheiterns einer Operation über dieses ausgeworfene Objekt analysiert werden.

```
class CUserArchiveX
{
        // attributes
        public:
                CFileException  uax_FileException;
                int uax_MainCause;
        // operations
        public:
                CUserArchiveX(CFileException *x=NULL,int y=0);
                CUserArchiveX(const CUserArchiveX &);
};
```

Beschreibung der Member-Variablen

Name der Variablen	*Beschreibung*	*Typ*
FileException	Im Falle eines Dateifehlers enthält diese Struktur die näheren Angaben	CFileException, MFC-Bibliothek, wird von CFile ausgeworfen
MainCause	Die generelle Fehlerkategorie	integer

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUserArchiveX	Konstruktor	CFileException * Zeiger auf die File-Exception (von MFC); Int Fehlertyp für MainCause
CUserArchiveX	Kopierkonstruktor	const CUserArchiveX & Referenz des zu kopierenden Materials

2.4.4.3. Übungsverwaltung

2.4.4.3.1. Die Klasse CUnitInfo - Informationen zu den Lektionen

Die Klasse CUnitInfo ist ein Container, der die Zuordnung zwischen Dateinamen und logischer Dateinummer enthält.

```
class CUnitInfo
{
        // attributes
        public:
                CString filename;
                CString title;
                UWORD nr;
        // operations
        public:
                CUnitInfo();
                CUnitInfo(const CString &,const CString &,
                                                        const UWORD);
                CUnitInfo(const CUnitInfo &);
};
```

Beschreibung der Member-Variablen

Name der Variablen	*Beschreibung*	*Typ*
filename	physischer Dateiname der Lektionsdatei	CString; MFC-Bibliothek, Zeichenkettenvariable
title	Titel der Übungseinheit	CString, MFC-Bibliothek, Zeichenkettenvariable
nr	logische Nummer der Lektion	UWORD= unsigned short int

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUnitInfo	Standard-Konstruktor	N/A
CUnitInfo	Konstruktor, der die Membervariablen mit entsprechenden Werten initialisiert	1.Parameter: Dateiname als CString-Referenz 2.Parameter Titel als CString-Referenz 3.Parameter: logische Lektionsnummer
CUnitInfo	Kopier-Konstruktor	zu kopierende Struktur

2.4.4.3.2. Die Klasse CUnitNode - ein Baumknoten im Lektionssuchbaum

Basisklasse: CAvlNode

Die Klasse CUnitNode ist eine spezielle Ausformung eines AVL-Baum-Knotens, der als Nutzlast CUnitInfo's trägt.

```
class CUnitNode:public CAvlNode
{
        friend CUnitTree;
        // attributes
        public:
                CUnitInfo       content;
        // operations
        public:
                CUnitNode();
                CUnitNode(const CUnitNode &);
                CUnitNode(const CUnitInfo &);
                CUnitNode(const CString &,const CString &,
                                                        const UWORD);
                virtual void *put(void *);
                virtual CBinNode *Duplicate();
        protected:
                virtual int Cmp(const CBinNode &);
};
```

Beschreibung der Member-Variablen

CUnitInfo content; enthält alle Daten über eine Lektion

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUnitNode	Standard-Konstruktor	N/A
CUnitNode	initialisiert vorgeg. Nutzlast	const CUnitInfo & Info-Container
CUnitNode	Kopierkonstruktor	const CUnitNode & Kopiervorlage
CUnitNode	Konstruktor	1. Parameter: Dateiname 2. Parameter: Lektionstitel 3. Parameter: logische Lektionsnummer
Cmp	Vergleichsoperation	Vergleichsmaterial

2.4.4.3.3. Die Klasse CUnitTree - Der Suchbaumkopf

Basisklasse: CAvlTree

Die Klasse CUnitTree stellt die spezielle Ausformung des AVL-Suchbaumes für die Lektionsdaten dar. Sie verwendet Knoten vom Typ CUnitNode.

```
class CUnitTree:public CAvlTree
{
         // operations
        public:
                CUnitTree();
                CUnitNode *Find(UWORD);
                CUnitNode *Find(const CUnitNode &p);
};
```

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUnitTree	Standard-Konstruktor, legt einen leeren Baum an	N/A
Find	Suche nach einem bestimmten Eintrag, Rückgabewert=Zeiger auf Knoten oder NULL	Suchschlüssel; entweder Lektionsnummer oder äquivalenter Knoten

2.4.4.3.4. Die Klasse CUnitDB - Die "Datenbank" der Lektionen

Die Klasse CUnitDB stellt die Schnittstelle des Programms zu den Lektionsdaten dar. Im gesamten System kann nur ein Objekt dieser Klasse existieren.

```
class CUnitDB
{
        // attributes
        private:
                CUnitTree    units;
        // operations
        public:
                CUnitDB();
                CTaskList *Get(UWORD);
                UWORD GetNr(const CString &);
        private:
                void MakeIDX();
};
```

Beschreibung der Member-Variablen

Name: units

Beschreibung: Der Suchbaum mit allen verfügbaren Lektionen

Typ: CUnitTree; AVL-Baum mit CUnitInfo als Nutzlast

Beschreibung der Member-Funktionen

Name der Funktion	*Beschreibung*	*Parameter*
CUnitDB	Standard-Konstruktor, baut den gesamten Suchbaum auf	N/A
Get	Gibt eine Liste mit den Übungen der gesuchten Lektion zurück	Nummer der Lektion, unsigned short int
GetNr	Sucht die Lektionsnummer zum Titel der Lektion	Lektionstitel
MakeIDX	Den Lektionsindex generieren	N/A

2.5. Dateiformate

2.5.1. Format der Lektionsdateien

2.5.1.1. Allgemeines

Die Lektionsdateien enthalten alle Aufgabenstellungen, deren Typisierung und deren Lösungen. Sie sind reine ASCII-Dateien unter Verwendung der arabischen Windows - Codepage. Zur Steuerung des Programmablaufs sind Kommando-Sequenzen in den Text eingefügt. Als sequenzeinleitendes Zeichen (CSI) wurde das Zeichen "<" verwendet, zum Sequenzabschluß das Zeichen ">". Auf das CSI folgt das Kommando bzw. ein "/" und danach das Kommando zum Einleiten bzw. Beenden einer Sequenz.

2.5.1.2. Strukturelemente

2.5.1.2.1. <lesson #>, </lesson>

Öffnet bzw. schließt den Dateikopf mit globalen Informationen. Dabei wird das # durch eine natürliche Zahl zwischen 1 und 65535 ersetzt, die eine eindeutige Zuordnung der Lektionen ermöglichen soll. Ein <lesson #>-Block muß in jeder Lektionsdatei genau einmal enthalten sein und muß sich auf globaler Ebene befinden.
Die Zahl bestimmt sich aus der Nummer der Lektion und der Position der Übungseinheit innerhalb dieser Lektion über:
Nummer=(Lektionsnummer)*100+(Position in der Lektion)

2.5.1.2.2. <postext>, </postext>

Pro Lektionsdatei existiert genau ein solcher Block auf der <lesson #>-Ebene. Durch Zeilenumbrüche getrennt erfolgt hier eine Aufzählung der Texte, die nach richtiger Bearbeitung einer Aufgabe ausgegeben werden sollen. Bei Angabe mehrerer solcher Texte (durch CR getrennt) erfolgt die Auswahl im Einzelfall zufällig.

2.5.1.2.3. <negtext>, </negtext>

Wie <postext>, nur zur Angabe von Texten bei fehlerhafter Bearbeitung einer Aufgabe.

2.5.1.2.4. <title>, </title>

Pro Lektionsdatei existiert genau ein solcher Block auf der <lesson#>-Ebene. Er enthält die Bezeichnung der Übungseinheit.

2.5.1.2.5. <head>, </head>

Der Block enthält die Überschrift für die Übungen, wie zum Beispiel Übungsanweisungen. Diese bleiben bis zum nächsten Erscheinen eines solchen Blocks gültig.

2.5.1.2.6. <task>,</task>

Auf globaler Ebene existiert mindestens ein solcher Block, der eine Einzelaufgabenstellung beinhaltet. Nach dem öffnenden Tag muß eines der im folgenden beschriebenen Tags folgen, welches den Aufgabentypus näher erklärt.

<mc>

Existiert auf der <task>-Ebene. Aufgabentypus Multiple-Choice.
Benötigt kein schließendes Tag.

<lt>

Existiert auf der <task>-Ebene. Aufgabentypus Lückentext.
Benötigt kein schließendes Tag

<vk>

Existiert auf der <task>-Ebene. Aufgabentypus Vokalisierungsübung.
Benötigt kein schließendes Tag.

<lx>

Existiert auf der <task>-Ebene. Aufgabentypus Lexik.
Benötigt kein schließendes Tag.

<tb>

Existiert auf der <task>-Ebene. Aufgabentypus Schautafel (interaktiv).
Benötigt kein schließendes Tag.

<ex>

Existiert auf der <task>-Ebene. Definiert ein Beispiel.
Benötigt kein schließendes Tag.
Die Typen <mc>, <lt>, <vk>, <lx>, <tb> und <ex> schließen einander aus.

2.5.1.2.7. <solve>, </solve>

Innerhalb dieses Blocks werden die Lösungen der Aufgaben angegeben. Existiert auf <Task>-Ebene.

2.5.1.2.8. <arab>,</arab>

Innerhalb dieses Blockes angegebener Text ist als arabischer Text zu interpretieren. Blöcke dieser Art können auf allen Ebenen außer der globalen vorkommen.

2.5.1.2.9. <example>, </example>

Dies ist eine Alternative zu der Angabe <task> <ex> ... </task>.

2.5.1.2.10. <choice>,</choice>

Existiert nur in Multiple-Choice-Aufgaben und enthält die Auswahlmöglichkeiten.

2.5.2. Format der Nutzerdateien

Die Nutzerdateien enthalten Informationen über den Namen des Nutzers, die von ihm bereits abgeschlossenen Lektionen und fehlerhaft bearbeitete Aufgaben.

Die Daten werden in Form von Blöcken binär abgelegt.

Jeder Block hat den gleichen Grundaufbau:

4 Byte	ID
1 DWord	Blocklänge in Byte
...	Daten

Blocklänge Null bedeutet File-Ende.

ID	*Inhalt*
VERS	Version (Ablage in 1 DWord)
NAME	Nutzer-Name
PDAT	verschlüsselte Daten über die Aufgabenabarbeitung
DATE	Datum der letzten Sitzung

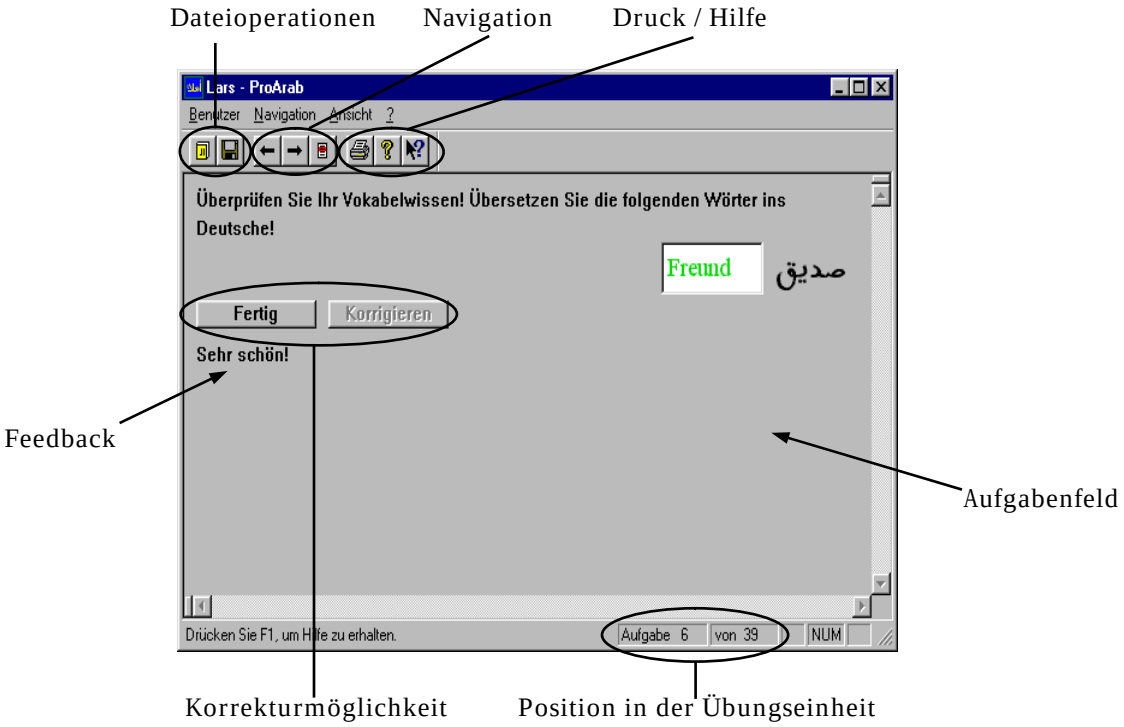
Die verschlüsselten Daten ihrerseit haben wiederum einen Blockaufbau. Jedoch ist das erste DWord zur Identifizierung des Paßwortes reserviert. Danach beginnen wieder Blöcke obigen Aufbaus:

ID	*Inhalt*
DONE	vollständig bearbeitete Lektionen; Nr. als Word abgelegt
ERRA	Fehlerhaft bearbeitete Aufgaben; Ablage als DWord: UPPER Word: Lektionsnummer LOWER Word: Nr. der Aufgabe
INCO	Richtig bearbeitete Aufgaben von unvollständig bearbeiteten Lektionen, Aufbau wie ERRA
TIME	Dauer der letzten Sitzungen gesamt

3. Programm

3.1. Oberfläche

Die Oberfläche des Programms soll die Aufmerksamkeit des Nutzers auf die Aufgaben lenken und ihm einfache Möglichkeiten zur Navigation innerhalb der Lektionen geben. Weiterhin soll ein Feedback zu jeder Aufgabe die Richtigkeit bestätigen bzw. darauf hinweisen, daß ein Fehler gemacht wurde und die Möglichkeit der automatischen Korrektur der Fehler geben.
Um diesen Punkten gerecht zu werden, wurde ein einfaches Layout des Programmfensters gewählt:

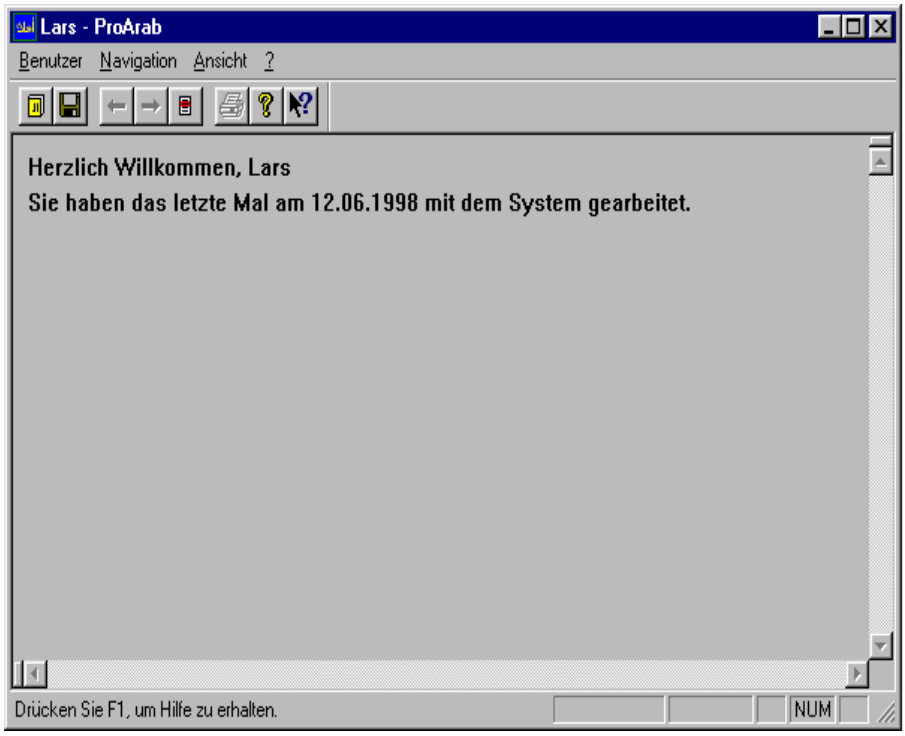


Das einfache Layout der Programm-Oberfläche wurde auch deshalb gewählt, um einigen allgemeinen Forderungen zu ergonomischen Aspekten bei der Softwareentwicklung gerecht zu werden.
In [Bal 96] werden einige Grundregeln beschrieben, die hier umgesetzt wurden. Eine klare Aufteilung des Fensters in verschiedene Aufgabenbereiche fördert die Verständlichkeit des Programms, da die Benutzer bestimmte Funktionalitäten voneinander abgegrenzt vorfinden und Zusammenhänge so leichter erkennen können. Weiterhin sollten die für eine bestimmte Plattform entwickelten Anwendung möglichst ein ähnliches

Aussehen besitzen, damit die Eingewöhnungsphase in neue Applikationen verkürzt wird und Bedienungsfehler vermindert werden ("Einmal erlernte und durch das System geprägte Aufgabenlösungsstrategien müssen ohne größere Umstellungsprozesse auf verschiedene Aufgabentypen übertragen werden können. Durch eine möglichst einheitliche Benutzungsoberfläche über verschiedene Anwendungen hinweg kann dies erreicht werden." [Bal 96], S.460f). Durch die Entwicklung des Programms mit Visual C++ und der konsequenten Nutzung der von diesem System gegebenen Hilfen zur Generierung einer Oberfläche wurde ein weitestgehend style-guide-konformes Programm mit relativ geringem Aufwand für die Darstellung möglich. Die Benutzer des Programms finden somit Funktionen, die ihnen von anderen Applikationen her bekannt sind, an den vertrauten Positionen. Durch Anpassen der speziellen Funktionen an das "look and feel" von Windows 95-Programmen können auch diese Funktionen in ihrem Inhalt leicht erschlossen werden.
Weiterhin wurden zur Erleichterung der Arbeit mit dem Programm die wichtigsten Navigationsfunktionen mit Tastaturkürzeln versehen, so daß diese schneller ausgeführt werden können, was zu einer Erhöhung der Bedienungsfreundlichkeit des Programms führt.

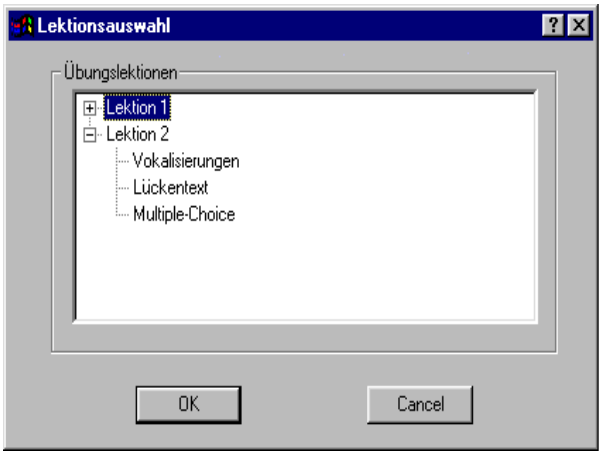
3.2. Beispieldurchlauf

Nach dem Start des Programms wird der Benutzer aufgefordert, sich im Programm anzumelden. Dadurch wird ermöglicht, daß alle Aktionen, die er in den Übungen durchführt, erfaßt werden können und bei einem späteren Start des Programms wieder restauriert werden und zu entsprechenden Reaktionen führen können. Das Paßwort dient gleichzeitig zwei Zwecken. Zum einen dient es der Authentifizierung des Benutzers bei einer späteren erneuten Anmeldung, so daß die eindeutige Zuordnung eines Namens zu einer bestimmten Statusdatei gesichert ist. Zum anderen werden die Daten über die Richtigkeit der bereits bearbeiteten Übungen mit diesem Paßwort verschlüsselt, so daß niemand anderes Zugriff darauf hat.



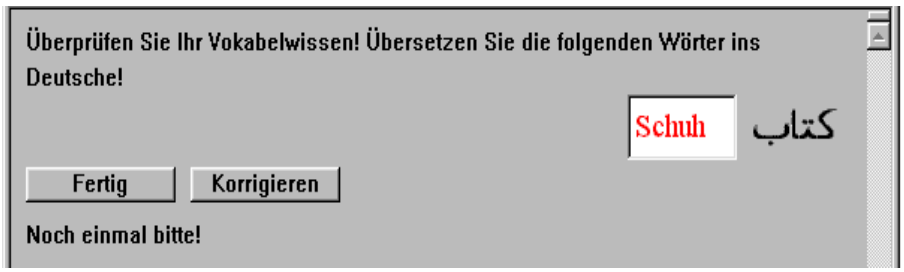
Nach erfolgreichem Anmelden steht dem Benutzer die gesamte Funktionspalette des Programms zur Verfügung. Wurden bei vorherigen

Sitzungen Aufgaben fehlerhaft bearbeitet, erhält er die Möglichkeit, diese zu wiederholen und seine Kenntnisse zu verbessern. Die einzelnen Aufgaben werden dabei unabhängig von der Lektionseinteilung anhand der gespeicherten Informationen geladen und angezeigt. Der Wiederholungslauf stellt einen strikt linearen Durchlauf durch die zu wiederholenden Aufgaben zur Verfügung. Damit soll verhindert werden, daß die Resultate durch den Benutzer selbst "geschönt" werden können, indem nach erfolgter Korrektur zu der Aufgabe zurückgekehrt wird und die nun bekannte, aber nur im Kurzzeitgedächtnis gespeicherte Lösung eingetragen wird.
Der Wiederholungslauf kann jederzeit durch die Auswahl einer Lektion abgebrochen werden und so zu dem normalen Arbeitspensum übergegangen werden.



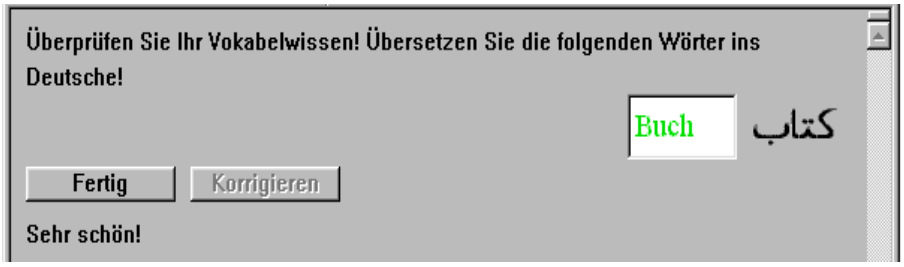
Die einzelnen Lektionen sind in verschiendene Übungseinheiten unterteilt, um einerseits die Komplexität der Lektionen überschaubarer zu machen und zum anderen dem Benutzer die Möglichkeit zu geben, präziser den gewünschten Komplex der Übungen auszuwählen. Dadurch wird die allgemeine Forderung nach stärkerer Selbststeuerung des Lernenden im Lernprozeß erfüllt. Alternativ können zum Beispiel schwächere Lernende durch die Numerierung der Lektionen einen linearen Kurs durch das Angebot nehmen.
Nach der Auswahl der Übungseinheit wird die erste Aufgabe geladen und deren Bearbeitung erwartet. Unter Umständen wird hier auch ein erklärendes Beispiel gezeigt, daß die Erfüllung der Aufgabenstellung erläutert. Das ist vor allem bei dem Gebrauch von speziellen Termini in der Aufgabe für ungeübte Anwender wichtig.
In der Statuszeile wird dann auch angezeigt, die wievielte Aufgabe des aktuellen Übungskomplexes gerade angezeigt wird und wieviele Aufgaben der Komplex insgesamt umfaßt.

Die Aufgabe wird durch Eingabe der Lösung in das entsprechende Feld bearbeitet. Nach erfolgtem Eintrag kann eine Überprüfung der Lösung durch Betätigen des "Fertig"-Buttons ausgelöst werden.

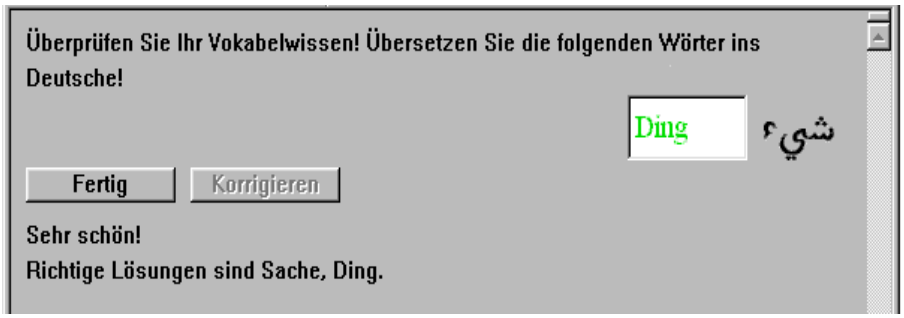


Wurde die Aufgabe fehlerhaft bearbeitet, wird dies durch die rote Färbung des Eintrages verdeutlicht und zusätzlich ein entsprechender Hinweis eingeblendet. Außerdem steht ab diesem Moment die Möglichkeit, den Eintrag korrigieren zu lassen, zur Verfügung. Das falsche Lösen der Aufgabe wird natürlich sofort intern vermerkt, damit eine entsprechende Liste der falsch bearbeiteten Aufgaben (mit den richtigen Lösungen) bei Bedarf gedruckt werden kann und zu Beginn der nächsten Übungssitzung diese Aufgabe zur Wiederholung angeboten werden kann.
Wird die Aufgabe richtig gelöst, dann erfolgt eine grüne Einfärbung des Eintrags und ein entsprechender Feedback wird ausgegeben. Intern wird die Aufgabe als gelöst markiert und der "Korrigieren"-Button bleibt

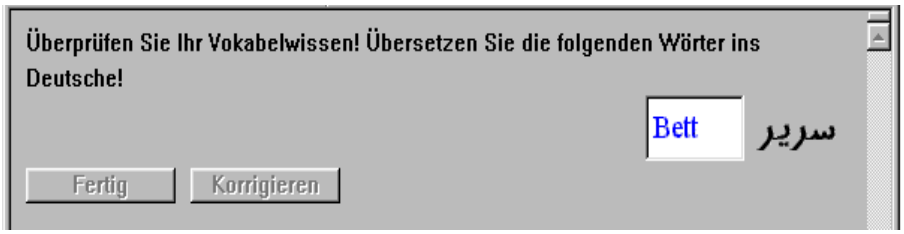
gesperrt.



Der Benutzer kann nun zur nächsten Aufgabe übergehen.
Um die Entstehung von "Vokabelgleichungen" zu vermeiden, wird bei Aufgaben, die mehrere Lösungsmöglichkeiten bieten, ein Hinweis darauf gegeben:



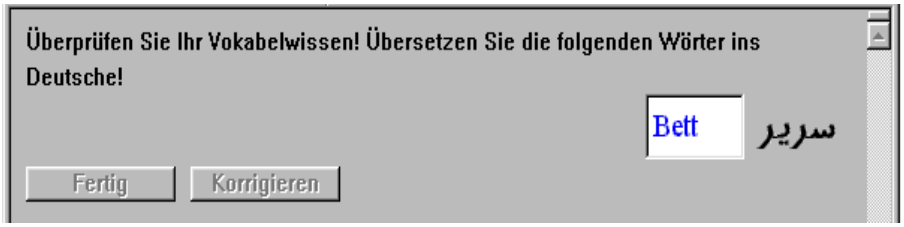
Wenn die Korrektur-Funktion in Anspruch genommen wird, dann wird im Eingabefeld die Lösung in blauer Schrift dargestellt und der "Fertig"-Button gesperrt.



3.3. Umsetzung der Übungstypen

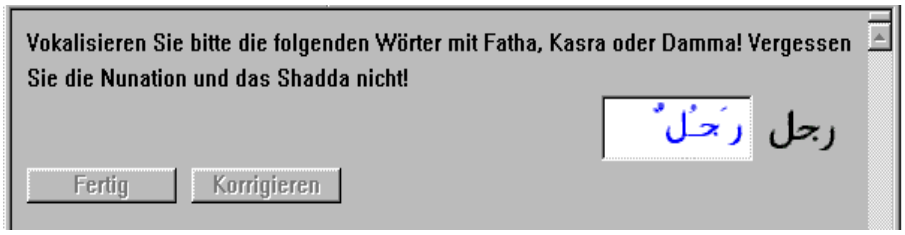
3.3.1. Lexik

Der Übungstyp Lexik läßt sich zur direkten Vokabelkontrolle von sowohl deutschen als auch arabischen Vokabeln einsetzen und läßt mehrere Lösungsmöglichkeiten zu.



3.3.2. Vokalisierungen

Die Vokalisierungen stellen ein besonderes Problem des Arabischen dar. Sie sind nur für arabischen Text einsetzbar. Es sind Mehrfachlösungen zugelassen. Da jedoch oft mit einer Änderung der Vokalisierung ein Wandel der Bedeutung einhergeht, sollte davon möglichst kein Gebrauch gemacht werden, um den Lernenden nicht zu falschen Verknüpfungen von Vokalisierung und Bedeutung zu verleiten.

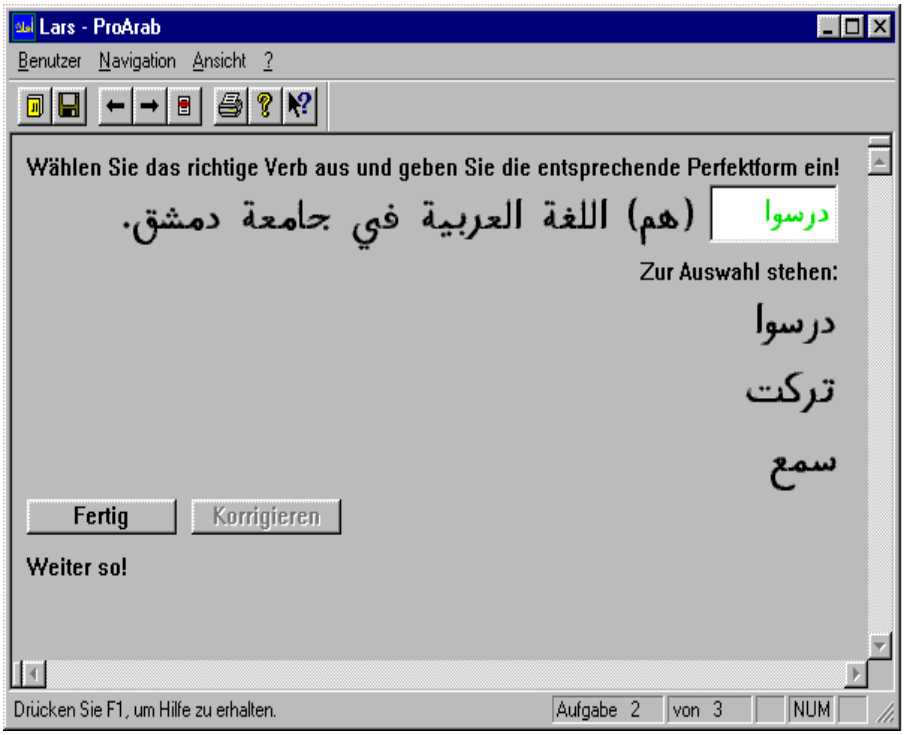


3.3.3. Datumsangaben

Der Übungstyp Datumsangaben brauchte nicht mehr explizit implementiert werden, da er nur eine Untermenge des Übungstypus Lexik darstellt und die ehemals beabsichtigte feinere Auswertung des Eintrags als unnötig erschien.

3.3.4. Multiple-Choice

Bei den Multiple-Choice-Aufgaben bestanden die Alternativen in der Umsetzung, entweder über eine Auswahlbox die richtige Lösung einfach anklicken zu lassen oder die Lösung noch einmal neu eingeben zu lassen. Die zweite Möglichkeit ist nicht nur technisch leichter umzusetzen, sie ist auch der Einprägeleistung des Lernenden förderlich, da er sich einer produktiveren Aufgabenstellung gegenüber sieht als das bei einer einfachen Auswahl der Fall gewesen wäre und die Gedächtnisaktivität so höher ist.

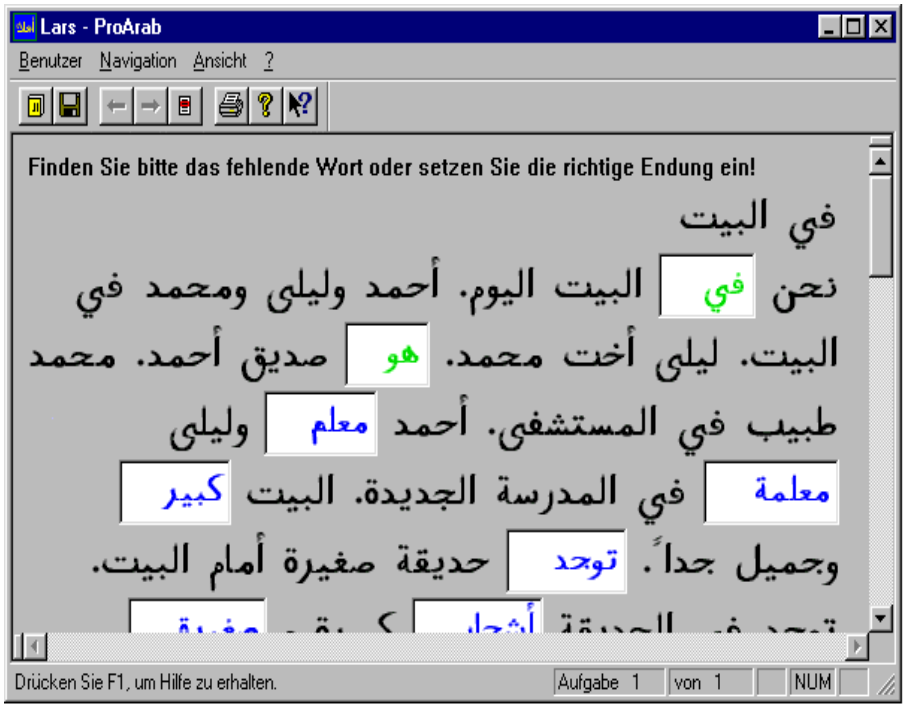


3.3.5. Tabellen

Die Implementierung des Meta-Typs Tabellen wurde nicht umgesetzt. Die Gründe dafür sind in einem gewissen Zeitmangel bei der Implementierung zu suchen. Die Hauptanwendung der Tabellen, nämlich die Darstellung von kurzen Dialogen, läßt sich jedoch mit dem Übungstyp Lückentext verwirklichen und die Konjugation der Verben über die Vokalisierungen überprüfen. In einer Weiterführung dieses Projekts sollte dieser Übungstyp jedoch nicht mehr fehlen.

3.3.6. Lückentext

Die komplexeste Übungsstruktur stellt der Lückentext dar. Durch ihm ist der Lernende gezwungen, mannigfaltige Aktionen durchzuführen, damit er den Sinn des Textes versteht und die Lücken sinnvoll und in der richtigen grammatischen Form füllen kann.



Bei der Korrektur eines Lückentextes bleiben natürlich die richtigen Eingaben erhalten, nur die falschen oder leeren Einträge werden gefüllt. Dies macht eine differenziertere Fehlereinschätzung durch den Lernenden möglich. In der Wiederholung muß jedoch der gesamte Text neu ausgefüllt werden.

4. Abkürzungsverzeichnis

CALL	Computer Assisted Language Learning, computerunterstütztes Fremdsprachenlernen
CBT	Computer Based Training, computerbasierte Übung / computerbasierter Unterricht
CR	Carridge Return, Zeilenende-Zeichen
CSI	Command Sequence Introducer, Kommandosequenz einleitendes Zeichen
DLL	Dynamic Link Library
DWord	Double-Word, Long-Integer-Zahl, 32-bit
ECB	Electronic Code Book, Verschlüsselungsmodus, bei dem die einzelnen Blöcke nacheinander mittels Algorithmus unabhängig von einander chiffriert werden.
MFC	Microsoft Foundation Classes; eine von Microsoft zur Verfügung gestellte Klassenbibliothek, die hauptsächlich die Oberflächen- Generierung unterstützen soll
OLE	Object Linking and Embedding
OOA	object oriented analysis, objektorientierte Analyse
OOD	object oriented design, objektorientorientiertes Design / Entwurf
OOP	object oriented programming, objektorientierte Programmierung
RFC	request for comments, von der Network Working Group initiiertes System der Systematisierung online vorliegender Fachtexte.

5. Kurzzusammenfassung

Die Computertechnik hält in allen Gebieten des Lebens Einzug. Auch bei der Aus- und Weiterbildung werden die Stimmen laut, die eine Integration von computergestützten Elementen im Unterricht fordern.
Die vorliegende Arbeit hat es sich zur Aufgabe gemacht, die Möglichkeiten eines Programms auszuloten, das speziell für die Übungen in des Sprachunterrichts Arabisch an der Universität Leipzig entstehen sollte. Dabei wurden sowohl die pädagogischen Aspekte des Computereinsatzes als auch die aktuellen technischen Möglichkeiten berücksichtigt.
Als Ergebnis dieser Betrachtungen wurde eine einfache Anwendung generiert, die es erlaubt, Übungseinheiten auf den Computer zu verlagern. Neues Wissen soll damit nicht vermittelt werden. Damit sollten viele Probleme, die aus der Zeit der Programmierten Instruktion her bekannt sind, umgangen worden sein.
Das Programm bietet 4 universell einsetzbare Übungstypen an, mit denen das Wissen der arabischen Schriftsprache gefestigt werden können. Das Programm wurde so konzipiert, daß weitere Übungstypen zu einem späteren Zeitpunkt relativ leicht implementiert werden können. Das wurde unter anderem durch die strikte Einhaltung der objektorientierten Programmierparadigmen erreicht.
Berücksichtigt wurde auch, daß sich die Übungseinheiten nach kurzer Einarbeitung in die Beschreibungssprache leicht erstellen lassen, so daß die Zahl der Lektionen erweitert werden kann.

6. Bibliographie

[Ada 97] Adams, C.:"The CAST-128 Encryption Algorithm", RFC 2144

[Bal 82] Balzert, Helmut:"Die Entwicklung von Software-Systemen", Mannheim/Leipzig/Wien/Zürich, BI Wissenschaftsverlag, 1982

[Bal 96] Balzert, Helmut:"Lehrbuch der Software-Technik -- Softwareentwicklung", Heidelberg, Berlin, Oxford, Spektrum Akademischer Verlag GmbH, 1996

[Bal 98] Balzert, Helmut:"Lehrbuch der Software-Technik -- Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung", Heidelberg, Berlin, Spektrum Akademischer Verlag GmbH, 1998

[Bau 95] Bausch / Christ / Krumm (Hrsg.): "Handbuch Fremdsprachenunterricht",Tübingen, Basel, UTB / Francke Verlag, 1995

[Bei 95] Beims, Hans Dieter: "Praktisches Softwareengeneering: Vorgehen, Methoden, Werkzeuge", München, Wien, Hanser-Verlag, 1995

[Bol 94] Boldt, Olaf: "Graphematische und morphologische Probleme beim Scannen arabischer Texte", Diplomarbeit, Orientalisches Institut der Universität Leipzig, 1994

[Boo 94] Booch, Grady:"Objektorientierte Analyse und Design",Bonn, Paris, Reading, Mass. et al., Addison-Wesley (Deutschland) GmbH, 1994

[Bur 95] Burger, Günter (Hrsg.): "Fremdsprachenunterricht in der Erwachsenenbildung", Max Hueber Verlag, 1995

[Con 88] Conrad, Rudi [Hrsg.]: "Lexikon sprachwissenschaftlicher Termini", Leipzig, Bibliographisches Institut, 1988

[Ehn 86] Ehnert, Rolf/ Piepho, Hans-Eberhard (Hrsg.): "Fremsprachen lernen mit Medien", Max Hueber Verlag, München, 1986

[Gur 96] Gurewich/Gurewich: "Visual C++ in 21 Tagen", Haar bei München, Markt und Technik Verlag, 1996

[Iss 97] Issing; Klimsa (Hrsg.):"Lernen mit Multimedia", Beltz/Psychologie Verlags-Union, Weinheim, 1997

[Lip 95] Lippman, Stanley B.:"C++ Einführung und Leitfaden", Bonn;München;Reading, Mass. et al., Addison-Wesley, 1995

[Mor 96] Morgenstern, Ralf: "Visual C++ 4.x", Bonn, Reading et. al., Addison-Wesley, 1996

[Mut 81] Mutlak, Ingelore:"Die Arbeit am Wortschatz im Arabischunterricht unter linguistischen und fremdsprachendidaktischen Aspekten", Dissertation zur Promotion A, Sektion Afrika-Nahostwissenschaften, Universität Leipzig, 1981

[Pag 94] Pagel, Bernd-Uwe/Six, Hans-Werner: "Software-Engeneering --Die Phasen der Softwareentwickung", Bonn, Paris et al., Addison-Wesley (Deuschland) GmbH, 1994

[Pom 93] Pomberger/Blaschek:"Software Engeneering, Prototyping und objektorientierte Software-Entwicklung", München, Wien, Hanser-Verlag, 1993

[Raa 88] Raasch, Albert/Bludau, Michael/Zapp, Franz Joseph (Hrsg.): "Aspekte des Lernens und Lehrens von Fremdsprachen", Verlag Moritz Diesterweg, 1988

[Ram 89] Rampillion, Ute: "Lerntechniken im Fremdsprachenunterricht", Max Hueber Verlag, Ismaning, 1989

[Rüs 88] Rüschoff, Bernd: "Fremdsprachenunterricht mit computergestützten Materialien: didaktische Überlegungen und Beispiele", München, Max Hueber Verlag, 1988

[Sch 71] Dr. Scharf, Arnold: "Zur Arbeit mit kombinatorischen Übersichten bei der Festigung und Aktivierung von Kenntnissen", in Fremdsprachenunterricht 12/1971, Berlin, Verlag Volk und Wissen, 1971

[Sch 92] Schweighofer, Karin: " CBT - Computer Based Training: Interaktives Lernen mit dem Computer aus pädagogischer Sicht", Universitätsverlag Rudolf Trauner, Linz, 1992

[Sch 96] Schulmeister, R.:" Grundlagen hypermedialer Lernsysteme",Bonn et al., Addison-Wesley Verlag, 1996

[Sei 93] Seidel, Christoph [Hrsg.]: "Computer Based Training", Göttingen, Stuttgart, Verlag für Angewandte Psychologie, 1993

[Tha 93] Thaller, Georg Erwin: "Qualitätsoptimierung der Software-Entwicklung", Braunschweig, Vieweg Verlag, 1993

[Tha 95] Thaller, Georg Erwin:"Software-Dokumente -- Funktion, Planung, Erstellung", Hannover, Verlag Heinz Heise Gmbh & Co KG, 1995

[Uth 78] Utheß, Herbert/Utheß, Sabine: "Der Fremdsprachenaneignungsprozeß als Einheit von Erfassen, Einprägen, Einüben und Anwenden", in: Fremdsprachenunterricht Heft 7-8/1978, Berlin, Volk und Wissen Verlag, 1978

[Wil 98] Williams, Al: "MFC Black Book", Albany, NY; Belmont, CA; Bonn et al., Coriolis Group Books, 1998

[You 93] Yourdon, Edward: "Die westliche Programmierkunst am Scheideweg - die Schlüsseltechniken der Softwareentwicklung für das 21. Jahrhundert", Prentice Hall/ Hanser, München, Wien, London, 1993