Erfassung von Internetfragebögen

Wolf F. Lesener
Humboldt-Universität zu Berlin
Rechenzentrum
10099 Berlin
wflesener@rz.hu-berlin.de

Zusammenfassung

Umfragen unter Nutzung des Internets sind ein sehr beliebtes Instrument, um an Universitäten Datenerhebungen im Rahmen von Diplomarbeiten oder Dissertationen durchzuführen. Dabei sind die Interviewer nicht nur Sozialwissenschaftler, bei denen Umfragen geradezu zum Handwerkszeug zählen und ein entsprechend fundiertes Wissen während des Studienganges erworben wird, sondern Umfragen werden auch von Landwirten, Geografen, Juristen und Medizinern durchgeführt bzw. avisiert.
Selbstverständlich kann eine Zentraleinrichtung, wie ein Universitätsrechenzentrum, Mitarbeitern und Studenten, die eine Umfrage durchführen wollen, nur allgemeine Unterstützung bieten, d.h. es darf nicht in jedem Fall Programmieraufwand entstehen. Insbesondere sind die HTML Dokumente, die den Fragebogen im Web realisieren, von den Interviewern selbst beizubringen. Dabei wird unter anderem zum Layout, den Möglichkeiten des <FORM> Tags und zu speziellen Techniken, wie z.B. der Realisierung von Filterführung und Required Response beraten.
Während der Felddauer ausgefüllte Fragebögen liefern ihre Daten in einer oder mehreren eMails, die auf einem POP3-Server gesammelt werden. Alle eMails zu einem Fragebogen bilden je eine Zeile der Datentabelle. Das Datenformat der erhaltenen eMails hängt von einem serverseitig angewandten CGI-Programm ab. Liefert dieses Script beispielsweise Comma-Separated-Value-Dateien (`*.csv`), könnte man den SAS-Import-Wizzard aufrufen und bekäme eine SAS Datentabelle zur weiteren Auswertung. Zwangsläufig bleiben, unabhängig vom CGI-Programm, viele Informationen, die das eMail sendende HTML Dokument bereits enthält, unberücksichtigt und so müssen LABEL und User Defined Formats nachträglich hinzugefügt werden, was bei umfangreichen Fragebögen korrespondierenden Aufwand verursacht. Wird dieser Aufwand vom HTML Dokument entkoppelt, bedeutet jede Modifikation des HTML Dokumentes zugleich eine Überarbeitung des Nachtrags.

Wir haben versuchsweise mit SAS Version 8.2 ein Programm angefertigt, mit dem man aus einem HTML Dokument der Version 4.0, das den Fragebogen repräsentiert, ein SAS Programm generieren kann, das analog der beschriebenen Anwendung des Import-Wizzards aus den eintreffenden eMails eine SAS Datentabelle generiert, wobei gleichzeitig aus dem HTML Dokument möglichst viele Attribute der Variablen, wie z.B. NAME, TYPE, LABEL, FORMAT, LENGTH, sowie deren Voreinstellungen abgeleitet werden.
Schwierigkeiten während der Generierung des Interpreter-Programms resultieren

- aus Inkompatibilitäten zwischen SAS und HTML (z.B. Namenslängen, Case Sensitivität),
- dem extensiven Einsatz von Werkzeugen zur dynamischen HTML Programmierung (z.B. JavaScript und Server Side Includes),
- HTML Syntaxfehlern oder
- noch unentdeckten Fehlern im erstellten Generator-Programm.

In solchen Fällen muss u.U. das HTML Dokument vorab angepasst werden.
Das generierte SAS-Programm ist zunächst nur als eine, allerdings recht brauchbare, Maske anzusehen, d.h. es bedarf wohl stets einer Nacharbeit. Meist besteht die in einer Reduzierung von Texten auf den für LABEL und FORMAT essentiellen Anteil. Der Beitrag

- tangiert die allgemeinen Probleme bei Internet-Umfragen,
- verweist auf einige grundsätzliche Gestaltungshinweise und
- gibt ein vereinfachtes Beispiel.

Keywords: Datenerfassung, Fragebogen, HTML-Formular, SAS-Datentabelle

1 Voraussetzungen

Bevor man eine Umfrage mit Hilfe des Internet durchführt, sollten vorab sorgfältig Vor- und Nachteile, die damit verbunden sind, gegeneinander abgewogen werden.

Nachteilig wirken und überlagern sich:

- fehlende Repräsentativität,
- Selbstselektion,
- fehlende Kontrolle situativer Merkmale (z.B. Beurteilung von Verpackungfarben am Bildschirm)

„Als repräsentativ wird eine Stichprobe angesehen, wenn jede Beobachtungseinheit der Grundgesamtheit die gleiche Chance hatte, in die Stichprobe einzugehen (Zufallsstichprobe) oder wenn in Abhängigkeit von der Zusammensetzung der Grundgesamtheit die Stichprobe entsprechend anteilig zusammengesetzt wird.“ (JANETZKO, 1999, S. 141)

Vorteilhaft und kumulierend wirken:

- Zeitersparnis,
- Mehrfachverwendbarkeit,
- relativ einfache Teilnehmerrekrutierung,
- leichte Einbindung der Ergebnisse in weitere Abläufe

Nachteile kann man nur durch kluge Strategien in ihren Auswirkungen minimieren. Z.B. *nth visitor method*, d.h. jeder n-te Besucher einer Web-Page wird zur Umfrage geleitet, was der Selbstselektion entgegenwirkt. Derartige Strategien sind zwar für Internet-Umfragen sehr wichtig, tangieren aber nicht die Datenerfassung und sind somit kein Bestandteil des Beitrages.

Die Frage, ob eine Internet-Umfrage das geeignete Arbeitsinstrument ist, muss vorab sorgfältig geklärt und positiv entschieden werden.

1.1 Teilung der Arbeitslast

Die von einem universitären Rechenzentrum erwartete Unterstützung ist vielfältig und reicht

- von der Bereitstellung des Fragebogens in Form von Web-Formularen
- über die Datenerfassung
- bis hin zur statistischen Auswertung.

Eine derartige Aufgabenfülle kann auf Dauer nur erfolgreich bewältigt werden, wenn man das Problem „Umfrage via Internet“ in

- dezentral jeweils durch die Auftraggeber und
- zentral durch das Rechenzentrum (natürlich in Abstimmung mit dem aktuellen Auftraggeber)

zu leistende Arbeiten gliedert und die Aufgaben des Rechenzentrums weitestgehend standardisiert:

1. Die **Vorbereitungs- und Entwurfsphase**
   Der erste Entwurf zum Web-Layout der Umfrage, d.h. ein anfängliches **HTML Dokument**, wird **vom Auftraggeber** selbst erstellt. Man kooperiert zumeist mit Studenten und sollte weder deren Fähigkeiten unterschätzen noch deren Kreativität ausschalten, zumal die Autoren einer Umfrage immer Vorstellungen von ihrer Arbeit besitzen, die sie nicht klar formulieren, aber für sich selbst weitestgehend realisieren können. Dies betrifft viele Fragen des Web-Layouts, wie Schriftarten und -größen, Farben, Anordnung der Elemente und auch inhaltliche Aspekte, wie die exakte Formulierung der Fragen.
   Es gibt für den Entwurf von Fragebögen spezielle Generator-Programme, die aber nicht Gegenstand dieses Beitrages sind.

2. **Überarbeitung der Vorlage**
   Das HTML Dokument muss, da es einer **standardisierten Bearbeitung** unterzogen werden soll, einige Bedingungen erfüllen und wird meist aus technischen und nur sehr bedingt aus inhaltlichen Gründen modifiziert. Abschließend wird es auf einem Web-Server unter einer vom Auftraggeber zu propagierenden URL für die Probanden als sogenanntes **pull medium** bereitgestellt, d.h. es wird vom Probanden aufgesucht, auf den eigenen PC geladen, dort von einem Browser visualisiert und vom Probanden ausgefüllt als eMail zurückgesandt (vgl. Abb. 1).

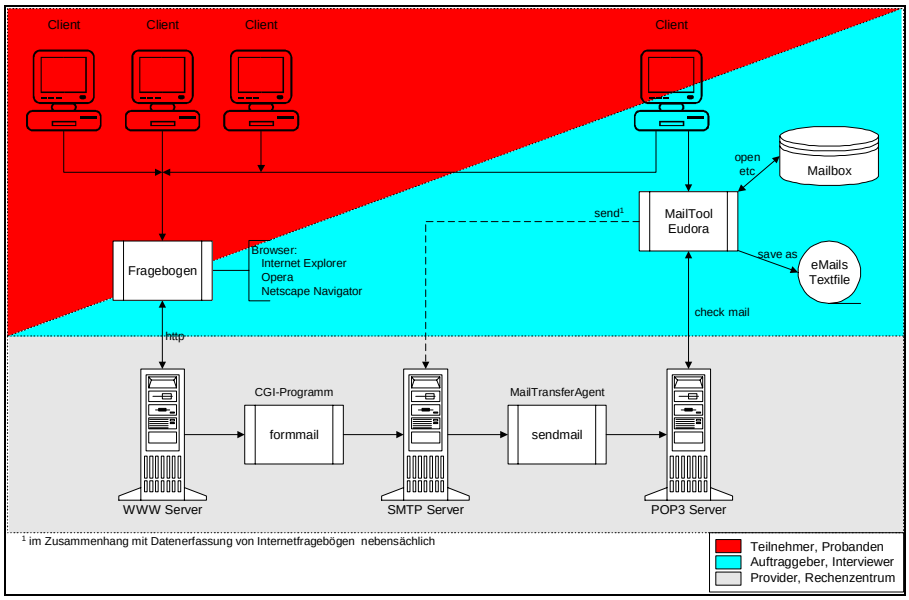

Abbildung 1: Informationsfluss bei Umfragen via Internet

3. **Generierung des Erfassungsprogramms und Pretestphase**
   Aus dem HTML Dokument generiert ein standardisiertes SAS Programm (**Generator**) ein spezifisches SAS Programm (**Interpreter**) zur Erstellung einer SAS Datentabelle aus den per eMail eintreffenden ausgefüllten Formularen und ein Verzeichnis aller erkannten Variablen bzw. Felder und deren

Attribute. Der zunächst abgeleitete Interpreter ist vor seinem Einsatz immer zu editieren und zu testen. (vgl. Abb. 2).

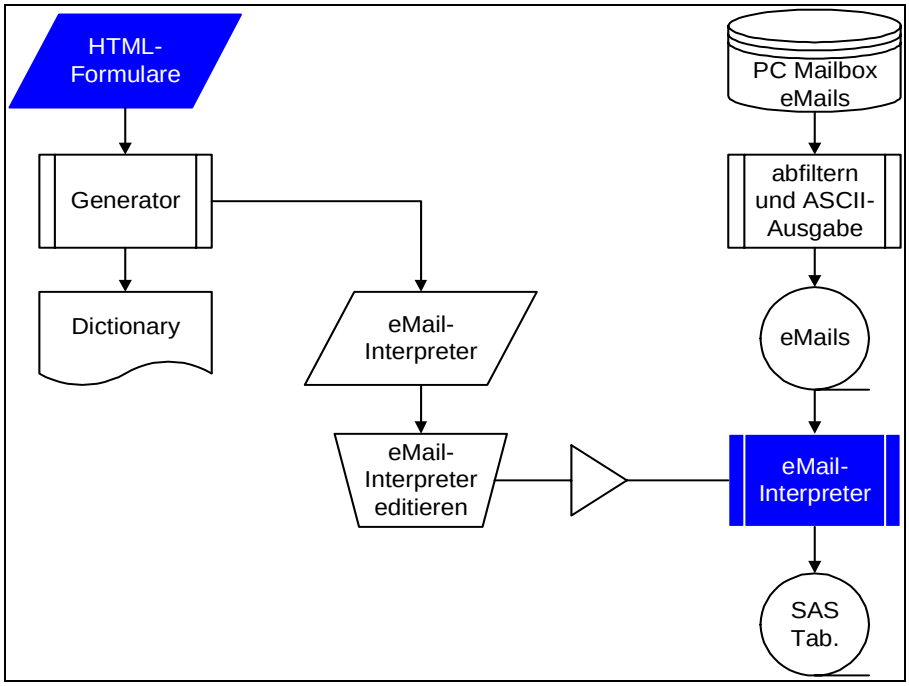

Abbildung 2: Generieren des eMail-Interpreters

4. **Datenerfassung**
   Während der **Felddauer** (Zeitraum für die Vorhaltung der Fragebögen) werden mit dem Interpreter je nach Bedarf, der sicher vom Datenaufkommen abhängt, die Daten der aktuell erhaltenen eMails in einer **aggregierten SAS Datentabelle** (vgl. Abb. 3) erfasst.

5. **Evaluierung**
   Nach Ablauf der Felddauer wird die SAS Datentabelle in einem Evaluierungsschritt von erkennbaren Fehleinträgen bereinigt und bildet dann die **Datenbasis für die statistische Auswertung.**

1.2 Die Entwicklungsumgebung

Internet Explorer, Netscape Navigator und **Opera** sind die Browser, für die das Verfahren erprobt wurde.

SAS ist das Programmsystem, dem die zentrale Rolle zukommt:

- mit dem der Programm-Generator programmiert wurde,
- unter dem das vom Generator erzeugte Interpreter-Programm arbeitet,

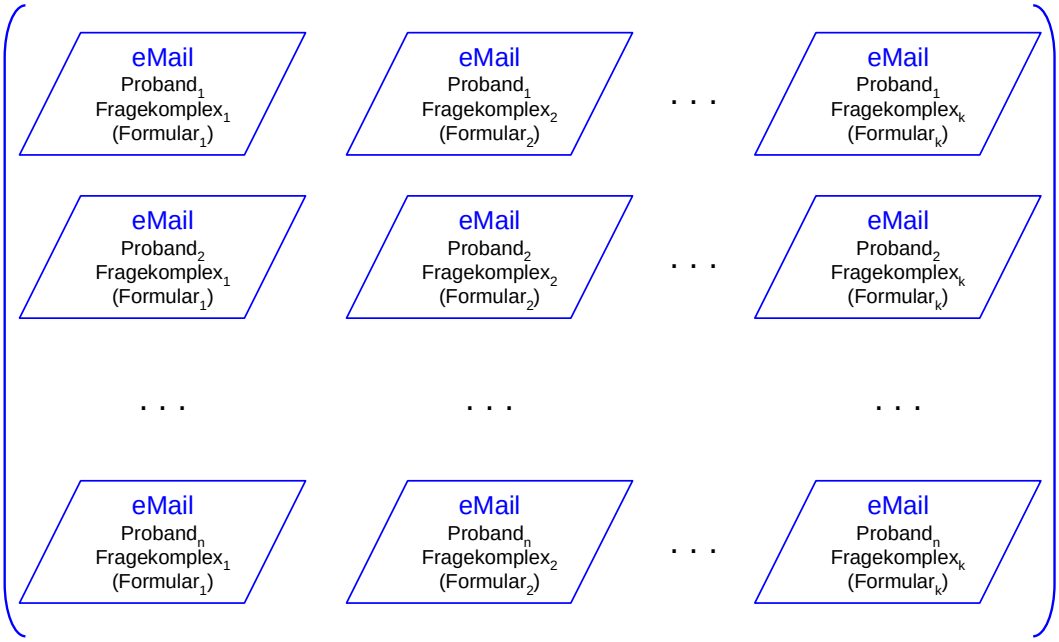

Abbildung 3: Datenaggregation

- in dessen systemeigenen Format die Daten gespeichert werden und
- mit dessen Hilfe die statistischen Auswertungen durchgeführt werden sollten.

Eudora managt als Mail-Tool den Datentransfer vom POP3-Server und die eMails auf dem PC. **FormMail** arbeitet als CGI-Programm und bereitet den Inhalt beliebiger HTML Formulare als Name-Werte-Paare auf. Die Verwendung dieses Perl-Programms ist im vorgestellten Verfahren zwingend. In der Testumgebung auf dem PC wird die für Windows adaptierte Version `winformmail.pl` verwendet. Der **Sambar Server** und **ActivePerl** erlauben, HTML-Formulare im Zusammenspiel mit CGI-Programmen unabhängig vom produktiven Web-Server, dem Administrator und Zugriffsrechten auf dem eigenen PC und z.B. unter Windows zu testen.

Einer **HTML Validierung** sollte man ganz zu Beginn des Projektes die ursprünglich vom Auftraggeber gelieferten HTML Dokumente unterziehen. Syntaxfehler an der HTML 4.0 Notation kann der SAS Programm-Generator im Gegensatz zu den Browsern in keinem erkannten Fall tolerieren. Da jedoch nicht alle Tags vom Generator ausgewertet werden müssen, kann man den Generator seinerseits nicht zur HTML Validierung einsetzen.

Tabelle 1: Verwendete Software

Software	**Version**[1]	**URL**[2]	***Rechenzentrum***	***Auftraggeber***	***Proband***
Windows	98[3]		x		
SAS	8.2	`http://www.sas.com`	x	x	
Fragebogen-Generatoren[4]		vgl. JANETZKO, 1999 S. 314ff	(x)	(x)	
Browser					
Internet Explorer[5]	6.0	`http://www.microsoft.com/downloads`	x	x	x
Netscape Navigator[5]	6.2	`http://home.netscape.com/download`	x	x	
Opera	6.0	`http://www.opera.com`	x	x	
Testumgebung					
Eudora	5.1	`http://www.eudora.com`	x	x	
FormMail	1.6[6]	`http://www.worldwidemart.com/scripts/formmail.shtml`	x		
Sambar Server[7]	5.0	`http://www.sambar.co.za`	x		
ActivePerl	522	`http://www.elsop.com/perl`	x		
HTML-Validatoren					
CSE HTML Validator Lite	2.50	`http://www.htmlvalidator.com/lite`	x		
W3C HTML-Validation Service		`http://validator.w3.org/file-upload.html`			

[1] Installierte und für den Beitrag genutzte Version; wenn nicht explizit anders angegeben im Dezember 2001 aktuelle Version.

[2] Mit Ausnahme von *Windows* und *SAS*, sind von allen Programmen für nichtkommerzielle Zwecke gebühren freie Varianten im Web zu erhalten.

[3] Hier wäre *Windows XP* die aktuelle Version, deren 32 Bit Variante von SAS 8 unterstützt werden soll.

[4] Fragebogen-Generator-Programme wurden nicht verwendet und sind kein Gegenstand des Beitrages, d.h. sie werden hier zur Beachtung empfohlen, aber in keiner Weise bewertet.

[5] Diese beiden Browser sind die am weitesten verbreiteten und ihr gemeinsamer Marktanteil wird auf mehr als 90% geschätzt (STEYER, 1997, S. 94).

[6] Hier wäre *FormMail* 1.9 die aktuelle Version; unter Windows 98 kommt für Testzwecke eine bzgl. des Unix-Kommandos *sendmail*, das als Mail Transfer Agent (MTA) agiert, modifizierte Programmvariante *winformmail.pl* der Version 1.6 zur Anwendung.

[7] Der *Sambar Server* wird nur zu Testzwecken genutzt und muss dazu als WWW Server und SMTP Proxy Server konfiguriert und aktiviert werden; setzt auf dem PC die Installation von ActivePerl voraus.

Tabelle 2: Angewendete Computer Sprachkonzepte

Computer-Sprachen	**Client**[8]	**Server**[8]	erforderliche **Kenntnisse**[9]		
			Auftraggeber	*Entwickler*	*Teilnehmer*
HTML	x		gut	sehr gut	
JavaScript[10]	x			gut	
Cascaded Style Sheets	x		gut	gut	
Perl		x		gering	
Base SAS Language[11]	x		gering	sehr gut	

In der Entwicklungs- bzw. Testumgebung für Fragebögen

- wird der produktive Web-Server (vgl. Abb. 1) durch den *Sambar Server*, der auf dem Entwickler PC als WWW Server und SMTP Proxy Server installiert wird, ersetzt (vgl. Abb. 4),

- werden auf dem lokalen *Sambar Server* die HTML Dokumente der Fragebögen einschl. aller Script Functions, Images, Sounds usw. ab dem Verzeichnis *cgi-bin* gespeichert,

- ist in den <FORM> Tags des Fragebogens dem ACTION Attribut als URL des CGI-Programms `http://127.0.0.1/cgi-bin/winformmail.pl` zuzuweisen.[12]

Eine Bereitstellung differenzierter Zugänge für den Entwickler mit besonderem Protokollierungs Modus und den Probanden ohne zusätzliche Protokollierung hat sich als günstig erwiesen und ist in den HTML Dokumenten leicht zu etablieren (vgl. Fragekomplexe: Anmerkungen 9-12).

[8] In der Testumgebung sind Client und Server derselbe PC (Entwicklerarbeitsplatz).

[9] Die Einteilung ist als grobe Orientierung für eine Minimalanforderung gedacht. Dabei bedeuten:

keine Angabe	=	Kenntnisstand beliebig, weil nicht gefordert
sehr gut	=	selbständige kreative Anwendung
gut	=	reproduzierende, am Beispiel orientierte Anwendung
gering	=	black-box-Anwendung bestehender Programme

[10] Event handling wird im Beispiel dieses Beitrages ausschl. mit JavaScript realisiert. Die Auswahl der Script Sprache erfolgte ohne Wertung gegenüber anderen Script Sprachen, wie z.B. VisualBasic.

[11] SAS Installationen benötigen nur der Auftraggeber und der Entwickler jedoch nicht die Teilnehmer.

[12] Nach Fertigstellung des Fragebogens erfährt die Zuweisung der URL die einzige Änderung an den HTML Dokumenten vor deren Übernahme auf den produktiven Web-Server. Neben der Server Adresse ist als CGI-Programm dann das Perl Script ***formmail.pl*** (statt *winformmail.pl*) aufzurufen.

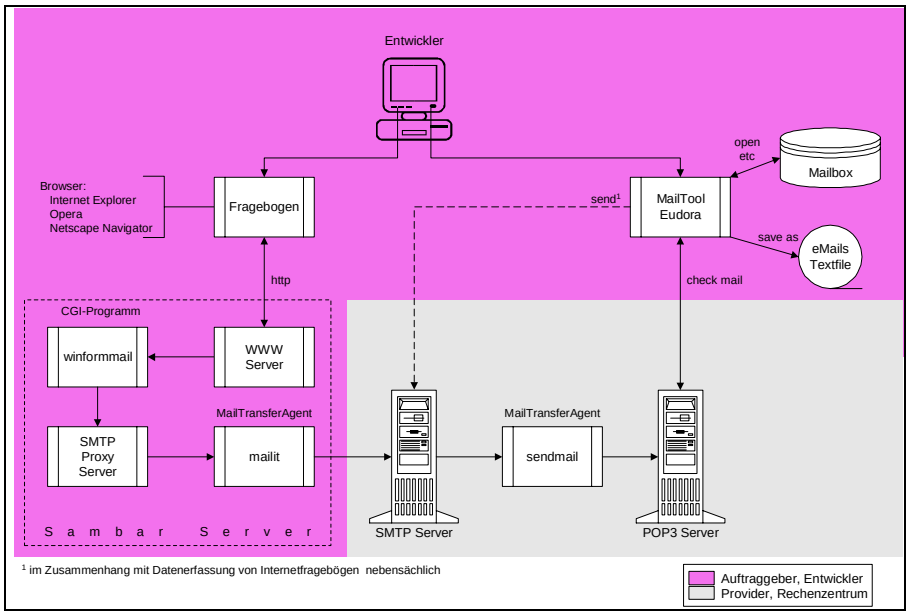

Abbildung 4: Informationsfluss bei Umfragen via Internet (Testumgebung mit *Sambar Server*)

2 Gestaltung von Internet-Fragebögen

2.1 Grundlegende Techniken

Eine erste Seite (ohne Fragestellungen) sollte folgendes beinhalten (vgl. JANETZKO, 1999, S.56):

- Anrede des Befragten
- Ziel der Untersuchung - soweit dies nicht die Untersuchung gefährdet
- Untersuchende Einrichtung
- Name und eMail-Adresse des verantwortlichen Untersuchungsleiters
- Anleitung zur Bearbeitung des Erhebungsinstruments
- Voraussichtliche Bearbeitungsdauer
- Hinweise zur Verwendung der erhobenen Daten
- Zusicherung der Anonymität
- Antwortappell
- gegebenenfalls Rücklauftermin
- Danksagung für Mitarbeit
- Falls Anreize (incentives) für nötig erachtet werden, sollten sie möglichst geringwertig sein, damit keine Verfälschung der Motivation für die Beantwortung auftritt

Erst von dieser Einführungs-Seite gelangt ein Proband zum ersten Fragekomplex.

Die Zerlegung des ursprünglich angelieferten Fragebogens in Fragekomplexe ist der massivste und meist erforderliche Eingriff in das von den Auftraggebern angelieferte urprüngliche HTML Dokument und resultiert aus der zunächst unzureichenden technischen Umsetzung der Steuerungs- und Gestaltungsmöglichkeiten bei Erhebungen im Internet (vgl. Tab. 3):

Tabelle 3: Steuerungs- und Gestaltungsmöglichkeiten bei Erhebungen im Internet (JANETZTKO, 1999, S. 163)

Merkmal	**Erläuterung**
Randomisierung	Zufallsgesteuerte Darbietung aller oder eines Teils der Items eines Erhebungsinstrumentes
Popping-Up	Dem Besucher einer Web-Seite wird ein Fragebogen präsentiert, sobald ein bestimmtes Kriterium erreicht ist
Range-Checks	Überprüfung plausibler Bereiche (z.B. bei Altersangaben)
Filterführung	Antwortabhängige Präsentation von Nachfolge-Items
Kontrolle von Mehrfachbearbeitungen	z.B. durch Kontrolle von IP-Adresse und Zeitfenster
Automatische Erzeugung von Datendateien	Via CGI-Programmierung Anlegen von Datendateien, die in Statistik-Programme oder Datenbanken einlesbar sind
Automatische Statistische Analyse	Online zeitgetaktet oder auf Anfrage möglich
Zeitmessungen	Messung der Zeiten, die für die Beantwortung von Items benötigt wurden
Re-Entry	Präsentation von Ergebnissen in einer Netzumgebung (als Ergebnisrückmeldung für Teilnehmer einer Erhebung oder als Online-Statistik)
Paßwort-Schutz	Vorkehrungen gegen unbefugtes Bearbeiten von Erhebungsbögen

Auf automatischer Erzeugung von Datendateien liegt der Schwerpunkt des Beitrages. Er wäre jedoch recht nutzlos, wenn nicht verfahrenskonforme Lösungen für andere wichtige Techniken wie z.B.

- Plausibilitätskontrollen einschl. Konsistenzprüfungen,
- Filterführung (bedingte Fragen oder Fragekomplexe) bzw.
- One-Screen-One-Question (unbedingte Fragen oder Fragekomplexe),
- Fortschrittsanzeige und
- Kontrolle der Mehrfachbearbeitung aufgezeigt würden.

2.2 Das Beispiel

Das gesamte Beispiel ist fiktiv und veranschaulicht lediglich einige Möglichkeiten und deren Umsetzung. Ziel ist, ein Verfahren zur Speicherung der Daten aus Internetfragebögen in SAS Datentabellen vorzustellen.

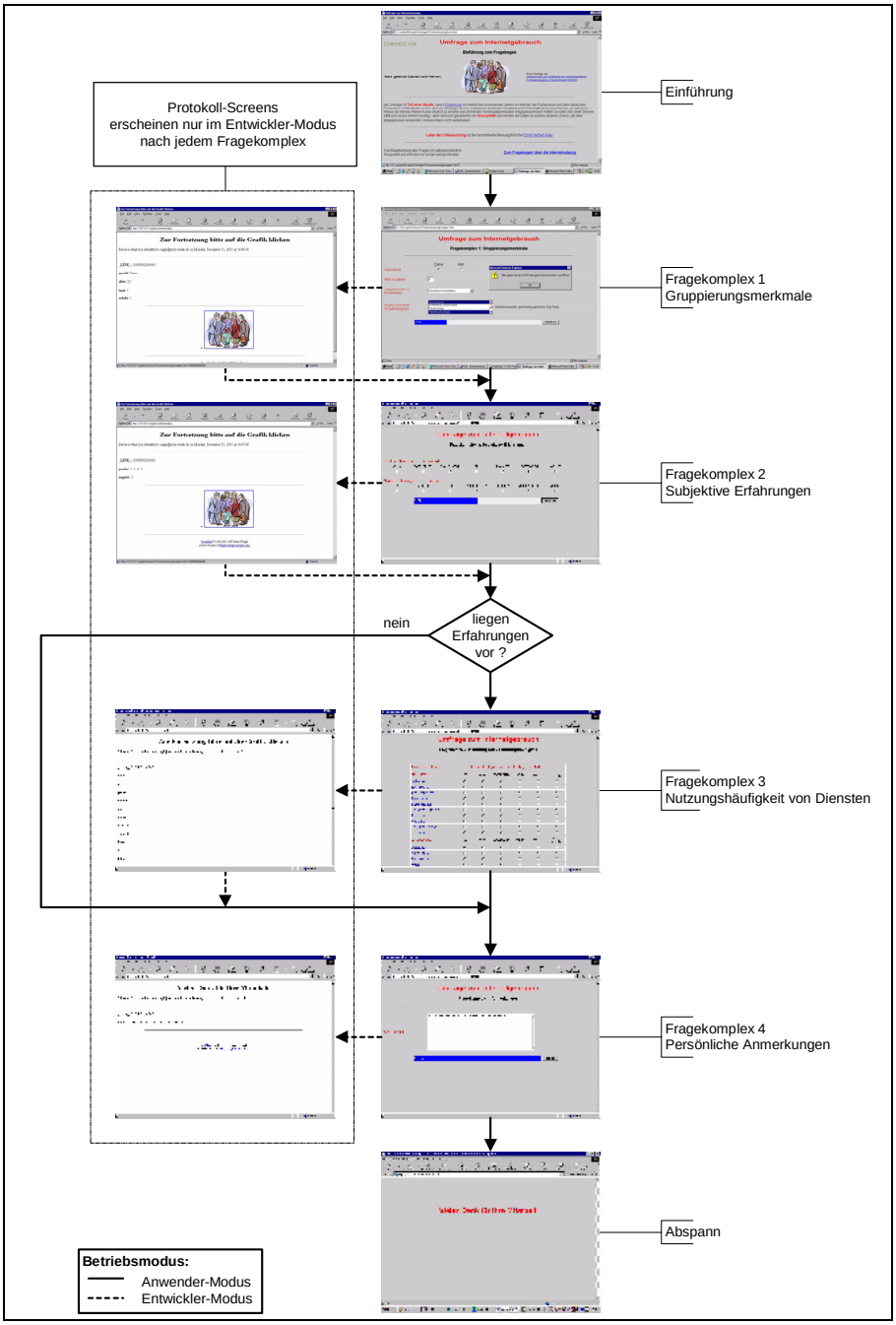

Abbildung 5: Die Screens der Beispiel-Umfrage „Nutzung des Internets“

2.2.1 Die Einführungseite

Sambar50/cgi-bin/Umfrage/InternetnutzungCover.htm[13]

Abbildung 6: Die Einführungsseite (Browserdarstellung durch Internet Explorer)

Der **HTML Quelltext** der Einführungsseite ist einfach und enthält lediglich einige Verweise auf:

- die **Homepage** der die Untersuchung durchführenden Firma oder Einrichtung,
- die **optimalen Browser** und die zugehörigen download Webseiten der Herstellerfirmen,
- eine **softwareseitige Unbedenklichkeitsversicherung**[14],
- die späteren oder auch laufend veröffentlichten **Resultate**,
- die Erreichbarkeit des **Untersuchungsleiters** (Name, eMail Adresse) und
- die beiden Verweise auf das Dokument mit dem **Fragekomplex 1**, die durch JavaScript im query string mit der *document.write* Methode dynamisch ergänzt werden:

```
<SCRIPT>
   var today = new Date();
   document.write
   ("<A HREF='InternetnutzungKomplex1.htm?2" + today.getTime() + "'>");
</SCRIPT>
   Zum Fragebogen &uuml;ber die Internetnutzung</A>
```

[13] Die Datei befindet sich auf der Proceedings-CD

[14] Zur Verwendung von Cookies vgl. `http://www.schoenberger-dix.de/index_2.html`

Der **query string** folgt der URL nach dem ? und besteht lückenlos aus zwei Teilen:

- dem Code für den Betriebsmodus
  (1 = Entwickler-Modus; 2 = Anwender-Modus) und
- der Ortszeit in ms seit dem 1.1.1970
  (= untergeordneter Schlüssel des Fragebogen ID)

Es gibt einen zweiten unsichtbaren Zugang zum Fragebogen (transparente Grafik; oben links am Hinweis-Text erkennbar), der den **Entwickler-Modus** aktiviert. Für diese Arbeitsweise ist das CGI-Programm *FormMail* mit entsprechender Konfiguration anzusteuern, d.h. sind in den Fragekomplexen drei konfigurierenden Variablen die angegebenen Werte zuzuweisen. Diese Arbeitsweise ist zwar für den Entwickler vorgesehen, es schadet aber nicht, wenn ein Anwender damit operiert.

2.2.2 Fragekomplexe

Jeder Fragekomplex wird als HTML Formular realisiert, dessen Inhalt mit der POST Methode durch das CGI-Programm FormMail als eMail an die vereinbarte Adresse gesandt wird. Sobald mehrere Fragekomplexe zu einem Fragebogen gehören, muss eine eineindeutige Zuordnung der eintreffenden eMails zu den Fragebögen möglich sein. Aus diesem Grunde wird jede eMail mit zwei **Schlüsselmerkmalen** versehen:

- einem übergeordneten Schlüssel = IP-Adresse des Absender-PC und
- einem untergeordneten Schlüssel = Uhrzeit in ms bei Aufruf des ersten Komplexes

Die **IP-Adresse** gehört zu den CGI-Environment-Variablen und kann durch FormMail ausgelesen und zusätzlich zu den Variablen der Eingabefelder des Formulares mit der eMail versendet werden. Die **Tageszeit in Millisekunden** (seit dem 1.1.1970) wird durch eine JavaScript Function von der Einführungsseite bereitgestellt und von dort als Wert der CGI-Environment Variablen QUERY_STRING von Fragekomplex zu Fragekomplex weitergereicht.

Fragekomplex 1: Gruppierungsmerkmale

Sambar50/cgi-bin/Umfrage/InternetnutzungKomplex1.htm[15]

Abbildung 7: Fragekomplex 1 „Gruppierungsmerkmale“(mit aktivierter Meldung zur Plausibilitätskontrolle beim Eingabefeld *alter*; das Eingabefeld *kinder* ist ein popUp-Feld und wird erst ab $alter > 15$ sichtbar)

Bitte beachten Sie die um einen query string ergänzte **URL**.

HTML Quelltext zu Fragekomplex 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
   <TITLE>Umfrage zur Internetnutzung</TITLE>
   <META  HTTP_EQUIV="Content-Type" CONTENT="text/html;
          charset=iso-8859-1">
   <META  NAME="author" CONTENT="W.F.">
   <SCRIPT SRC="start.js">   </SCRIPT>                    (1)
   <SCRIPT SRC="checkInt.js"></SCRIPT>
   <SCRIPT SRC="bar.js">     </SCRIPT>
```

[15]Die Datei befindet sich auf der Proceedings-CD

```
<SCRIPT LANGUAGE="JavaScript">                                          ②
<!--
function popUp(alter,kinder)
{
 if (document.getElementById)
 {
  if (alter.value > 15)
  {
   if (document.getElementById('KinderFrage').style.visibility
       == 'hidden')
   {
    document.getElementById('KinderFrage').style.visibility
    ='visible';
    document.getElementById('KinderAnzahl').style.visibility
    ='visible';
    if (navigator.userAgent.indexOf('MSIE') != -1 &&
        navigator.userAgent.indexOf('Opera')== -1)
        document.sound.play();
   }
  }
  else { document.getElementById('KinderFrage').style.visibility
         ='hidden';
         document.getElementById('KinderAnzahl').style.visibility
         ='hidden';
         kinder.value='';
       }
 }
}
//-->
</SCRIPT>
```

```
</HEAD>
```

```
<BODY BGCOLOR="#CCCCCC"                                                 ③
      onLoad ="popUp(document.forms[0].alter,
                document.forms[0].kinder);">
```

```
<FONT FACE="arial">
```

```
<FORM NAME    ="komplex1"                                               ④
      METHOD  ="post"
      ACTION  ="http://127.0.0.1/cgi-bin/winformmail.pl"
      onSubmit="start('Umfrage/InternetnutzungKomplex2.htm');
                return checkInt(document.forms[0].alter,1,99) &&
                       checkInt(document.forms[0].kinder,0,12)">
```

```
<!-- 1. Wohin soll der Server das aufbereitete Formular als eMail
        senden? -->
```

```
<INPUT TYPE ="hidden"                                                    (5)
       NAME ="recipient"
       VALUE="eagle@rz.hu-berlin.de">
```

```
<!-- 2. Woran sollen die eMails, die das aufbereitete Formular
        enthalten, erkennbar sein? -->
```

```
<INPUT TYPE ="hidden"                                                    (6)
       NAME ="subject"
       VALUE="Internetnutzung">
```

```
<!-- 3. Es folgen zwei Eingabefelder, die technisch bedingt sind,
        um die eineindeutige Zuordnung der eintreffenden eMails
        zum zugehörigen Fragebogen zu realisieren:
     a) Partielles Auslesen der CGI-Environment-Variablen
        REMOTE_ADDR=IP-Adresse; verwendbar als übergeordneter
        MERGE-Schlüssel HTTP_USER_AGENT=Browser;
        Protokoll zum verwendeten Client (optional) -->
```

```
<INPUT TYPE ="hidden"                                                    (7)
       NAME ="env_report"
       VALUE="REMOTE_ADDR, HTTP_USER_AGENT">
```

```
<!-- b) _LINK_=Konstante; verwendbar als untergeordneter
        MERGE-Schlüssel Wertzuweisung zum VALUE-Attribut
        erfolgt durch JavaScript-Function start.js -->
```

```
<INPUT TYPE ="hidden"                                                    (8)
       NAME ="_LINK_">
```

```
<!-- 4. Aufruf des nachfolgenden Fragekomlexes durch Einstellung von
        Konfigurationsvariablen des CGI-Programms (FormMail)
     A) ohne Protokoll (Anwender-Modus)
        URL des nachfolgenden Fragekomplexes (oder Trailers)
        Wertzuweisung zum VALUE-Attribut erfolgt durch
        JavaScript-Function start.js -->
```

```
<INPUT TYPE ="hidden"                                                    (9)
       NAME ="redirect">
```

```
<!-- B) mit Protokoll (Entwickler-Modus)
     a) URL des nachfolgenden Fragekomplexes (oder Trailers)
        Wertzuweisung zum VALUE-Attribut erfolgt durch
        JavaScript-Function start.js -->
```

```
<INPUT TYPE ="hidden"                                                    (10)
       NAME ="return_link_url">
```

```
<!-- b) Titel des Protokolls als Bedienungshinweis
        umfunktionieren -->
```

```
<INPUT TYPE ="hidden"                                                    (11)
       NAME ="title"
       VALUE="Zur Fortsetzung bitte auf die Grafik klicken">
```

```
<!-- c) Darstellung für die Schaltfläche zur Fortsetzung mit nächstem
        Fragekomplex nur relevant, wenn return_link_url.value nicht
        leer bleibt -->
```

```
<INPUT TYPE ="hidden"                                                    (12)
       NAME ="return_link_title"
       VALUE="&lt;CENTER&gt;&lt;
              IMG SRC='Umfrage/UmfrageLogoTr.gif'
                  ALT='&Uuml;bergang zum n&auml;chsten
                        Fragenkomplex'&gt;
              &lt;/CENTER&gt;">
```

```
<!-- 5. Reaktion auf fehlende notwendige Eingaben durch Einstellung
        von Konfigurationsvariablen des CGI-Programms (FormMail)
     a) VALUE enthält URL des Dokuments mit der Fehlerinformation;
        ist der engl. Standardversion vorzuziehen, da dort zwar die
        Namen der betroffenen Eingabefelder angegeben werden, was
        aber einem Anwender wenig sagt -->
```

```
<INPUT TYPE ="hidden"                                                    (13)
       NAME ="missing_fields_redirect"
       VALUE="Error.htm">
```

```
<!-- b) Liste der Namen der unbedingt auszufüllenden Eingabefelder
        bedingt auszufüllende Felder und Plausibilitäts- und
        Konsistenzkontrollen sind durch event handling und
        assoziierte JavaScripts zu realisieren die Liste darf keine
        Namen enthalten, die nicht im Formular auftreten -->
```

```
<INPUT TYPE ="hidden"                                                    (14)
       NAME ="required"
       VALUE="geschl, land, schubi">
```

```
<!-- Fragenkomplex 1:
     Gruppierungsmerkmale zur Probandenklassifizierung -->

<CENTER>
<H1><FONT COLOR="#FF0000">Umfrage zum Internetgebrauch</FONT></H1>
<H3>Fragekomplex 1: Gruppierungsmerkmale</H3>
</CENTER>
<HR WIDTH="80%">

<!-- 6. Das Geschlecht des Probanden -->
<P>
```

```
<TABLE BORDER="0">                                                    (15)
<TR><TD WIDTH="200">
    <TD WIDTH="100"><CENTER>Dame</CENTER>
    <TD WIDTH="100"><CENTER>Herr</CENTER>
<TR><TD><FONT COLOR="#FF0000">Geschlecht</FONT>
    <TD><CENTER><INPUT TYPE ="radio"
                       NAME ="geschl"
                       VALUE="Dame">
        </CENTER>
    <TD><CENTER><INPUT TYPE ="radio"
                       NAME ="geschl"
                       VALUE="Herr">
        </CENTER>
</TABLE>
```

```
</P>

<!-- 7. Filterführung innerhalb eines Screens
        (abhängige Eingabefelder)
        Steuerung der Sichtbarkeit von PopUp-Feldern

     a) Das Alter des Probanden
        TYPE=TEXT ist HTML-Voreinstellung
        bei rein numerischen Eingaben sollte SIZE die Stellenanzahl
        enthalten und TYPE für das vorgestellte Generator-Programm
        nicht expliziert werden.
        Der event handler onBlur wird aktiv, wenn das Feld den Fokus
        verliert und bewirkt hier die Überprüfung der Eingabe bzgl.
        der Lage innerhalb des (min,max)-Intervalls
       (im Beispiel: min=1 max=99)

     b) Anzahl der leiblichen Kinder
        Vom Alter abhängiges Feld: Frage soll nur erscheinen,
        wenn Alter>15 ist. -->
```

```
<TABLE>                                                              (16)
   <TR><TD WIDTH="200">
           <FONT COLOR="#FF0000">Alter in Jahren</FONT>
       <TD><INPUT NAME ="alter"
                  SIZE ="2"
                  MAXLENGTH="2"
                  onBlur    ="checkInt(alter,1,99);
                              popUp(alter,kinder);"
                  onMouseOut="popUp(alter,kinder);">
       <TD WIDTH="100"> 
       <TD WIDTH="220" ID="KinderFrage">
           <FONT COLOR="#FF0000">Anzahl der leiblichen Kinder
           </FONT>
       <TD ID="KinderAnzahl">
           <INPUT NAME ="kinder"
                  SIZE ="2"
                  MAXLENGTH="2"
                  onBlur="checkInt(kinder,0,12);
                          if (this.form.kinder.value == '0')
                             this.form.kinder.value = '00';">
</TABLE>
```

```
<!-- 8. Das Bundesland, in dem der Proband seinen Hauptwohnsitz in
        Deutschland hat; hier realisiert als Beispiel
        für Auswahlliste mit exklusiver Antwortmöglichkeit
        wäre auch mit TYPE=RADIO realisierbar -->
```

```
<TABLE><TR><TD WIDTH="200" VALIGN="middle">                          (17)
       <FONT COLOR="#FF0000">Hauptwohnsitz in Bundesland </FONT>
       <TD><SELECT NAME ="land" SIZE ="1">
           <OPTION VALUE=0>
           <OPTION VALUE=1>Schleswig-Holstein
           <OPTION VALUE=2>Freie und Hansestadt Hamburg
           <OPTION VALUE=3>Freie Hansestadt Bremen
           <OPTION VALUE=4>Niedersachsen
           <OPTION VALUE=5>Nordrhein-Westfalen
           <OPTION VALUE=6>Hessen
           <OPTION VALUE=7>Rheinland-Pfalz
           <OPTION VALUE=8>Baden-W&uuml;rttemberg
           <OPTION VALUE=9>Freistaat Bayern
           <OPTION VALUE=10>Saarland
           <OPTION VALUE=11>Bundeshauptstadt Berlin
           <OPTION VALUE=12>Brandenburg
           <OPTION VALUE=13>Mecklenburg-Vorpommern
           <OPTION VALUE=14>Sachsen-Anhalt
           <OPTION VALUE=15>Freistaat Sachsen
           <OPTION VALUE=16>Freistaat Th&uuml;ringen
           <OPTION VALUE=17>nicht in Deutschland
       </SELECT></TABLE>
```

```
<!-- 9. Die Schulbildung des Probanden
        hier realisiert als Beispiel für Auswahlliste mit
        Mehrfachantwortmöglichkeit
        liefert einen Vektor mit n(=12) Komponenten
        Konsistenzprüfung durch Script Programmierung
        wäre auch mit TYPE=CHECKBOX realisierbar -->
```

(18)

```
<TABLE>
<TR><TD WIDTH="200" VALIGN="middle">
        <FONT COLOR="#FF0000">Abgeschlossene Schulbildung(en)
        </FONT>
    <TD><SELECT NAME ="schubi"
                SIZE ="4"
                MULTIPLE
                onBlur="var n=this.form.schubi.options.length;
                        if (this.form.schubi.options[0].selected)
                        {
                        for (var i=1; i<n; i++)
                        {
                        if (this.form.schubi.options[i].selected)
                        {
                   alert ('Bitte korrigieren Sie den Widerspruch
                           in der Angabe SCHULBILDUNG');
                        for (var j=0; j<n; j++)
                   this.form.schubi.options[j].selected=false;
                        }
                        }
                        }">
        <OPTION VALUE=1>keine abgeschlossen
        <OPTION VALUE=2>Gesamtschule
        <OPTION VALUE=3>Allg.bildende Polytechnische Oberschule
        <OPTION VALUE=4>Realschule
        <OPTION VALUE=5>Gymnasium
        <OPTION VALUE=6>Erweiterte Oberschule
        <OPTION VALUE=7>Fachschule
        <OPTION VALUE=8>Fachhochschule
        <OPTION VALUE=9>Hochschule
        <OPTION VALUE=10>Spezialschule
        <OPTION VALUE=11>Berufsschule
        <OPTION VALUE=12>sonstige Schule
        </SELECT>
    <TD><FONT SIZE="-1"> 
        Mehrfachauswahl:
        gleichzeitig gedr&uuml;ckte <I>Strg</I>-Taste
        </FONT>
</TABLE>
```

```
<!-- *. Notwendiger SUBMIT-Button -->
<CENTER>
```

```
<TABLE BORDER="0" WIDTH="700">                                    (19)
<TR><SCRIPT>
        bar(25)
    </SCRIPT>
    <TD WIDTH="100">
        <INPUT TYPE ="submit"
               VALUE="weiter &gt;&gt;">
</TABLE>
```

```
</CENTER>
</FORM>
</FONT>
```

```
<EMBED NAME="sound" SRC="Erinner.wav" AUTOSTART="false"           (20)
       HIDDEN="true" LOOP="false">
```

```
</BODY>
</HTML>
```

Anmerkungen

Die Anmerkungen 1-14 sind verfahrenstechnisch bedingt. Der HTML Code dieser Abschnitte ist in den Fragekomplexen 1 bis 4 fast identisch, wobei die marginalen Unterschiede

- in ggf. mehr oder weniger zu ladenden JavaScript Functions (vgl. Anmerkung 1)
- in lokal benötigtem JavaScript Code, (vgl. Anmerkung 2)
- in der Bezeichnung des Formulars (vgl. Anmerkung 3)
- im Namen des Nachfolge-Dokumentes und (vgl. Anmerkung 3)
- der Benennung evtl. notwendig auszufüllender Felder bestehen. (vgl. Anmerkung 14)

1. Einfügung von **JavaScript**[16] Code vom Server in das HTML Dokument, der vom Client (Browser) ausgeführt wird (vgl. JavaScript Quelltexte)

   Relative Pfade beziehen sich auf die URL des beherbergenden HTML Dokumentes. Im Beispiel
   `http://127.0.0.1/cgi-bin/Umfrage/InternetnutzungKomplex1.htm` werden die JavaScript Dateien im Verzeichnis *cgi-bin/Umfrage* des WWW Servers gesucht. Die JavaScript Dateien sind nicht an diese Position gebunden, d.h. sie können in beliebigen Server Verzeichnissen gespeichert werden.

[16] Bei der Codierung von JavaScript Anweisungen, ist Groß- und Kleinschreibung zu beachten, da die Sprache **casesensitiv** ist!

2. Nutzereigener **lokaler JavaScript Code**[17]
   Die Function *popUp* wird hier zur Steuerung der Sichtbarkeit des Fragetextes und des Eingabefeldes zur *Anzahl der leiblichen Kinder* unter Beachtung ihrer beider **Abhängigkeit** vom *Alter* des Probanden genutzt. Es wird nur nach Kindern gefragt, wenn die Person älter als 15 Jahre ist.

   Spezifisch wird vom Internet Explorer ein **akkustisches Signal** ausgegeben, wenn das abhängige Feld nachträglich sichtbar wird. Das Signal muss in einer Audiodatei abgespeichert vorliegen und wird mit einem <EMBED> Tag in das HTML Dokument eingebunden. (vgl. Anmerkung 20) Da dies so nur bei Verwendung des Internet Explorers funktioniert, muss dessen aktuelle Verwendung zur Laufzeit abgetestet werden. Dabei ist mit größter Sorgfalt vorzugehen, da der Bowser *Opera 6.0* als *navigator.appName* irreführend *Microsoft Internet Explorer* angibt!

   Bei Verwendung von ältlichen Browsern, die die Methode *getElementById* nicht unterstützen, wird das abhängige Feld stets unbedingt veröffentlicht. Dadurch können auch Personen, die jünger als 16 Jahre sind plötzlich Werte für die Anzahl der Kinder aufweisen!

3. Globale Definitionen zum **Formular**
   `NAME="name"` Bezeichnung des Formulars;
   Alle Beispiele sind so gehalten, dass das NAME Attribut im <FORM> Tag stets optional ist; Es könnte aber in JavaScript Anweisungen diskret adressiert werden.

   `METHOD="post"` Methode der Datenübermittlung an das CGI-Programm
   Im Gegesatz zu METHOD=„get“, bei der maximal 256 Zeichen per QUERY_STRING übermittelt werden, wird hier eine zunächst unbegrenzte Zeichenmenge via Datei *stdin* übertragen; METHOD=„post“ ist für das Verfahren unverändert beizubehalten.

   `ACTION="CGI-Programm"` URL eines CGI-Programms
   Wird von einem (aktiven) WWW Server ausgeführt (im Beispiel: 127.0.0.1). Muss hier ab Verzeichnis *cgi-bin* gespeichert werden, d.h. in einem expliziten Pfad muss dieses Verzeichnis immer vorkommen; relative Pfade dürfen dieses Verzeichnis nicht toppen; Programm dient der Aufbereitung der eMail (arbeitet als Mail User Agent).
   In den Beispielen ruft die URL `http://127.0.0.1/cgi-bin/winformmail.pl`[18] ein für das Windows Betriebssystem leicht modifiziertes *FormMail* Programm und lässt dies vom *Sambar Server*, der zur Testumgebung (vgl. Die Entwicklungsumgebung) gehört und seinerseits natürlich gestartet sein muss, ausführen.
   Bei produktiver Anwendung ist die URL bzgl. Server und Programm zu modifizieren: `http://server/cgi-bin/formmail.pl`

[17] Bei der Codierung von JavaScript Anweisungen, ist Groß- und Kleinschreibung zu beachten, da die Sprache **casesensitiv** ist!
[18] Die Datei befindet sich auf der Proceedings-CD

`onSubmit="start(parameter)"` event handler
Wird vor Übergabe der Steuerung an das CGI-Programm (vgl. Anmerkung 19) aktiviert und führt abschließende Operationen aus.
Es erfolgt ein Aufruf der JavaScript Function ***start*** mit dem angegebenen Parameter. Als Parameter kann der Name eines nachfolgenden Dokumentes incl. des Pfades angegeben werden. Eine leere Zeichenkette signalisiert dabei das Ende des Fragebogens.
Die Dokumente müssen ab dem Verzeichnis *cgi-bin* auf einem (aktiven) WWW Server (im Beispiel: 127.0.0.1) gespeichert werden, d.h. in einem expliziten Pfad muss dieses Verzeichnis immer vorkommen; relative Pfade dürfen dieses Verzeichnis nicht toppen. Zusätzlich ist nochmals die **Plausibilitätsprüfung**[19] aller betreffenden Variablenwerte zu wiederholen, da nicht sicher ist, dass der Proband vorherige Warnungen beachtet hat. Wurden Warnungen ignoriert, erfolgt keine Übergabe des Formularinhaltes an das CGI-Programm. Sind mehrere Variablen zu testen, ist die Konjunktion der Rückkehrwerte der *checkInt* Function zu bilden.

Verzichtet der Fragebogen-Entwickler auf clientseitige Plausibilitätschecks einschl. des Tests auf gültige Zahlendarstellungen, wird das generierte Interpreter-Programm u.U. ungültige Argumente der SAS Function *INPUT* monieren und folgerichtig Missing Value zuweisen.

4. Sobald bedingte Visualisierung von Teilen einer Web-Seite erforderlich ist, müssen mit Hilfe des **onLoad** event handlers im <BODY> Tag bereits beim Laden des HTML Dokumentes die Bedingungen für Sichtbarkeit bzw. Unsichtbarkeit berücksichtigt werden. Dazu sollte die JavaScript Function (hier *popUp*) aufgerufen werden, die auch auf die Veränderungen infolge von Eingaben auf das steuernde Merkmal (hier *alter*) reagiert (vgl. Anmerkung 16).

5. Eingabefelder vom `TYPE="hidden"` werden vom Browser nicht visualisiert. Ihre Werte werden aber an das CGI-Programm (*FormMail*) übermittelt und vom Entwickler u.a. zur Steuerung des CGI-Programms eingesetzt. Für diesen Fall sind die Namen der Felder und ihr Wertevorrat vorgegeben. (vgl. Tab. 4)[20]

   `NAME="recipient"` ⇒ VALUE enthält die **eMail Adresse des Empfängers**
   `VALUE="eMail-Adresse"` eMail-Adresse des Empfängers
   Dieses Eingabefeld enthält eine **notwendige Angabe** des Entwicklers (Auftraggebers)

6. Zu `TYPE="hidden"` vgl. Anmerkung 5
   `NAME="subject"` ⇒ VALUE enthält den Inhalt des **Subject Feldes der eMail**
   `VALUE="text"` Inhalt des Subject-Feldes aller eMails, die vom Formular initiiert werden; dient als Filter für die Mailbox (vgl. Speichern der eMails als Textdateien)

[19] Plausibilitätskontrollen schließen natürlich stets den Test auf gültige externe Zahlendarstellung ein.

[20] Man muss Groß- und Kleinschreibung exakt einhalten, da das CGI-Programm in Perl codiert wurde und Perl eine casesensitive Sprache ist!

7. Zu `TYPE="hidden"` vgl. Anmerkung 5
   `NAME="env_report"` Auslesen der Werte von **CGI-Environment-Variablen**
   ⇒ VALUE enthält eine Liste der Variablennamen
   `VALUE="REMOTE_ADDR, HTTP_USER_AGENT"`
   `REMOTE_ADDR` **IP-Adresse** des Absender-PC = übergeordneter Schlüssel für die Datenaggregation (Abb. 3)
   `HTTP_USER_AGENT` Bezeichnung des vom Probanden verwendeten **Browsers**

8. Zu `TYPE="hidden"` vgl. Anmerkung 5
   Dieses Eingabefeld dient nicht der Steuerung des CGI-Programms, sondern als untergeordneter **Schlüssel für die Datenaggregation** (vgl. Abb. 3)
   Der Wert der Variablen _LINK_ wird duch die JavaScript Function *start*, die durch den event handler onSubmit (vgl. Anmerkung 3) aufgerufen wird, aus der QUERY_STRING Variablen des CGI-Environments ausgelesen und auch darüber an das nachfolgende Dokument weitergeleitet. _LINK_ enthält für alle Formulare (Fragekomplexe), die zum selben Fragebogen gehören, eine Konstante, die durch Konkatenation zweier Informationen entsteht. Verkettet werden der **Betriebsmodus** (1: mit Protokoll = Entwickler-Modus, 2: ohne Protokoll = Anwender-Modus) und die **Tageszeit in Millisekunden** zum Zeitpunkt des Aufrufs des ersten Formulars (Fragekomplex) des Fragebogens von der Einführungsseite. Bezugsbasis für die Tageszeit ist der 1.1.1970.

9. Zu `TYPE="hidden"` vgl. Anmerkung 5
   `NAME="redirect"` ⇒ VALUE enthält URL eines nachfolgend darzustellenden Dokumentes, d.h. des **nächsten Fragekomplexes**
   VALUE wird durch den eventhandler onSubmit (vgl. Anmerkung 3), d.h. die dadurch aufgerufene JavaScript Function *start* zugewiesen, wenn als Betriebsmodus=2: Ohne Protokoll, d.h. normaler Anwendungsfall gesetzt ist. Der Betriebsmodus ist das erste Zeichen des query strings und wird bei Zugang über die Einführungsseite entweder auf 1 (mit Protokoll) oder auf 2 (ohne Protokoll) gesetzt.

10. Zu `TYPE="hidden"` vgl. Anmerkung 5
    `NAME="return_link_url"` ⇒ VALUE enthält URL eines nachfolgend darzustellenden Dokumentes, d.h. des **nächsten Fragekomplexes**
    VALUE wird durch den eventhandler onSubmit (vgl. Anmerkung 3), d.h. die dadurch aufgerufene JavaScript Function *start* zugewiesen, wenn als Betriebsmodus=1: Mit Protokoll, d.h. Beobachtung durch den Entwickler gesetzt ist. Der Betriebsmodus ist das erste Zeichen des query strings und wird bei Zugang über die Einführungsseite entweder auf 1 (mit Protokoll) oder auf 2 (ohne Protokoll) gesetzt.

11. Zu `TYPE="hidden"` vgl. Anmerkung 5
    `NAME="title"` ⇒ VALUE enthält den Text der **Überschrift eines Protokolls** (vgl. Browserdarstellung der Protokollierung)
    `VALUE="text"` wird zum Bedienungshinweis umfunktioniert
    Wird die JavaScript Function *start* mit Parameter, d.h. unter Angabe des Namens eines Nachfolge-Dokumentes aufgerufen (vgl. Anmerkung 3), lautet der

Hinweis „Zur Fortsetzung bitte auf die Grafik klicken“, d.h. das VALUE Attribut bleibt bzgl. der Vereinbarung im HTML Dokument unverändert. Ohne Parameter lautet der Text: „Vielen Dank für Ihre Mitarbeit“ und wird von der JavaScript Function gesetzt.

12. Zu `TYPE="hidden"` vgl. Anmerkung 5
    Dieses Feld wird vom CGI-Programm *FormMail* nur ausgewertet, wenn das VALUE Attribut des Feldes *return_link_url* nicht leer ist.
    `NAME="return_link_title"` ⇒ VALUE enthält **Schaltfläche** zum Aufruf des nachfolgend darzustellenden Dokumentes

    ```
    VALUE="<CENTER>
           <IMG SRC='grafik'
            ALT='Übergang zum nächsten Fragekomplex'>
           </CENTER>"
    ```

    Die *grafik* muss auf einem (aktiven) WWW Server (im Beispiel: 127.0.0.1) ab Verzeichnis *cgi-bin* gespeichert werden, d.h. in einem expliziten Pfad muss dieses Verzeichnis immer vorkommen; relative Pfade dürfen dieses Verzeichnis nicht toppen.

13. Zu `TYPE="hidden"` vgl. Anmerkung 5
    `NAME="missing_fields_redirect"` ⇒ VALUE enthält **URL eines nutzereigenen Dokumentes**, das auf fehlende notwendige Eingaben verweist
    `VALUE="URL"` z.B. *Error.htm* befindet sich im *cgi-bin* Verzeichnis des Servers

14. Zu `TYPE="hidden"` vgl. Anmerkung 5
    `NAME="required"` ⇒ VALUE enthält Namensliste von Eingabefeldern, welche vom Probanden **unbedingt auszufüllen** sind
    `VALUE="geschl, land, schubi"` Namen von Eingabefeldern des aktuellen Formulars. Es dürfen **keine formularfremden Eingabefelder** auftreten
    Wird ein Feld vom CGI-Programm *FormMail* als nicht ausgefüllt erkannt, wird das vereinbarte Dokument oder eine Standard-Fehlermeldung (in engl. Sprache) angezeigt. Die engl. Fehlermeldung konkretisiert die Meldung durch Angabe der betroffenen Variablennamen, ist damit aber nur für den Entwickler geeignet.
    Bedingte Ausfüllung und Plausibilitätstests erfordern event handling und damit assoziierten JavaScript Code.
    Ein Beispiel für einen Plausibilitätstest realisiert der event handler onBlur beim Eingabefeld *alter*;
    Ein Beispiel für Konsistenzkontrolle zeigt der event handler onBlur beim Eingabefeld *schubi*.

Die Anmerkungen 15 – 18 und 20 beziehen sich auf den spezifischen Anteil am Fragekomplex 1 des Beispiel-Fragebogens.
Alle Fragen, die ein Proband beantworten soll, müssen natürlich von dessen Browser visualisiert werden. Um die Übersichtlichkeit zu wahren, stehen insbesondere Cascaded Style Sheets (CSS) und Tabellen in HTML zur Verfügung.
Die Fragen sollten verfahrensbedingt immer in Tabellen eingepasst sein.
Auf den Einsatz von Cascaded Style Sheets wurde im Beispiel verzichtet.

15. Tabelle zur Angabe des **Geschlechts** `TYPE="radio"` verwendbar für (beliebig viele) exklusive Alternativen
    `NAME="name"` frei wählbarer Variablenname vgl. Namen von Eingabefeldern für Nutzerdaten; Variable enthält ein kategorielles Merkmal dessen Wertevorrat eine ganzzahlige Skala ist
    `VALUE="wert"` Stufe der Skala, die von diesem Radio Button repräsentiert wird

16. Tabelle zur Angabe des **Alters** in Jahren und zur evtl. Angabe der Anzahl der leiblichen **Kinder**. Die Frage nach der Anzahl der Kinder wird altersabhängig ab 16 Jahren gestellt.
    `TYPE=` wird verfahrensbedingt nicht spezifiziert; für HTML und damit für Browser gilt default `TYPE="text"`; der Programmgenerator erkennt hier auf ein numerisches Feld
    `NAME="name"` frei wählbarer Variablenname; vgl. Namen von Eingabefeldern für Nutzerdaten; Variable enthält kontinuierliche Werte
    `SIZE` und `MAXLENGTH` sichtbare und max. eingebbare Zeichen = Stellenanzahl
    Der JavaScript Code beim eventhandler **onBlur** realisiert eine Plausibilitätskontrolle[21] und die popUp-Visualisierung[22] des abhängigen Eingabefeldes (incl. der zugehörigen Fragestellung). Bei Verlegung des Fokus wird der JavaScript Code abgearbeitet, testet die Eingabe auf das Intervall (1, 99) und gibt ggf. eine Meldung aus. Dieser Test muss vor Abschicken des Formulars erneut durchgeführt werden, da nicht sicher ist, dass der Proband evtl. Fehler bei Erstanzeige korrigiert hat. (vgl. Anmerkung 19)

    Um eine strukturelle Kontrolle einer Information vorzunehmen, wäre ebenfalls hier der Script Code, zumindest ein Function Aufruf, einzufügen.

    Verzichtet der Fragebogen-Entwickler auf clientseitige Plausibilitätschecks einschl. des Tests auf gültige Zahlendarstellungen, wird das generierte Interpreter-Programm u.U. ungültige Argumente der SAS Function *INPUT* monieren und folgerichtig Missing Values zuweisen.

    Der event handler **onMouseOut**[23] reagiert bei abziehender Maus und realisiert ebenso die popUp-Visualisierung des abhängigen Eingabefeldes nebst zugehöriger Fragestellung. Dadurch erscheint das abhängige Feld ggf. in einer mehr natürlichen Abfolge bei der Ausfüllung des Formulars.

    Auch die evtl. eingegebene Anzahl der Kinder wird einer Plausibilitätskontrolle[21] unterzogen, d.h. die Eingabe wird auf Lage im Intervall (0, 12) getestet. Zusätzlich ist eine einzelne Null durch z.B. '00' zu ersetzen (vgl. Besondere

[21] Plausibilitätskontrollen schließen natürlich stets den Test auf gültige externe Zahlendarstellung ein

[22] Nur der **Internet Explorer (6.0)** bot die Möglichkeit, die Eigenschaft visibility via CSS-Klassen zu steuern:
```
.hide { visibility: hidden; }
.show { visibility: visible; } und im JavaScript:
document.all.feldname.className='hide'|'show'
```

[23] Der Browser ***Opera (6.0)*** **reagiert nicht auf mouse events** wie onMouseOut, onMouseOver, onMouseMove

Werte von Eingabefeldern). Unterbleibt die Ersetzung '0' ⇒ '00', gelangt statt '0' Missing Value in die Datenerhebung!

17. Tabelle zur Angabe des **Bundesland**es, in dem der Proband seinen Hauptwohnsitz hat;
    Wird über eine Auswahlliste mit einem <SELECT> Tag und eingeschlossenen <OPTION> Tags realisiert. Diese Liste lässt hier sinnvollerweise nur eine diskrete Auswahl zu.

18. Tabelle zur Angabe der abgeschlossenen **Schulbildung**(en);
    Wird über eine Auswahlliste mit einem <SELECT> Tag und eingeschlossenen <OPTION> Tags realisiert. Diese Liste lässt hier mehrere Auswahlen zu, was natürlich durch einen Bedienhinweis mitgeteilt werden muss und eine Erfassung in der SAS Datentabelle als Vektor mit entsprechender Komponentenanzahl bewirkt.

    Da die Liste der Optionen neben vielen Schultypen, auch *keine abgeschlossene Schulbildung* als Auswahlmöglichkeit zulässt, kann es zu widersprüchlichen Eingaben kommen, wenn die multiple Auswahl die Negativauskunft neben Positivauskünften einschließt. Folgerichtig muss bereits clientseitig eine **Konsistenzkontrolle** erfolgen, was der event handler **onBlur** im <SELECT> Tag realisiert.

Der **Submit Button** sollte, allein damit er nicht versehentlich zu früh angeklickt wird, auch das **physisch letzte grafische Eingabeelement in einem Formular** sein. Seine Gestaltung wird aber formal in allen Fragekomplexen weitestgehend gleich ausfallen, wobei das Beispiels eine unverbindliche Anregung darstellt.

19. Tabelle zur Angabe des **Submit** Buttons
    Der <SCRIPT> Tag ruft die JavaScript Function *bar*, die mit der *document.write* Methode der _ESC_ Variablen[24] als Wert den prozentualen Fortschritt bei Ausfüllung des Fragebogens zuweist und zwei Zellen der Tabelle als Fortschrittsanzeige in Form eines horizontalen Balkens, der entsprechend der Parameterangabe (Prozentwert) anteilig eingefärbt ist, generiert.

    `NAME=` ohne Angabe des NAME Attributes wird kein Wert durch das Eingabefeld übertragen
    `TYPE="submit"` kennzeichnet den Button als Submit Button, d.h. bei Anklicken wird das Formular vom CGI-Programm ausgewertet (vgl. Anmerkung 3)
    `VALUE="text"` enthält die Beschriftung des Buttons; ist ein NAME Attribut vereinbart, wird *text* als Wert der Variablen in die eMail übernommen.

    Die Verwendung des event handlers **onClick** bei einem Submit Button verursacht u.U. Probleme, weil innerhalb des <FORM> Tags Submit implizit ausgeführt werden kann, z.B. durch Drücken der **ENTER** Taste in einem Eingabefeld vom `TYPE="text"`. Daher sollten abschließende Operationen besser vom event handler **onSubmit** des <FORM> Tags ausgeführt werden. (vgl. Fragekomplex 4: Abschließende Anmerkungen).

[24] _ESC_ dient als Filter zur Erkennung von Abbrechern

Am Ende des <BODY> Tags werden noch die benötigten **Audiodateien** eingebettet: Die hier angegebene Methode funktioniert nur mit dem Internet Explorer.

20. `NAME="sound"` Referenz für Aufruf der JavaScript Methode zum Abspielen eines Sounds: document.*sound*.play()
`SRC="URL"` URL einer wav-Datei[25]; Die Datei wird am besten mit dem aufrufenden Dokument im selben Server Verzeichnis gespeichert
`AUTOSTART="false"` Das Abspiel des Sounds erfolgt per explizitem Aufruf in der JavaScript Function *popUp* (`='true'` bewirkt das Abspiel nach dem Laden des Dokumentes)
`HIDDEN="true"` Es wird keine Player-Konsole auf der Web-Seite visualisiert
`LOOP="false"` Der Sound ertönt exakt einmal (`='true'` erzeugt eine Endlosschleife[26])

Fragekomplex 2: Subjektive Erfahrungen

Sambar50/cgi-bin/Umfrage/InternetnutzungKomplex2.htm[27]

Abbildung 8: Fragekomplex 2 „Subjektive Erfahrungen“

[25] Im Beispiel wurde die Datei *Erinner.wav*, die mit dem Windows Betriebssystem geliefert wird, verwendet

[26] nur bei Verlassen der Seite verstummt der Apparat – man muss unwillkürlich an einschlägige Erfahrungen des Alten aus Frankfurt denken ...

[27] Die Datei befindet sich auf der Proceedings-CD

Fragekomplex 2 enthält eine **Filterführung**, d.h. bestätigt der Proband weder irgendeine positive noch eine negative Erfahrung, wird Fragekomplex 3 ausgelassen und sofort mit dem abschließenden Fragekomplex 4 fortgesetzt. Die Realisierung erfolgt mit JavaScript Anweisungen, die vom event handler **onSubmit** des <FORM> Tags ausgeführt werden:

```
<FORM NAME    ="komplex2"
      METHOD  ="post"
      ACTION  ="http://127.0.0.1/cgi-bin/winformmail.pl"
      onSubmit="var test=false;
                for (i=1; i<7; ++i)
                   test=document.forms[0].positiv[i].checked ||
                   document.forms[0].negativ[i].checked ||
                   test;
                if (test)
                   start('Umfrage/InternetnutzungKomplex3.htm');
                else
                   start('Umfrage/InternetnutzungKomplex4.htm')">
```

Außerdem werden die **Abhängigkeiten der Antworten** berücksichtigt, d.h. die Konsistenz der Antworten gewahrt, indem die Verneinung von positiven oder negativen Erfahrungen wechselseitig zuvor evtl. angeklickte Checkboxes konkreter positiver oder negativer Erfahrungen löscht. Auch hierzu werden JavaScript Anweisungen codiert, die je vom event handler **onClick** der Eingabefelder (Checkboxes) ausgeführt werden (Analog für die Negativerfahrungen):

```
<TR><TD ALIGN="center">
    <INPUT TYPE ="checkbox"
           NAME ="positiv"
           VALUE=+0
           onClick="for (i=1; i<7; ++i)
                     this.form.positiv[i].checked='';">
    <TD ALIGN="center">
    <INPUT TYPE="checkbox"
           NAME ="positiv"
           VALUE=1
           onClick="this.form.positiv[0].checked='';">

usw.

    <TD ALIGN="center">
    <INPUT TYPE ="checkbox"
           NAME ="positiv"
           VALUE=6
           onClick="this.form.positiv[0].checked='';">
```

Fragekomplex 3: Nutzungsart und -häufigkeit

Sambar50/cgi-bin/Umfrage/InternetnutzungKomplex3.htm[28]

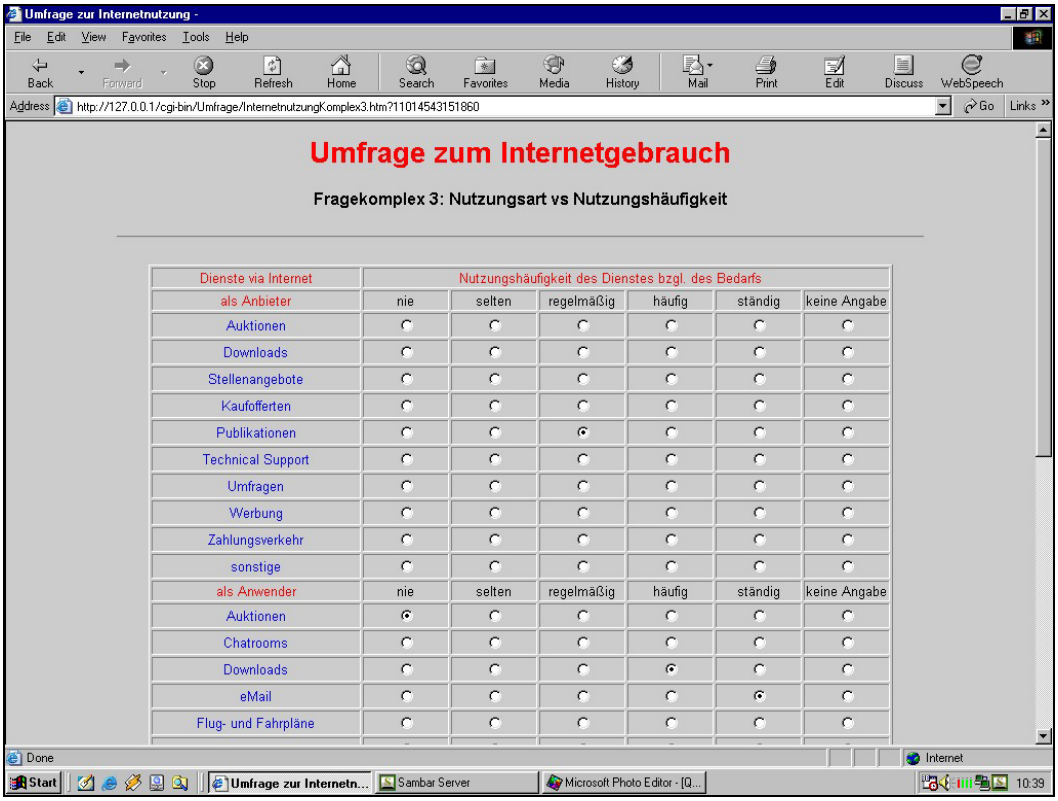

Abbildung 9: Fragekomplex 3: „Nutzungshäufigkeit je Nutzungsart“

Bei der Gestaltung von Tabellen sollte darauf geachtet werden, dass

- wenn, dann nur vertikales Scrollen erforderlich wird und
- stets eine Spaltenüberschrift sichtbar ist

Außerdem ist es günstig, wenn bei umfangreicheren Tabellen ein spalten- bzw. zeilenweises Löschen der Eingaben möglich ist, ohne dass, wie bei einem Formular-Reset-Button, gleich das ganze Formular auf den Anfangszustand gelöscht wird. Dies ist bei Radio Buttons, die stets exklusive Alternativen repräsentieren, völlig problemlos, ohne jede Script Programmierung, realisierbar.

[28] Die Datei befindet sich auf der Proceedings-CD

Fragekomplex 4: Abschließende Anmerkungen

Sambar50/cgi-bin/Umfrage/InternetnutzungKomplex4.htm[29]

Abbildung 10: Fragekomplex 4: „Abschließende und zusammenfassende Anmerkungen“

Selbst, wenn sich eine programmierte Auswertung derartiger Texte schwierig gestaltet, so ist es höflich und möglicherweise auch interessant abschließend eine Meinung der Probanden einzuholen.

Wirklich wichtig ist aber, dass in allen Zweigen des Fragebogens, die sich möglicherweise infolge von Filterführung aufgetan haben, die Fortschrittsanzeige stetig voranschreitet und mit 100% endet, da der Wert=100 als Indikator für vollständig ausgefüllte Fragebögen genutzt wird (vgl. JavaScript Function *bar.js*[28]). Somit eignet sich eine Seite, wie Fragekomplex 4, gut zur abschließenden Zusammenführung aller Zweige eines Fragebogens.

Verfahrensbedingt muss für jedes **TEXTAREA** eine **Script Anweisung** abgearbeitet werden, die im eingegebenen Text enthaltene Zeilenwechsel (NewLines) durch Leerzeichen ersetzt. Wenn kein implizites Submit, z.B. durch Drücken der **ENTER Taste**, in Eingebefeldern vom `TYPE="text"` ausgelöst werden kann, kann die Script Anweisung zur Substitution der NewLines vom event handler **onClick** des Submit Buttons ausgeführt werden (wie im Beispiel):

[29]Die Datei befindet sich auf der Proceedings-CD

```
<INPUT TYPE ="submit"
       VALUE="beenden"
       onClick="this.form.note.value=this.form.note.value.replace
                (/\x0D\x0A/g, ' ');">
```

Sind implizite Submits möglich, muss die Script Anweisung vom event handler **onSubmit** des <FORM> Tags in leicht modifizierter Codierung ausgeführt werden:

```
onSubmit="document.forms[0].note.value=
          document.forms[0].note.value.replace(/\x0D\x0A/g, ' ');"
```

Sachlich korrekt kann man die Script Anweisung auch beim <TEXTAREA> Tag vom event handler **onBlur**, d.h bei Abziehen des Fokus, ausführen lassen. In diesem Fall bleibt die Veränderung aber für den Probanden bis zum Submit auf dem Bildschirm sichtbar und führt möglicherweise zu Irritationen.

2.2.3 JavaScript Quelltexte

JavaScript ist eine casesensitive Sprache.
Script Code wird vom Programmgenerator nicht berücksichtigt und wird dann zu einem Problem, wenn mit ihm dynamisch Definitionen von Eingabefeldern und deren Wertevorräten, etwa mit der *document.write* Methode, zur Ausführzeit vorgenommen werden. Solange der Script Code nur zur Ablaufsteuerung und Eingabeüberwachung genutzt wird, dürften keine Probleme entstehen. Letztlich kann eine derart positive Aussage aber nur unter Vorbehalt gemacht werden, da die Phantasie der Anwender einer Programmiersprache nie zu unterschätzen ist – und das ist nun auch wirklich gut so!

Es gibt viele Möglichkeiten der Einfügung von Script Anweisungen:

- komplett und direkt beim event handler, d.h. ohne speziellen <SCRIPT> Tag
- nur mit Funktionsaufruf beim event handler, was einen zugehörigen <SCRIPT> Tag erfordert,
- explizit in einem <SCRIPT> Tag oder
- im Kopf des Dokumentes in einem <SCRIPT> Tag via SRC Attribut als zur Ausführzeit vom Server zu ladender Code.

Steuerung der sequentiellen Abfolge der Formulare

Nur wenn die JavaScript Function *start* mit Parameter, d.h. unter Angabe des Namens eines Nachfolge-Dokumentes (u.a. zur **Filterführung**) aufgerufen wird, wird je nach Betriebsmodus

- mode=1 ⇒ mit Protokoll: das VALUE Attribut von *return_link_url* oder
- mode=2 ⇒ ohne Protokoll: das VALUE Attribut von *redirect*

auf einen String gesetzt, der durch Konkatenation von

- Parameter,
- Fragezeichen,
- Betriebsmodus und
- Tageszeit in Millisekunden entsteht.

Man erhält so eine URL mit **QUERY_STRING** und rettet den untergeordneten Schlüssel für die Datenaggregation über verschiedene Formulare als Konstante.

Sambar50/cgi-bin/Umfrage/start.js[30]

```
function start(next)
   {
    var today  = new Date();
    var ms     = today.getTime();
    var mode   = new String("2");
    var url    = window.location.search;
    if (url.length)
    {
     mode= url.substr(1,1);
     if (url.length>2) ms= url.substring(2,url.length);
    }
    document.forms[0]._LINK_.value= mode+ms;
    if (mode == "1")     // mit Protokoll (für Entwickler)
    {
     if (next.length) document.forms[0].return_link_url.value =
                        next+"?"+mode+ms;
         else         document.forms[0].title.value           =
                        "Vielen Dank f&uuml;r Ihre Mitarbeit";
    }
    if (mode == "2")     // ohne Protokoll (für Anwender)
    {
     if (next.length) document.forms[0].redirect.value        =
                        next+"?"+mode+ms;
         else         document.forms[0].redirect.value        =
                        "Thanks.htm";
    }
    return;
   }
```

Plausibilitätskontrolle

Die Function *checkInt* dient der Prüfung ob eine positive ganze Zahl vorliegt, die sich innerhalb eines Intervalls befindet, dessen Grenzen *min* und *max* vorgeben werden und zum Intervall gehören.

[30]Die Datei befindet sich auf der Proceedings-CD

Die Function ist zweimal aufzurufen:

- vom event handler **onBlur** beim Eingabefeld daselbst
  hierdurch wird geprüft und ggf. eine Korrekturaufforderung veranlasst,
- vom event handler **onSubmit** beim <FORM> Tag
  hierdurch wird die Prüfung einschl. der Korrekturaufforderung wiederholt. Gleichzeitig wird ggf. der Rückkehrcode der Function auf *false* gesetzt, was eine Übergabe der Formulardaten an das CGI-Programm *FormMail* und damit den Datenversand per eMail unterbindet.

Würde der zweite Aufruf der Plausibilitätskontrolle vom event handler **onClick** des Submit Buttons ausgeführt, bekäme man u.U. Probleme, sobald Submit implizit, z.B. durch Drücken der ENTER Taste in einem Eingabefeld vom `TYPE="text"`, ausgelöst wird.

Bei implizit ausgelöstem Submit wird kein event handler onClick im Submit Button aktiviert.

Sambar50/cgi-bin/Umfrage/checkInt.js[31]

```
function checkInt(item, min, max)
{
if (item.value.length>0 &&                  // any data?
   (isNaN(item.value)                    || // valid number?
    item.value != parseInt(item.value)|| // integer?
    item.value<min                       || // less than minimum?
    item.value>max))                        // greater than maximum?
    {
     alert ("Bitte geben Sie als " + item.name.toUpperCase() +
            " eine ganze Zahl zwischen
            " + min + " und " + max + " ein");
     returnVal = false;
    }
  else returnVal = true;
  return  returnVal;
}
```

Fortschrittsanzeige (horizontaler Balken)

Die Fortschrittsanzeige ist hier als horizontales Balkendiagramm mit der *document.write* Methode realisiert.

Sie zeigt dem Probanden, wie weit er nach Abschicken des aktuellen Fragekomplexes prozentual mit der Bearbeitung des gesamten Fragebogens vorangekommen ist und

[31] Die Datei befindet sich auf der Proceedings-CD

verhindert **unnötige Abbrüche**, die aus aufkeimender Ungeduld des Befragten resultieren.

Außerdem wird der prozentuale Fortschritt als Wert der für den Probanden unsichtbaren Variablen **_ESC_** mit der eMail, die von dem Fragekomplex veranlasst wird, übertragen. Die Variable _ESC_ ist eine der technischen Variablen, die jedem Fragebogen automatisch hinzugefügt werden. Gilt in einem Fragebogen (Zeile der SAS Datentabelle) _ESC_<100, so wird dies von einem unvollständig ausgefülltem Fragebogen verursacht und somit bildet _ESC_ ein Filter zu sehr einfachen **Erkennung der Abbrecher**.

Sambar50/cgi-bin/Umfrage/bar.js[32]

```
function bar(widthPct)
{
 document.write
   ("<TD><INPUT NAME='_ESC_' TYPE='hidden' VALUE='" + widthPct +
    "'><TD ALIGN='left' BGCOLOR='#0000FF' WIDTH='" +
    eval(6*widthPct) + "'><FONT COLOR='white'>" + widthPct +
    "%</FONT>");
 if (widthPct<100)
 document.write
   ("<TD BGCOLOR='#FEFEFE' WIDTH='" + eval(600-6*widthPct) +
    "'> ");
 return;
}
```

2.2.4 Eingabefelder zur Steuerung des CGI-Programms

Bei den Eingabefeldern in HTML Formularen sind zwei Gruppen zu unterscheiden:

- Felder zur Übertragung von **Metadaten** zur Steuerung des CGI-Programms *FormMail* (vgl. Tab. 4)
  Sie sind stets vom `TYPE="hidden"` und übertragen (direkt) keine Daten in die SAS Datentabelle.

- Felder zur Übertragung von **Nutzereingaben** (vgl. Namen von Eingabefeldern für Nutzerdaten)
  Die Schreibweise der Namen wird mit erster Notation festgelegt.
  Die Namen der Eingabefelder bilden zugleich die Namen der SAS Variablen in der erzeugten Datentabelle.

[32]Die Datei befindet sich auf der Proceedings-CD

Tabelle 4: Für die Steuerung des CGI-Programms FormMail reservierte Namen von Eingabefeldern (vgl. `www.worldwidemart.com/scripts/formmail.shtml`)

NAME=[33]	**Vereinbarung**[34]	**VALUE=**
recipient	notwendig	eMail-Adresse des Empfängers
env_report	erforderlich	Liste von Namen von CGI-Environment-Variablen[33]
redirect	erforderlich	URL des Folge-Dokumentes
subject	empfohlen	subject-text
realname[35]	eher vermeiden	Klarname des Absenders
email	eher vermeiden	eMail-Adresse des Absenders
required	problemabhängig	Liste von Variablennamen
missings_fields_redirect	problemabhängig	URL eines Dokumentes, das anstelle des defaults ausgegeben wird, wenn zu *required* Variablen Wertzuweisungen fehlen
sort[36]	optional	Anordnungsvorschrift der Felder in der eMail
print_config	optional	Liste von Variablennamen (aus Tab. 4)
print_blank_fields	problemabhängig	**1**
Alle folgenden Angaben[37] gestalten die Protokoll-Seite(n) (vgl. Abb. 11)		
return_link_url	empfohlen	URL des Folge-Dokumentes
title	empfohlen	Überschrift (Bedienungshinweis)
return_link_title	empfohlen	Schaltfläche für Aufruf des Folge-Dokumentes (günstig: Grafik)
background[36]	eher vermeiden	URL eines Hintergrund-Images; alternativ zu *bgcolor*
bgcolor[36]	eher vermeiden	Hintergrundfarbe; alternativ zu *background*
text_color[36]	eher vermeiden	Textfarbe
link_color[36]	eher vermeiden	Farbe für Links
vlink_color[36]	eher vermeiden	Farbe für besuchte Links
alink_color[36]	eher vermeiden	Farbe für aktive Links (Links mit Fokus)

[33] Die Notation der Namen ist casesensitiv.

[34] Aus der Sicht von *FormMail* ist nur die Angabe *recipient* notwendig und alle anderen sind optional

[35] Feld ist nur zu nutzen, wenn der Proband selbst zustimmt die Anonymität aufzuheben; Durch Abschreckung gering motivierter Personen bei Abfrage von Name und eMail-Adresse („Hohe-Hürde-Methode“, JANETZKO, 1999, S.150) kann man die Teilnehmermotivation und damit die Datenqualität verbessern

[36] Verfahrensneutrales Feld; beschäftigt FormMail, hat aber keinen inhaltlichen Einfluss

[37] Im Beispiel nur wirksam, wenn Fragebogen im Entwickler-Modus (über Geheimeingang) ausgefülllt wird

2.2.5 FormMail-Protokoll zum ersten Fragekomplex

Die folgende Abb. 11 zeigt eine zusätzliche Protokoll-Seite, die zu jedem Fragekomplex (HTML Formular) ausgegeben wird, aber nur erscheint, wenn der erste Fragekomplex über den speziellen Eingang des Entwicklers (vgl. Die Einführungseite) aufgesucht wurde, d.h. der Fragebogen im Entwickler-Modus ausgefüllt wird.

Der Inhalt der eMail, die das CGI-Programm sendet, unterscheidet sich im Entwickler-Modus nicht vom Anwender-Modus, so dass ein zufälliger Betrieb im Entwickler-Modus durch Anwender ebenso korrekte Daten liefert.

Die konkrete Auswahl eines Betriebsmodus bedarf jedoch der entsprechenden Ansteuerung des CGI-Programms FormMail (vgl. Anmerkungen 9 - 12 zum HTML Quelltext des ersten Fragekomplexes).

Abbildung 11: FormMail-Protokoll zum ersten Fragekomplex

2.2.6 Das eMail Format

Die nachfolgende Abb. 12 zeigt die Gestaltung und den Inhalt einer eMail, die vom CGI-Programm *FormMail* im Ergebnis der Verarbeitung eines HTML Formulars versendet wurde.

Für den Programmgenerator gilt es, nun ein Interpreter-Programm zu erzeugen, das aus eben diesen eMails eine SAS Datentabelle aggregiert (vgl. Abb. 3).

Als Mail-Tool wurde EUDORA 5.1 eingesetzt.

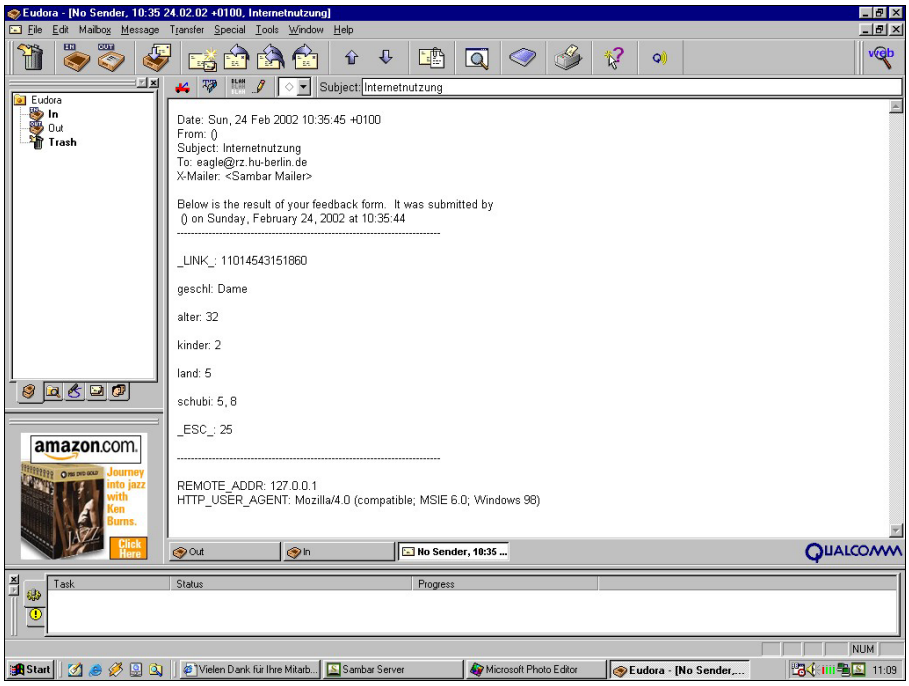

Abbildung 12: eMail mit Antworten zum Fragekomplex 1

2.2.7 Installation und Aufruf des Beispiels

Zunächst muss in den 4 HTML Dokumenten mit den Fragekomplexen *InternetnutzungKomplex1.htm – InternetnutzungKomplex4.htm*[38] die **eMail-Adresse des Empfängers** im Eingabefeld *recipient* ersetzt werden.

Danach unterscheidet sich die Installation des Beispiels von der Installation eines produktiven Fragebogens nur in der Wahl des Web-Servers und des verwendeten CGI-Programms:

- Das Beispiel wird, wie ein Fragebogen in der **Entwicklungsphase**, mit allen HTML Dokumenten, Script Programmen, Images, Sounds und anderen erforderlichen Komponenten im bzw. ab dem Verzeichnis *cgi-bin* des Sambar Servers **lokal auf dem PC** gespeichert. Der Sambar Server muss als WWW Server und Mail Proxy Server installiert und aktiv sein. Als CGI-Programm wird ***winformmail.pl*** eingesetzt (bzw. belassen).
- Ein ausgetesteter Fragebogen wird für die **Felddauer** analog auf einen produktiven **Web-Server** unter Verwendung des CGI-Programms ***formmail.pl***[39] gespeichert. Hier sind aber Zugriffsrechte zu beantragen, Verhandlungen mit Administratoren zu führen ...

[38] Die Dateien befinden sich auf der Proceedings CD im Verzeichnis Sambar50/cgi-bin/Umfrage

[39] Die Datei befindet sich im Web unter `http://www.worldwidemart.com/scripts/formmail.shtml`

- Beide Änderungen betreffen ausschl. die <FORM> Tags. Keine andere Modifikation sollte an den HTML Dokumenten beim Übergang von einer Testversion auf dem PC zu einer produktiven Version auf dem Web-Server vorgenommen werden. Der Programmgenerator erzeugt dann von beiden Versionen identische Interpreter-Programme.

Das Beispiel kann von einem Browser durch Eingabe der URL aufgerufen werden. Z.B. in der Testumgebung mit dem Sambar Server:
http://127.0.0.1/cgi-bin/Umfrage/InternetnutzungCover.htm[40]

3 Arbeitsweise des Programmgenerators

3.1 Allgemeine Restriktionen

- Alle HTML Dokumente sollten der Syntax von HTML 4.0 entsprechen, was vorab durch HTML Validatoren (vgl. `http://www.htmlvalidator.com/lite`, `http://validator.w3.org/file-upload.html`) geprüft bzw. recht kompakt bei STEYER, 1997 nachgelesen werden kann.
- Alle HTML Dokumente müssen in ISO Latin 8859-1 codiert sein. Nur die zu diesem Code gehörigen HTML Entities werden erkannt. (*Anhang/Code.html*[40])
- Als Mail User Agent kann nur das CGI-Programm *formmail.pl*[41] mit der `METHOD="post"` angewendet werden. Für die Test- und Entwicklungsumgebung unter WINDOWS ist *winformmail.pl* verfügbar.
- Script Code wird vom Programmgenerator nicht berücksichtigt.
- JavaScript Entities sind unzulässig.
- Server Side Includes **werden vom Programmgenerator nicht berücksichtigt**.
- Die Formulare sollten weitestgehend als **Tabellen** angelegt werden.
- Schachtelung von Formularen ist unzuläsig.

[40] Die Datei befindet sich auf der Proceedings-CD
[41] Die Datei befindet sich im Web unter
`http://www.worldwidemart.com/scripts/formmail.shtml`

3.2 Umsetzung der Attribute von HTML nach SAS

3.2.1 Besondere Werte von Eingabefeldern

Bei den Eingabefeldern sind zwei Gruppen zu unterscheiden:

- Felder zur Übertragung von Metadaten zur Steuerung des CGI-Programms FormMail (vgl. Tab. 4);
  Sie sind stets vom `TYPE="hidden"` und übertragen (direkt) keine Daten in die SAS Datentabelle
- Felder zur Übertragung von Nutzereingaben
  Die Namen der Eingabefelder bilden zugleich die Namen der SAS Variablen in der erzeugten Datentabelle.

Die Felder zur Übertragung von Nutzerdaten lassen sich unterteilen in Felder:

- mit vorgegebenem Wertevorrat (Auswahllisten, Radiobuttons, Checkboxes),
- mit variablem Inhalt (Nutzereingaben) und
- mit programmerzeugtem Inhalt (JavaScript)

Zero

Eingabefelder dürfen **keine einzelne Null** enthalten, weder als freie Eingabe noch als VALUE Attribut. Dieser Wert wird vom CGI-Programm nicht übermittelt, was dazu führt, dass der Variablen in der SAS Datentabelle, anstelle des richtigen Wertes Null, fälschlich Missing Value zugewiesen wird. Der Effekt kann natürlich bewusst ausgenutzt werden, um Missing Values zu erhalten. Ist dies unerwünscht, kann bei einem vorgegebenen Wertevorrat oder bei Zuweisung mit Script Anweisungen, als VALUE Attribut `"00", "+0", "-0"` u.ä. codiert werden. Numerische Nutzereingaben müssen einer Plausibilitätskontrolle unterzogen werden: Anschließend kann die singuläre Null mit einer Script Anweisung, die der eventhandler **onBlur** ausführt, modifiziert werden:

```
onBlur="if (this.form.feldname.value == '0')
                this.form.feldname.value = '+0';"
```

Beim Netscape Navigator muss für diese Wertzuweisung das **MAXLENGTH** Attribut, wenn es denn explizit angegeben ist, größer 1 sein! Bei Internet Explorer und Opera spielt das Attribut in diesem Zusammenhang keine Rolle.

NewLines

TextAreas werden vom Nutzer frei formatiert eingegeben und können u.a. NewLines (`'0D0A'X`) enthalten, die aber vor Aufruf des CGI-Programms mit einer Script Anweisung je gegen ein Leerzeichen auszutauschen sind (vgl. Fragekomplex 4: Abschließende Anmerkungen).

3.2.2 Namen von Eingabefeldern für Nutzerdaten

Die Namen der Eingabefelder bilden zugleich die Namen der Variablen in der erzeugten SAS Datentabelle. Die Schreibweise der Namen bzgl. Groß- und Kleinschreibung ist frei wählbar, darf aber im HTML Dokument nicht mehr variieren.
Reservierte Namen sind:

- alle Namen mit einem führenden Underline,
- die Namen der konfigurierenden Variablen des CGI-Programms (vgl. Tab. 4)
- die Namen der SAS Funktionen DIM, BOR, REVERSE, TRIM, SUBSTR, INPUT, LEFT

Wenn der Name eines Eingabefeldes mehrfach im selben <FORM> Tag auftritt, so muss das TYPE Attribut in allen Fällen gleich sein. Dann wird (mit Ausnahme von `TYPE="radio"`) ein eindimensionales Feld definiert, dessen Dimension von der Anzahl der multiplen Auswahlmöglichkeiten bzw. von der Zahl der gleichnamigen Eingabefelder abhängt. Es ist zu beachten, dass die Namen der Feldelemente aus dem Feldnamen, der um den dezimalen Elemente-Index verlängert wird, bestehen und für kein Element die maximal zulässige Namenslänge überschreiten dürfen.

Gleiche Namen von Eingabefeldern in verschiedenen <FORM> Tags sind unzulässig. Einige Beschränkungen resultieren aus der verwendeten SAS Version (vgl. Tab. 5)

Tabelle 5: Aus der verwendeten SAS Version resultierende Beschränkungen

	SAS 6	**SAS 8**
max. Namenslänge	8	32
Zeichenvorrat von Namen	Underline, engl. Alphabet, Ziffern	beliebig[42]
max. Länge von Labels	40	256
max. Länge von Zeichenketten	200	32767

3.2.3 Datentypen und -längen

HTML unterscheidet im <INPUT> Tag durch Angabe des TYPE Attributes 10 verschiedene Datentypen. Hinzu kommen noch die diskrete oder multiple Auswahlliste, die mit den <SELECT> und <OPTION> Tags realisiert werden und ein spezieller <TEXTAREA> Tag für umfangreiche, mehrzeilige Texte.

SAS unterscheidet in den Datentabellen nur zwei Datentypen: Numerische und Character Variable. Somit ist keine 1:1 Umsetzung durchführbar, sondern es muss eine Umsetzung nach bestimmten Regeln erfolgen (vgl. Tab. 6)

[42] Die Beschränkung des Zeichenvorrats wird in SAS 8 mit Hilfe von `OPTIONS VALIDVARNAME=ANY;` und unter Verwendung von Namensliteralen umgangen, aber nicht von allen SAS Komponenten unterstützt und sollte deshalb vorerst gemieden werden.

Tabelle 6: Eingabedaten und deren Attribute in der SAS Datentabelle

Eingabedaten	**HTML**	**SAS Variable**		
		Typ	*Länge*	*Wert(e)*
	<INPUT TYPE=...>			
reelle Zahlen	ohne TYPE Attribut	_numeric_	4[43] 8[45]	quantitative Variable[44]
Text	text	_character_	[46]	[44]
exklusive Alternativen	radio	_numeric_	3	kategorielle Variable
dichotome Merkmale	checkbox	_numeric_	3	0, 1
Passworte	password	_character_	[46]	[44]
Steuerungs-informationen	hidden	_character_	[47]	[44]
Point & Click-Koordinaten	image	_numeric_	4	Vektor mit 2 Komponenten (x, y) bzgl. Image
	submit	erzeugen keine Variable		
	reset			
	button			
	file			
	<SELECT>			
diskrete Auswahl	ohne MULTIPLE Attribut	_numeric_	3	kategorielle Variable $1, \ldots, n$ (n= Optionsanzahl)
multiple Auswahl	mit MULTIPLE Attribut	_numeric_	3	Vektor dichotomer Variable 0, 1
	<TEXTAREA>			
freie umfang-reichere Texte		_character_	[48]	[49]

[43] Wenn MAXLENGTH<7

[44] Zuweisung wie vom Anwender eingegeben. Man muss aber beachten, dass die einzelne Null `VALUE="0"` nicht übertragen wird. Ist die Null als Wert aus dem Formular zu übertragen, so muss sie ein Vorzeichen tragen oder z.B. verdoppelt werden (`"00"`).

[45] Wenn MAXLENGTH>6 oder MAXLENGTH nicht spezifiziert wurde.

[46] = MAXLENGTH Attribut. Fehlt die Angabe von MAXLENGTH im <INPUT> Tag, so wird das Minimum der globalen Macrovariablen TextArea und der maximalen Länge von Character Variablen, die die SAS Version zulässt, angenommen.

[47] = Anzahl der Zeichen des VALUE Attributes, d.h. wird VALUE durch Script Programmierung zugewiesen, so muss im HTML Dokument ein Platzhalter entsprechender Länge vereinbart werden

[48] = Minimum des mit der globalen Macro Variablen TextArea vereinbarten Wertes und der maximalen Länge von Character Variablen, die die SAS Version zulässt. Ist der Wert der Macro Variablen größer als die maximal zulässige Variablenlänge, so wird ein Vektor von Zeichenketten gebildet dessen Komponenten je die maximale Variablenlänge besitzt. Wird

Die SAS Datentabellen werden mit der Dataset Option **COMPRESS=YES** angelegt, da durch die Zeichenketten meist und sehr schnell überdimensionierte Längenattribute auftreten. In der SAS Version 6 ist damit eine Weiterverarbeitung im Direktzugriff ausgeschlossen. Diese Einschränkung besteht bei SAS Version 8 nicht mehr.

3.2.4 Label und Formate

Explizite Label und Formate für Eingabefelder kennt HTML nicht. Sie können für die SAS Variablen daher nur dem Kontext entnommen werden, was leider nicht immer zufriedenstellend gelingen kann und folglich die Hauptursache für ein nachträgliches Editieren des generierten Interpreter-Programms darstellt. Dennoch liefert das Generator Programm eine gute Ausgangsbasis, besonders, wenn die Formulare die Fragen in Tabellenform in <TABLE> Tags präsentieren.

Radiobuttons

Variablen, die in der SAS Datentabelle mit Radiobuttons gebildet werden, enthalten **kategorielle** Merkmale deren Wertevorrat der Anzahl der Radiobuttons entspricht, die sich über ein gleiches NAME Attribut auf die Variable beziehen.

Missing Values entstehen, wenn

- kein einziger Radiobutton, der zu dem Merkmal gehört angeklickt wird,
- ein Radiobutton angeklickt wird, dem `VALUE="0"` zugeordnet ist,
- die Variable aus einem, z.B. infolge von Filterführung, nicht aktivierten Fragekomplex stammt.

Die implizite **Voreinstellung** von Variablen, die mit Radiobuttons kreiert werden, ist 0 (not checked). Sie kann explizit durch Codierung des HTML Attributes CHECKED geändert werden.

Betrachtet man z.B. die Abfrage des Geschlechtes:

---

der Macro Variablen ein kleinerer Wert als die Textlänge des vom Anwender tatsächlich eingegebenen Textes zugewiesen, so wird der restliche Text nicht in die SAS Datentabelle übertragen.

[49] In Texten, die mit dem <TEXTAREA> Tag eingegeben werden, müssen vor ihrer Übertragung alle `'0D0A'X` (NewLine) durch Leerzeichen ersetzt werden, was Script Programmierung erfordert (vgl. Fragekomplex 4). Die Ursache liegt in der Formatierung der eMails durch das CGI-Programm FormMail.

```
<TABLE>
<TR><TD> 
    <TD>weiblich
    <TD>m&auml;nnlich
<TR><TD>Bitte geben Sie Ihr Geschlecht an:
    <TD><INPUT TYPE ="radio" NAME ="geschl" VALUE="Dame">
    <TD><INPUT TYPE ="radio" NAME ="geschl" VALUE="Herr">
</TABLE>
```

Veranschaulicht man sich die Tabelle, erkennt man, dass das Eingabefeld (=Variable) *geschl* zweimal in der zweiten Tabellenzeile auftritt, also einen Zeilenvektor von im Beispiel 2 Komponenten bildet:

	weiblich	männlich
Bitte geben Sie Ihr Geschlecht an:	bei Click: geschl=Dame	bei Click: geschl=Herr

Aus dieser Konstellation ergibt sich:

1. Die Eingabedaten sind vom `TYPE="radio"`.
   Somit ist ein **kategorielles Merkmal** zu bilden:
   Wenn `VALUE="Dame"` dann *geschl*=1;
   Wenn `VALUE="Herr"` dann *geschl*=2
   oder allgemein: *geschl*=Index von VALUE im Wertevorrat.
   Die Variable *geschl* in der SAS Datentabelle besitzt nur die Werte der Skala (1 oder 2), d.h. die Variable enthält (zufällig) ein **dichotomes Merkmal**.

2. Die auszugebenden Werte werden durch ein nutzerdefiniertes VALUE Format erzeugt. Dabei wird bei einem Zeilenvektor die Spaltenüberschrift (bei einem Spaltenvektor die Zeilenbeschriftung) als **Formatted Value** genutzt. Der Name des Formates wird durch Zählung gebildet, z.B.:

   ```
   PROC FORMAT   LIBRARY=LIBRARY;
        VALUE    _00001_     1 = "weiblich"
                             2 = "männlich";
   ```

   Der ursprünglich vom HTML Formular gelieferte Wert für *geschl* (Dame oder Herr) ist in der SAS Datentabelle nicht mehr präsent.

3. Als **LABEL** der Variablen wird bei einem Zeilenvektor die Zeilenbeschriftung (und bei einem Spaltenvektor die Spaltenüberschrift) verwendet, d.h. hier erzeugt der Generator die Anweisung

   ```
   LABEL    geschl  = "Bitte geben Sie Ihr Geschlecht an:";
   ```

Checkboxes

Variablen der SAS Datentabelle, die mit Checkbox gebildet werden, enthalten stets **dichotome Merkmale**, d.h sie haben jeweils nur zwei Werte (1 oder 0), die dem Zustand der Checkbox (checked oder not checked) entsprechen.

Missing Values entstehen nur, wenn die Variable aus einem, z.B. infolge von Filterführung, nicht aktivierten Fragekomplex stammt.

Die implizite **Voreinstellung** aller Variablen, die mit Checkboxes kreiert werden, ist 0 (not checked). Sie kann explizit durch Codierung des HTML Attributes CHECKED geändert werden.

Betrachtet man z.B. die Abfrage der persönlichen Erfahrungen mit dem Internet:

```
<TABLE>
   <TR><TD COLSPAN="7">Positive Erfahrungen mit dem Internet:
   <TR><TD>keine
       <TD>zeitsparend
       <TD>preisg&uuml;nstig
       <TD>lustig
       <TD>informativ
       <TD>motivierend
       <TD>andere
   <TR><TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=+0>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=1>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=2>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=3>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=4>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=5>
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=6>
</TABLE>
```

Das Vorzeichen bei `VALUE=+0` ist nötig, um den Wert `"+0"` vom CGI-Programm zu übertragen. Eine einzelne vorzeichenlose 0 wird nicht übertragen. (vgl. Besondere Werte von Eingabefeldern)

Veranschaulicht man sich die Tabelle, erkennt man, dass das Eingabefeld (=Variable) *positiv* siebenmal in der dritten Tabellenzeile auftritt, also einen Zeilenvektor von im Beispiel 7 Komponenten bildet:

Positive Erfahrungen mit dem Internet:	
keine	bei Click: `positiv[0] =on`
zeitsparend	bei Click: `positiv[1] =on`
preisgünstig	bei Click: `positiv[2] =on`
lustig	bei Click: `positiv[3] =on`
informativ	bei Click: `positiv[4] =on`
motivierend	bei Click: `positiv[5] =on`
andere	bei Click: `positiv[6] =on`

Wie man leicht sieht, kann der Befragte durch gleichzeitiges Anklicken der Checkbox „keine“ und mindestens einer beliebigen anderen Checkbox **widersprüchliche** und damit nicht mehr korrekt verwertbare Antworten abgeben. Durch zusätzliche Script Anweisungen, ist die **Konsistenz** ist recht einfach zu erhalten:

```
<TABLE>
   <TR><TD COLSPAN="7">Positive Erfahrungen mit dem Internet:
   <TR><TD>keine
       <TD>zeitsparend
       <TD>preisg&uuml;nstig
       <TD>lustig
       <TD>informativ
       <TD>motivierend
       <TD>andere
   <TR><TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=+0
                  onClick="for (i=1; i<7; ++i)
                           this.form.positiv[i].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=1
                  onClick="this.form.positiv[0].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=2
                  onClick="this.form.positiv[0].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=3
                  onClick="this.form.positiv[0].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=4
                  onClick="this.form.positiv[0].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=5
                  onClick="this.form.positiv[0].checked='';">
       <TD><INPUT TYPE="checkbox" NAME="positiv" VALUE=6
                  onClick="this.form.positiv[0].checked='';">
</TABLE>
```

Aus dieser Konstellation ergibt sich:

1. Die Eingabedaten sind vom `TYPE="checkbox"`. Der Name des Eingabefeldes wird mehrfach verwendet. Somit wird ein Vektor **dichotomer Merkmale** gebildet:
   Wenn `VALUE="+0"` dann *positiv1*=1 sonst *positiv1*=0
   ...
   Wenn `VALUE="6"` dann *positiv7*=1 sonst *positiv7*=0
   Alle Variablen, die Komponenten des Vektors *positiv* in der SAS Datentabelle sind, besitzen nur die Werte 0 oder 1

2. Nutzerdefinierte VALUE Formate werden für dichotome Variable, die z.B. durch Checkbox gebildet werden, nicht erzeugt, weil schlecht zu erraten ist, wie die formatted values aussehen sollten (ja/nein, yes/no, ein/aus, gesund/krank usw.)

3. Als **LABEL** der Variablen wird bei einem Zeilenvektor die Zeilenbeschriftung verkettet mit der Spaltenüberschrift (und bei einem Spaltenvektor die Spaltenüberschrift verkettet mit der Zeilenbeschriftung) verwendet. Da hier im Beispiel keine Zeilenbeschriftung existiert, wirkt eine Standardannahme, die kein restlos befriedigendes Resultat erbringt und nachträgliches Editieren erfordert:

```
LABEL    positiv1  = "keine / keine"
         positiv2  = "keine / zeitsparend"
         positiv3  = "keine / preisgünstig"
         positiv4  = "keine / lustig"
         positiv5  = "keine / informativ"
         positiv6  = "keine / motivierend"
         positiv7  = "keine / andere";
```

   Mit Hilfe eines CHANGE Kommandos über dem markierten Bereich kann leicht nachgearbeitet werden:

```
CHANGE 'keine /' 'positiv /' all
```

Listen mit diskreter Auswahl

Listen, die nur eine diskrete Auswahl zulassen, liefern ein **kategorielles Merkmal** dessen Skala die Werte von 1 bis n einschließt, wobei n die Anzahl der Optionen in der Auswahlliste ist.

Es gibt stets eine **voreingestellt selektierte Option**. Diese kann

- explizit mit dem SELECTED Attribut benannt werden oder
- implizit die erste Option aus der Liste sein (HTML Funktionalität).

Eine **Differenzierung zwischen impliziter oder expliziter Auswahl** einer Option anhand der per eMail erhaltenen Antwort ist nicht möglich. D.h. möchte der Interviewer unterscheiden, ob eine Antwort durch Mausklick des Probanden oder durch Auslassen der Beantwortung der Frage, also durch Voreinstellung entstand, so muss die Voreinstellung auf eine Alternative verweisen, die nicht zum eigentlichen Wertevorrat gehört (z. B. Leerfeld), bei der diese Unterscheidung bedeutungslos ist.

Missing Value wird der Variablen in der SAS Datentabelle nur dann zugewiesen, wenn

- eine Alternative angeklickt wird, der das Attribut `VALUE="0"` zugeordnet ist,
- sich die Auswahlliste in einem, z.B. infolge von Filterführung, nicht aktivierten Fragekomplex befindet.

Zum Beispiel die Frage nach dem Bundesland des Wohnortes:

```
<TABLE>
   <TR><TD>Hauptwohnsitz in Bundesland:
       <TD><SELECT NAME ="land">
              <OPTION VALUE=0>
              <OPTION VALUE=1>Schleswig-Holstein
              <OPTION VALUE=2>Freie und Hansestadt Hamburg
              <OPTION VALUE=3>Freie Hansestadt Bremen
              <OPTION VALUE=4>Niedersachsen
              <OPTION VALUE=5>Nordrhein-Westfalen
              <OPTION VALUE=6>Hessen
              <OPTION VALUE=7>Rheinland-Pfalz
              <OPTION VALUE=8>Baden-W&uuml;rttemberg
              <OPTION VALUE=9>Freistaat Bayern
              <OPTION VALUE=10>Saarland
              <OPTION VALUE=11>Bundeshauptstadt Berlin
              <OPTION VALUE=12>Brandenburg
              <OPTION VALUE=13>Mecklenburg-Vorpommern
              <OPTION VALUE=14>Sachsen-Anhalt
              <OPTION VALUE=15>Freistaat Sachsen
              <OPTION VALUE=16>Freistaat Th&uuml;ringen
              <OPTION VALUE=17>nicht in Deutschland
          </SELECT>
</TABLE>
```

Diese Tabelle hat nur eine Zeile mit zwei Spalten:

Hauptwohnsitz in Bundesland:	hier befindet sich die Auswahlliste; der übertragene Wert ist das VALUE Attribut oder, falls dies nicht spezifiziert wurde, der selektierte Text

Aus dieser Konstellation ergibt sich:

1. Die Eingabedaten stammen aus einer Liste von Werten. Es gibt immer nur einen ausgewählten Wert. Somit wird ein **kategorielles Merkmal** gebildet:
   Wenn `VALUE="0"` wird nicht übertragen; daher *land*=. (missing)
   Wenn `VALUE="1"` dann *land*=2
   Wenn `VALUE="2"` dann *land*=3 usw.
   Die Variable *land* in der SAS Datentabelle besitzt die Werte einer Skala von 2 bis n, wobei n die Anzahl der Alternativen in der Auswahlliste ist.

   Missing Values treten im Beispiel nicht auf, da sich die Variable *land* unter den als notwendig anzugebenden (*required*) Variablen befindet und so `VALUE="0"` (keine Übertragung) vom CGI-Programm als Fehler moniert wird.

2. Die auszugebenden Werte werden durch ein nutzerdefiniertes VALUE Format erzeugt. Dabei wird bei einem Zeilenvektor die Spaltenüberschrift (bei einem Spaltenvektor die Zeilenbeschriftung) als **Formatted Value** genutzt.

Der Name des Formates wird durch Zählung gebildet, z.B.:

```
PROC FORMAT   LIBRARY=LIBRARY;
     VALUE    _00001_     1 = " "
                          2 = "Schleswig-Holstein"
                          3 = "Freie und Hansestadt Hamburg"
                          4 = "Freie Hansestadt Bremen"
                          5 = "Niedersachsen"
                          6 = "Nordrhein-Westfalen"
                          7 = "Hessen"
                          8 = "Rheinland-Pfalz"
                          9 = "Baden-Württemberg"
                         10 = "Freistaat Bayern"
                         11 = "Saarland"
                         12 = "Bundeshauptstadt Berlin"
                         13 = "Brandenburg"
                         14 = "Mecklenburg-Vorpommern"
                         15 = "Sachsen-Anhalt"
                         16 = "Freistaat Sachsen"
                         17 = "Freistaat Thüringen"
                         18 = "nicht in Deutschland";
```

3. Als **LABEL** der Variablen wird bei Auswahllisten mit diskreter Auswahl der „nächstgelegene“ Text (1. links, 2. oberhalb, 3. rechts vom Eingabefeld) verwendet, d.h. hier erzeugt der Generator die Anweisung:

   ```
   LABEL    land  = "Hauptwohnsitz in Bundesland:";
   ```

   Man erkennt unschwer den relativ geringen Aufwand einer nachträglichen Editierung, z.B. zu:

   ```
   LABEL    land  = "Hauptwohnsitz in Bundesland";
   ```

Listen mit multipler Auswahl

Listen, die multiple Auswahlen zulassen, d.h. es wurde das Attribut MULTIPLE gesetzt, liefern einen **Vektor dichotomer Merkmale**, die je die Werte 1 und 0 (selected oder unselected) annehmen. Die Anzahl der Vektorkomponenten entspricht der Anzahl der Optionen in der Auswahlliste.

Missing Values entstehen nur, wenn der Vektor aus einem, z.B. infolge von Filterführung, nicht aktivierten Fragekomplex stammt.

Die implizite **Voreinstellung** aller Variablen, die mit multipler Auswahl aus einer vorgegebenen Liste kreiert werden, ist 0 (unselected). Sie kann explizit durch Codierung des HTML Attributs SELECTED bei einer oder mehrerer Alternativen geändert werden.

Zum Beispiel die Frage nach den abgeschlossenen Schulbildungen:

```
<TABLE>
  <TR><TD>Abgeschlossene Schulbildung(en)
      <TD><SELECT NAME ="schubi"
                  MULTIPLE
          <OPTION VALUE=1>keine abgeschlossen
          <OPTION VALUE=2>Gesamtschule
          <OPTION VALUE=3>Allg.bildende Polytechn. Oberschule
          <OPTION VALUE=4>Realschule
          <OPTION VALUE=5>Gymnasium
          <OPTION VALUE=6>Erweiterte Oberschule
          <OPTION VALUE=7>Fachschule
          <OPTION VALUE=8>Fachhochschule
          <OPTION VALUE=9>Hochschule
          <OPTION VALUE=10>Spezialschule
          <OPTION VALUE=11>Berufsschule
          <OPTION VALUE=12>sonstige Schule
          </SELECT>
      <TD><FONT SIZE="-1"> Mehrfachauswahl: gleichzeitig
                          gedr&uuml;ckte <I>Strg</I>-Taste</FONT>
</TABLE>
```

Diese Tabelle hat nur eine Zeile mit drei Spalten:

Abgeschlossene Schulbildung(en):	hier befindet sich die Auswahlliste; die übertragenen Werte sind die VALUE Attribute oder, falls die nicht spezifiziert wurden, der selektierte Text	Bedienungshinweis

Wie man am HTML Code erkennt, kann der Befragte durch gleichzeitige Auswahl von *keine abgeschlossene* und mindestens einer beliebigen anderen Alternative **widersprüchliche** und damit nicht mehr korrekt verwertbare Antworten abgeben.

Die **Konsistenz** muss natürlich bereits bei Ausfüllung des Formulars auf dem Client hergestellt werden, was zusätzliche Script Anweisungen erfordert:

```
<TABLE>
   <TR><TD>Abgeschlossene Schulbildung(en)
       <TD><SELECT NAME ="schubi"
                   MULTIPLE
                   onBlur="var n=this.form.schubi.options.length;
                   if  (this.form.schubi.options[0].selected)
                       {
                        for (var i=1; i<n; i++)
                            {
                             if (this.form.schubi.options[i].selected)
                                {
         alert ('Bitte korrigieren Sie den Widerspruch in der Angabe
                 SCHULBILDUNG');
                       for (var j=0; j<n; j++)
                            this.form.schubi.options[0].selected=false;
                                }
                             }
                        }">
               <OPTION VALUE=1>keine abgeschlossen
               <OPTION VALUE=2>Gesamtschule
               <OPTION VALUE=3>Allg.bildende Polytechn. Oberschule
               <OPTION VALUE=4>Realschule
               <OPTION VALUE=5>Gymnasium
               <OPTION VALUE=6>Erweiterte Oberschule
               <OPTION VALUE=7>Fachschule
               <OPTION VALUE=8>Fachhochschule
               <OPTION VALUE=9>Hochschule
               <OPTION VALUE=10>Spezialschule
               <OPTION VALUE=11>Berufsschule
               <OPTION VALUE=12>sonstige Schule
           </SELECT>
       <TD><FONT SIZE="-1"> Mehrfachauswahl: gleichzeitig
                           gedr&uuml;ckte <I>Strg</I>-Taste</FONT>
</TABLE>
```

Aus dieser Konstellation ergibt sich:

1. Die Eingabedaten stammen aus einer Liste von Werten. Es wurde das Attribut MULTIPLE gesetzt. Somit wird ein **Vektor dichotomer Merkmale** gebildet:
   Wenn `VALUE="1"` dann *schubi1*=1 sonst *schubi1*=0
   ...
   Wenn `VALUE="12"` dann *schubi12*=1 sonst *schubi12*=0
   Der Vektor *schubi* in der SAS Datentabelle hat 12 Komponenten, die je die Werte 0 oder 1 besitzen. Missing Values können nur auftreten, wenn der Vektor aus einem nicht frequentierten Fragekomplex stammt.

2. Nutzerdefinierte VALUE Formate werden für dichotome Variable, die z.B. durch Auswahllisten gebildet werden, nicht erzeugt, weil schlecht zu erraten ist, wie die formatted values aussehen sollten (ja/nein, yes/no, ein/aus, gesund/krank usw.)

3. Als **LABEL** der Variablen wird bei Auswahllisten mit multipler Auswahl der „nächstgelegene“ Text (1. links, 2. oberhalb, 3. rechts vom Eingabefeld) verkettet mit dem Text, der die Alternative beschreibt, verwendet, d.h. hier erzeugt der Generator die Anweisung:

```
LABEL    schubi1   = "Abgeschlossene Schulbildung(en) /
                      keine abgeschlossen"
         schubi2   = "Abgeschlossene Schulbildung(en) /
                      Gesamtschule"
         schubi3   = "Abgeschlossene Schulbildung(en) /
                      Allg.bildende Polytechn. Oberschule"
         schubi4   = "Abgeschlossene Schulbildung(en) /
                      Realschule"
         schubi5   = "Abgeschlossene Schulbildung(en) /
                      Gymnasium"
         schubi6   = "Abgeschlossene Schulbildung(en) /
                      Erweiterte Oberschule"
         schubi7   = "Abgeschlossene Schulbildung(en) /
                      Fachschule"
         schubi8   = "Abgeschlossene Schulbildung(en) /
                      Fachhochschule"
         schubi9   = "Abgeschlossene Schulbildung(en) /
                      Hochschule"
         schubi10  = "Abgeschlossene Schulbildung(en) /
                      Spezialschule"
         schubi11  = "Abgeschlossene Schulbildung(en) /
                      Berufsschule"
         schubi12  = "Abgeschlossene Schulbildung(en) /
                      sonstige Schule";
```

Man kann diese LABEL Anweisung leicht editieren, z.B. zu:

```
LABEL    schubi1   = "Schulbildung / keine abgeschlossen"
         schubi2   = "Schulbildung / Gesamtschule"
         schubi3   = "Schulbildung / Allg.bildende Polytechn. Oberschule"
         schubi4   = "Schulbildung / Realschule"
         schubi5   = "Schulbildung / Gymnasium"
         schubi6   = "Schulbildung / Erweiterte Oberschule"
         schubi7   = "Schulbildung / Fachschule"
         schubi8   = "Schulbildung / Fachhochschule"
         schubi9   = "Schulbildung / Hochschule"
         schubi10  = "Schulbildung / Spezialschule"
         schubi11  = "Schulbildung / Berufsschule"
         schubi12  = "Schulbildung / sonstige Schule";
```

Zahlen und Texte

Frei formatierte Nutzereingaben – **Zahlen und Texte** – werden je nach Häufigkeit des Auftreten des Namens des Eingabefeldes im Fragekomplex gespeichert:

- bei einmaligem Vorkommen als Skalar oder
- bei mehrmaliger Verwendung als Vektor

Der Typ der Skalare oder Vektorkomponenten wird durch das TYPE Attribut bestimmt:

- explizit `TYPE="text"` führt zur Speicherung von Zeichenketten,
- das fortgelassene TYPE Attribut führt zur Speicherung als numerische Daten.

Wechsel des Datentyps innerhalb von Vektoren ist unzulässig und führt zu Fehlern.

Die Gültigkeit von Eingaben (**Plausibilitäts-** und **Konsistenzkontrolle**) muss **clientseitig** durch Script Programmierung gesichert werden und betrifft insbesondere

- spezielle Strukturen (z.B. eMail-Adressen),
- gültige externe Zahlendarstellungen,
- Lage von Werten innerhalb/außerhalb numerischer Intervalle,
- Übertragung von einzelnen Nullen (vgl. Besondere Werte von Eingabefeldern),
- Streichung von NewLines in TEXTAREA Eingaben (vgl. Besondere Werte von Eingabefeldern),
- widerspruchsfreie (konsistente) Angaben bei abhängigen Antworten.

Missing Values können auftreten, wenn

- das Eingabefeld nicht ausgefüllt wurde,
- als Wert eine singuläre `"0"` eingegeben und unbehandelt (vgl. Besondere Werte von Eingabefeldern) übertragen wird,
- die Variable aus einem nicht frequentierten Fragekomplex stammt.

Die Gültigkeit von **Paßworten** kann man wohl kaum clientseitig prüfen, ohne die Sicherheit, die man sucht, zugleich wieder preiszugeben. Vmtl. gehört diese Prüfung zur Evaluierung der erhaltenen Daten (vgl. Evaluieren der Daten) und findet zu einem späteren Zeitpunkt an einem anderem Ort statt.

Zum Beispiel die Frage nach dem Alter des Probanden:

```
<TABLE>
   <TR><TD>Alter in Jahren:
       <TD><INPUT NAME ="alter"
                  SIZE ="2"
                  MAXLENGTH="2"
                  onBlur="checkInt(alter,1,99)">
</TABLE>
```

Die **Plausibilitätskontrolle**, d.h. ein Test ob der eingegebene Wert eine ganze Zahl ist, die zwischen 1 und 99 liegt, ist hier beim event handler onBlur angegeben. Man muss beachten, dass die Function *checkInt* (vgl. JavaScript Quelltexte) zweimal auszuführen ist:

- Innerhalb des Formulars beim Eingabefeld vom event handler **onBlur**, was ggf. nur eine Aufforderung zur Korrektur der Fehleingabe produziert,
- Am logischen Ende des Formulars im <FORM> Tag vom event handler **onSubmit** vor Übergabe der Daten an das CGI-Programm, was die Durchführung evtl. angemahnter Korrekturen überprüft und ggf. das Absenden des Formulars verhindert. Sind mehrere Felder zu prüfen, ist die Konjunktion der Return Values der Funktionsaufrufe zu bilden.

Die Tabelle mit dem Eingabefeld für das Alter hat nur eine Zeile mit zwei Spalten:

Alter in Jahren:	hier befindet sich das Eingabefeld

Aus dieser Konstellation ergibt sich:

1. Die Eingabedaten kommen aus einem frei formatierbaren Feld. Es wurde kein TYPE Attribut vereinbart. Somit wird der übertragene Wert einer **numerischen Variablen**, die ein **quantitatives Merkmal** enthält, zugewiesen.
2. Nutzerdefinierte VALUE Formate werden für frei formatierbare Eingabefelder nicht generiert.
3. Als **LABEL** der Variablen wird der „nächstgelegene“ Text (1. links, 2. oberhalb, 3. rechts vom Eingabefeld) verwendet, d.h. hier erzeugt der Generator die Anweisung:

   ```
   LABEL    alter  = "Alter in Jahren:";
   ```

   Nachbesserndes Editieren ist, wenn überhaupt lohnenswert, geringfügig:

   ```
   LABEL    alter  = "Alter in Jahren";
   ```

3.3 Installation

Es empfiehlt sich, ein **spezifisches SAS Desktop Icon** für den Programmgenerator zu verwenden. Auf diese Weise können eine Reihe von Voreinstellungen per Autoexec.sas standardisiert werden. Dies betrifft insbesondere

- Pfade der Programmbibliothek und von temporären Dateien,
- Systemoptionen wie z.B. VALIDVARNAME,
- die Druckerausrichtung LANDSCAPE / PORTRAIT und
- Setzen der Werte globaler Macrovariablen, wie z.B. eines FOOTNOTE Textes auf den Namen des Rechenzentrums

3.4 Aufruf

Auszuführen ist das Programm *Main.sas*[50] (vgl. Abb. 13), in dem notwendig eine Zuweisung zur HTML Maske anzugeben ist, die alle zur Umfrage gehörenden HTML Dokumente einschließt, die einen <FORM> Tag enthalten.

Schließt die Maske andere Dokumente ein, so führen

- weitere HTML Dokumente zu überflüssigem Aufwand und
- andere Dateitypen zu Fehlern.

Schließt die Maske nicht alle zum Fragebogen gehörigen HTML Dokumente ein, entsteht ein unvollständiges Interpreter-Programm, das letztlich nicht erfolgreich arbeiten kann.

Es dürfte kein Problem mit der Maske entstehen, wenn man vorab alle HTML Dokumente, die zum Fragebogen gehören und einen <FORM> Tag enthalten, zu einer Datei konkateniert. Auf diese Weise kann man u.a. auch die spätere Reihenfolge der Variablen in den Zeilen der SAS Datentabelle kontrollieren, da diese der Reihenfolge ihrer Definition im HTML Formular entspricht. Wird jedoch eine Maske verwendet, um die HTML Dokumente, die zum Fragebogen gehören, zuzuordnen, so ist die Reihenfolge der zugeordneten HTML Dokumente von der rein physischen Reihenfolge der Namenseinträge im Verzeichnis (nicht etwa von einer alphabetischen Ordnung) abhängig und damit weitestgehend zufällig. Man erhält dann nur eine segmentweise (je Fragekomplex) an der Reihenfolge ihrer Vereinbarung im HTML Dokument orientierte Anordnung der Variablen in den Zeilenvektoren der SAS Datentabelle.

Optional kann zusätzlich eine MAIL Maske vereinbart werden, die die Textfiles der zur Umfrage gehörenden eMails einschließt. Diese Maske wird jedoch erst im Interpreter-Programm wirksam und kann so auch später aktualisiert werden.

[50] Das Generator-Programm wird ausschl. im Rechenzentrum betrieben.

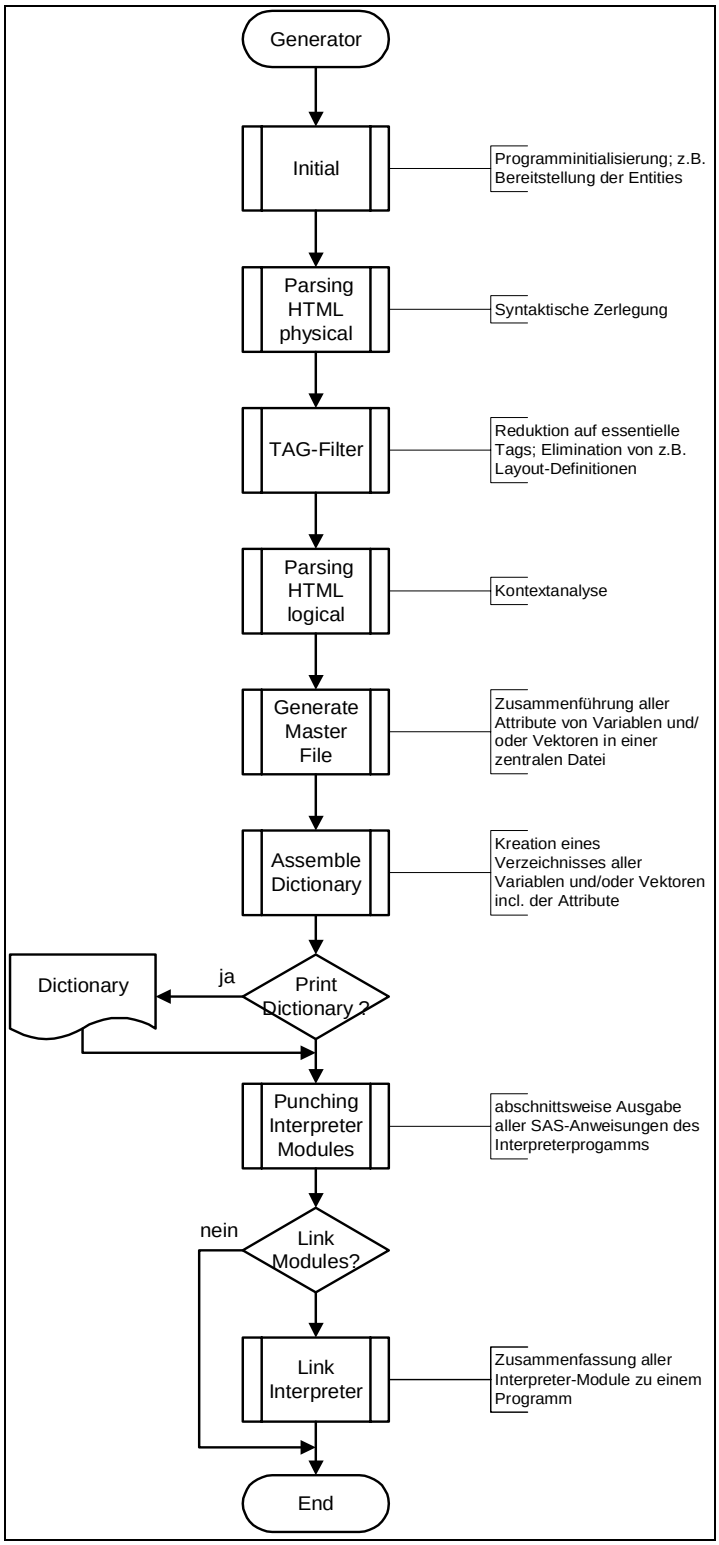

Abbildung 13: Arbeitsweise des Programmgenerators (Main.sas)

3.5 Resultate (Beispielfragebogen)

Die Resultate können im Rahmen des Beitrages nicht in gedruckter Form wiedergegeben werden. Sie sind jeweils unter dem angegebenen Dateinamen auf der Proceedings-CD gespeichert.

3.5.1 LOG Protokoll

Das LOG-Protokoll (*Anhang/Internetnutzung.log*[51]) wird im gleichnamigen SAS Fenster ausgegeben und enthält einige Ablaufdaten zur Generierung des Interpreter-Programms.

Die meisten Ausgaben sind entwicklernahe Informationen.

Fehlerausschriften und Warnungen werden den Arbeitsschritten des Generators zugeordnet und sind somit neben dem erkennbaren Fortschritt der Bearbeitung die primär wichtigen Informationen für Anwender.

3.5.2 Verzeichnis aller Variablen und Felder

Dieses Verzeichnis (*Anhang/Internetnutzung.lst*[51]) erscheint im OUTPUT Fenster des SAS Systems und enthält eine Liste aller erkannten Variablen- und Feldnamen und der zugehörigen Attribute:

- **LENGTH**,
- **LABEL**,
- **TYPE** (wie in HTML Dokument),
- **DIMENSION**
  0 = kein Eingang in die SAS Datentabelle
  1 = Skalar; Dimension 1 wird nicht explizit ausgegeben
  > 1 = Feld
- **Wertevorrat** von kategoriellen Merkmalen
  in der SAS Datentabelle,
  in der eMail, die vom Formular versendet wird
- Label für Werte kategorieller Merkmale (Grundlage für **Formate**)

Zusätzlich gibt es ggf. zwei Fehlerindikatoren:

- ambiguous types: Feldelemente eines Feldes wurden mit unterschiedlichen Typen definiert,
- ambiguous values: Einem kategoriellem Merkmal wurde im Wertevorrat mehrfach derselbe Wert zugeordnet.

[51] Die Datei befindet sich auf der Proceedings-CD

3.5.3 Das Interpreter-Programm

Der Name des Interpreter-Programms wird aus der HTML Maske durch Streichung aller Wildcard Zeichen und durch Verwendung der Extension *sas* abgeleitet. Im Beispiel entsteht so als Programmname *Internetnutzung.sas*.

Als Speicherort wird für alle Teile des Interpreter-Programms dasselbe Verzeichnis verwendet, in dem auch die HTML Dokumente gespeichert sind, die die Fragekomplexe enthalten.

Im Beispiel besteht die generierte Version des Interpreter-Programms aus ca. 800 SAS Anweisungen (*Anhang/Internetnutzung.sas*[52]), die nur unwesentlich zu modifizieren sind.

4 Arbeitsweise des Interpreter-Programms

4.1 Vorbereitungen

4.1.1 Editieren des generierten Interpreter-Programms

Vor Verwendung des generierten Interpreter-Programms ist eine Durchsicht unbedingt erforderlich. Das Editieren des Interpreter-Programms betrifft vor allem Texte, die dem Kontext entnommen wurden und die in FORMAT Prozeduren und LABEL Anweisungen einfließen.

Im Beispiel ist allein die LABEL Anweisung betroffen und wegen deren kommentierendem Charakter nicht zwingend in der angegebenen Weise zu codieren.
(*Anhang/InternetnutzungEdited.sas*[52],[53])

4.1.2 Speichern der eMails als Textdateien

Das Interpreter-Programm verarbeitet eingabeseitig Textdatei(en), die die gespeicherten eMails mit den Antworten der Probanden enthalten. Die Erzeugung der Textdateien lässt sich sehr einfach mit dem Mail-Tool EUDORA erreichen (vgl. Abb. 14), indem man die Mailbox

- nach ***subject*** (vgl. Anmerkung 6 zum HTML Quelltext des ersten Fragekomplexes) sortiert,
- alle eMails mit dem interessierenden *subject* **en bloc markiert** und
- dann via Menü **File** ⇒ **Save As** als Textdatei speichert.

[52]Die Datei befindet sich auf der Proceedings-CD

[53]Das editierte Interpreter-Programm kann vom Auftraggeber bei entsprechender Arbeitsumgebung in eigener Regie betrieben werden.

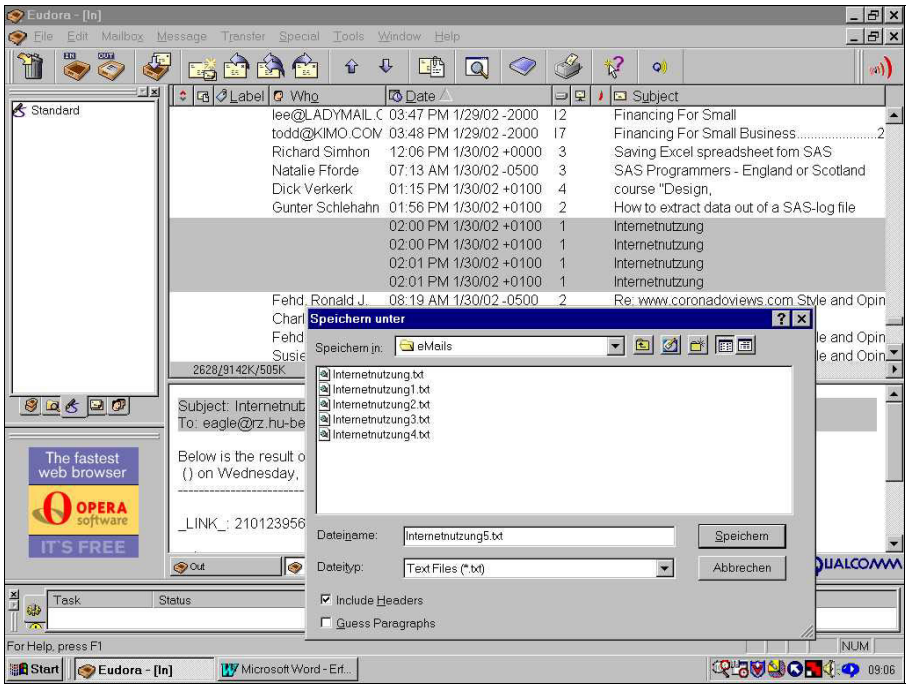

Abbildung 14: Speichern der markierten eMails als Textdatei

Zu beachten ist, dass bei der Speicherung die **Header** mit ausgegeben werden, was durch Aktivierung der entsprechenden Checkbox gewährleistet wird. Die Textdatei enthält alle markierten eMails in lückenlos konkatenierter Form ohne besondere Delimiter. Die Textdatei(en) können am Ende der Felddauer oder bei z.B. umfangreicherem eMail Eingang zyklisch gespeichert werden.

4.2 Aufruf

Der Name des Interpreter-Programms wird aus der HTML Maske, die bei Aufruf des Generator-Programms angegeben wurde, durch Streichung aller Wildcard Zeichen und durch Verwendung der Extension *sas* gebildet. Im Beispiel entsteht so als Programmname *Internetnutzung.sas*.

Als Speicherort für das Interpreter-Programm wird dasselbe Verzeichnis verwendet, in dem auch die HTML Dokumente gespeichert sind, die die Fragekomplexe enthalten.

Die editierte Version des Interpreter-Programms heißt im Beispiel ***InternetnutzungEdited.sas***[54].

Bevor das editierte Programm ausgeführt wird, muss eine Zuweisung zur MAIL Maske angegeben werden, die alle zur Umfrage gehörenden Textdateien einschließt, die die Antworten der Probanden enthalten.

[54]Die Datei befindet sich auf der Proceedings-CD

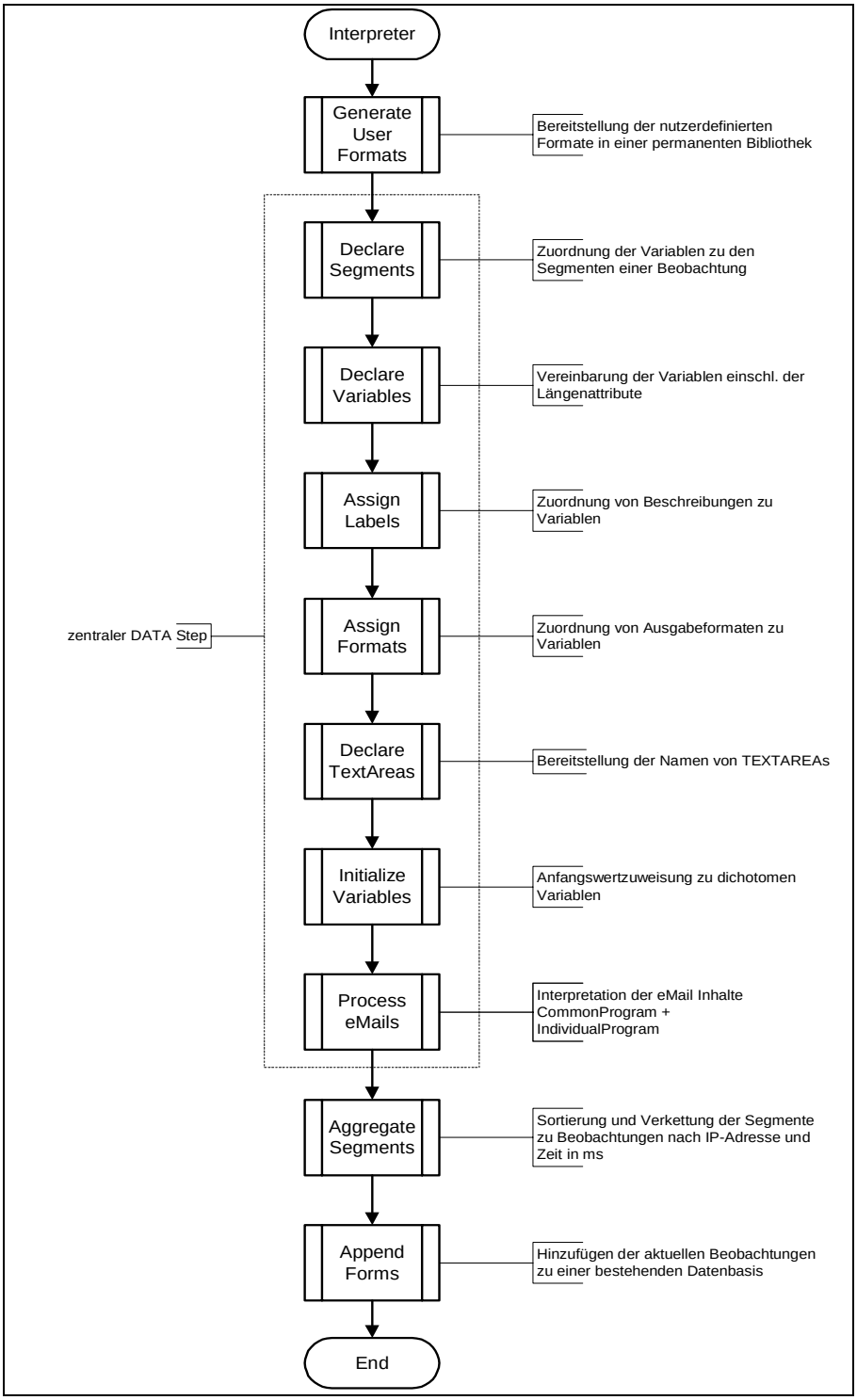

Abbildung 15: Arbeitsweise des Interpreter-Programms

Schließt die Maske

- andere Dateien ein, so führt dies zu Fehlern,
- nicht alle Textdateien mit erhaltenen eMails ein, entsteht eine unvollständige SAS Datentabelle.

Das editierte Interpreter-Programm kann vom Auftraggeber bei entsprechender Arbeitsumgebung in eigener Regie betrieben werden.

4.3 Die Resultate

Im Ergebnis der Ausführung des editierten Interpreter-Programms erhält man eine temporäre SAS Datentabelle *WORK.FORM*, deren Aufbau prinzipiell in Abb. 3 dargestellt ist. Diese temporäre Tabelle wird einer permanenten SAS Datentabelle ***LIBRARY.FORM*** mit gleicher Struktur am Ende hinzugefügt (appended).

Das Resultat zum Beispiel verdeutlicht PROC CONTENTS.
(*Anhang/InternetnutzungContents.lst*[55])

Zusätzlich wird ein permanenter Katalog von nutzerdefinierten Formaten ***LIBRARY.FORMATS*** angelegt.

Als Speicherort für die Resultate wird per Voreinstellung dasselbe Verzeichnis verwendet, in dem auch die HTML Dokumente gespeichert wurden, die die Fragekomplexe enthalten.

Jede Zeile der SAS Datentabelle (= Beobachtung = kompletter Fragebogen) enthält:

- die Variablen, die als Eingabefelder in den HTML Dokumenten der Fragekomplexe auftraten und
- zusätzlich einige technische Variable, die als Kennung je ein Underline am Namensanfang tragen

Technische Variable in Fragebögen sind stets:

_IP_	übergeordneter Schlüssel des Fragebogen-ID (IP-Adresse)
LINK	untergeordneter Schlüssel des Fragebogen-ID (Zeit in ms seit 1.1.1970)
CLIENT	zur Beantwortung verwendeter Browser
ESC	Filter zur Erkennung von Abbrechern (Abbrecher: _ESC_$<$ *level* mit $(0 \leq)$ *level* ≤ 100),
_RC_	Bearbeitungscode

Zusätzliche Technische Variable in Fragebögen, die **1 Fragekomplex** enthalten, sind:

DATE	Versanddatum des Fragebogens
TIME	Ortszeit bei Versand des Fragebogens

[55] Die Datei befindet sich auf der Proceedings-CD

Zusätzliche Technische Variable in Fragebögen, die ***n* Fragekomplexe** enthalten, sind:

_F_i	Nr. des Fragekomplexes	} mit $i = 1 \ldots n$
_D_i	Versanddatum des Fragekomplexes	
_T_i	Ortszeit bei Versand des Fragekomplexes	

Der **Bearbeitungscode** _RC_ ist ein dualer Summencode:

0	fehlerfrei
1	einer Variablen wurde ein unzulässiger Wert zugewiesen
2	eine Felddimension wurde überschritten
4	eine unbekannte Variable tritt in der eMail auf
8	eine eMail ist unvollständig; besteht nicht aus Cover- und Body-Section

Eine Besonderheit in der Arbeitsweise des Interpreter-Programms ist die Behandlung von eMails mit gleichem Fragebogen-ID (_IP_, _LINK_). Dies kann durch Verwendung des **BACK Buttons** des Browsers und erneutes Absenden des Formularinhaltes von jedem Probanden sehr schnell verursacht werden. Derartige Fälle werden als **Korrekturwunsch** gedeutet und nur die eMail mit der jüngsten Versandzeit füllt die Daten des betreffenden Fragekomplexes.

Ein ähnliches Problem kann dadurch entstehen, dass man das Interpreter-Programm mehrfach aufruft, dabei aber, zumindest teilweise, bereits interpretierte eMails erneut verarbeitet. Diese eMails führen dann infolge des Hinzufügens am Ende der permanenten SAS Datentabelle *LIBRARY.FORM* zu wiederholtem Auftreten von Beobachtungen, was in einem abschließenden Evaluierungsschritt behoben wird.

5 Evaluieren der Daten

Abschließend ist einmal das SAS Programm ***Evaluate.sas***[56] auszuführen.
Dieses Programm verarbeitet eingabeseitig die permanente SAS Datentabelle *LIBRARY.FORM* und erzeugt ausgabeseitig eine permanente SAS Datentabelle ***LIBRARY.BASE***, die die Datenbasis für die nun folgenden Auswertungen bildet.

Die Aufgabe des Programms ist die Reduktion der SAS Datentabelle auf den effektiven Anteil nach vorgebbaren Parametern, d.h.

zeilenbezogen:

- Streichen „leerer“, d.h. **ausschl. Missing** Values enthaltender Beobachtungen,
- Streichen mehrfacher, mit **gleichem Fragebogen-ID** vorkommender Beobachtungen,
- Streichen von unvollständigen Beobachtungen (**Abbrecher**: _ESC_ $\leq$ *level* mit $(0 \leq) level \leq 100$),
- Streichen von **fehlerhaften Beobachtungen** mit einem Bearbeitungscode _RC_>0

[56] Die Datei befindet sich auf der Proceedings-CD

spaltenbezogen:

- Streichen von Variablen, die weniger als die vorgegebene Mindestanzahl von gültigen Werten besitzen

Es wird ein Protokoll im LOG Fenster des SAS Systems aufgezeichnet.

Dem Evaluierungsschritt kann ein spezifisches Datenmanagement, die Ableitung neuer Merkmale u.ä. unmittelbar folgen.

6 Kritische Bewertung

Bei zwei Anwendungen liegen zu wenig Erfahrungen vor, um das Verfahren zu empfehlen oder zu verwerfen. Aus diesem Grunde wird das Programm auch nicht als vollendet angesehen und außer Haus gegeben. Interessenten sind aber zu einer kooperativen Zusammenarbeit mit dem Rechenzentrum der Humboldt-Universität zu Berlin herzlich eingeladen.

Literatur

[1] Janetzko, Dietmar (1999). *Statistische Anwendungen im Internet*, Addison-Wesley

[2] Born, Günter (1998). *HTML 4 Kompendium*, Markt+Technik Buch- und Software-Verlag GmbH

[3] Steyer, Ralph (1997). *HTML 4*, Data Becker

[4] Dellwig, Ellmar und Ingo (2001). *JavaScript 1.3*, Markt+Technik Verlag

[5] SAS Institute Inc. (1999). *SAS Macro Language: Reference Version 8*, SAS Institute Inc.

[6] Mocigemba, Dennis et al. (2001). *Methoden der Online-Befragung*, `http://www.demotopia.de`

[7] Shiran, Yehuda und Tomer (2000). *Doc JavaScript*, `http://www.webreference.com/js`