4 Methodik

Aus der in Kapitel 2 formulierten Zielstellung ergab die Anlyse der damit verbundenen Anforderungen, dass (1) bereits verfügbare elektronische Lehrmittelssysteme integriert werden müssen, (2) der automatisierbare Zugriff auf darin enthaltene Ressourcen herzustellen und als Dienst anzubieten ist und (3) der Wissenstransfer nahtlos in den Arbeitsfluß des Anwenders durch funktionale Kopplung von Diensten und Applikationen über das Internet erfolgen muss. Durch die angestrebten Synergieeffekte aufgrund dieser Interoperabilität zur Unterstützung des Wissenstransfers in Lehre und Weiterbildung, muss der methodische Ansatz auch Möglichkeiten der Integration in eine gruppenarbeitstaugliche, internetbasierte Lehr- und Kommunikationsplattform enthalten. In den nachfolgenden Unterkapiteln wird der methodische Ansatz, basierend auf einem entwickelten XML-basierten *Web-Services*, die implementierte Umsetzung und die Verschmelzung der Kopplung zwischen Anwendung und Web-Service zu einer modularen Erweiterung der kostenfrei verfügbaren Kommunikationsplattform. Die eingesetzten und entwickelten Methoden greifen auf die im Kapitel 3 beschriebenen Grundlagen und Materialien zurück.

4.1 Web-Service - Interoperabilität

Das Zusammenwirken von Programmen über standardisierte Schnittstellen und Protokolle bildet die Grundlage für eine effiziente Wissensübermittlung und Synergieeffekten einzelner Funktionsmodule. Im Folgenden sind die Methoden zur Realisierung der Zielsetzung einer spontanen (ad-hoc), interaktionsfreien Interoperabilität zusammengestellt. Das Zusammenwirken der dabei beteiligten Hardware und Software ist in Abbildung 6 dargestellt. Damit kann die Integration verschiedener Geräteklassen und Betriebssystemplattformen realisiert werden. Dies erfolgt mittels *SOAP*-Nachrichten in standardisierter Kommunikation, wodurch die angebotenen, funktionellen Dienste in konsumierender oder bilateraler Weise genutzt werden können. Die Kommunikation zwischen Applikation und *Web-Service* basiert auf einheitlich auf dem Transportprotokoll *HTTP*. Mit dieser Bindung an ein übliches Transportprotokoll für Dienstnutzung im Internet (vgl. *WWW*-Dienst im Kapitel 3.10) wird der Datenfluss nicht durch Blockaderegeln unterbrochen, die für abweichende Protokolle oftmals anzu-

treffen sind. Blockaden für unübliche Transportprotokolle werden individuell von den jeweiligen Netzwerkadministrationen definiert und sollen die Datensicherheit erhöhen. Eine Änderung dieser Blockaderegeln müsste von sämtlichen Netzwerkadministrationen durchgeführt werden. Die gewählte Transportprotokollbindung vermeidet diesen beträchtlichen Aufwand.

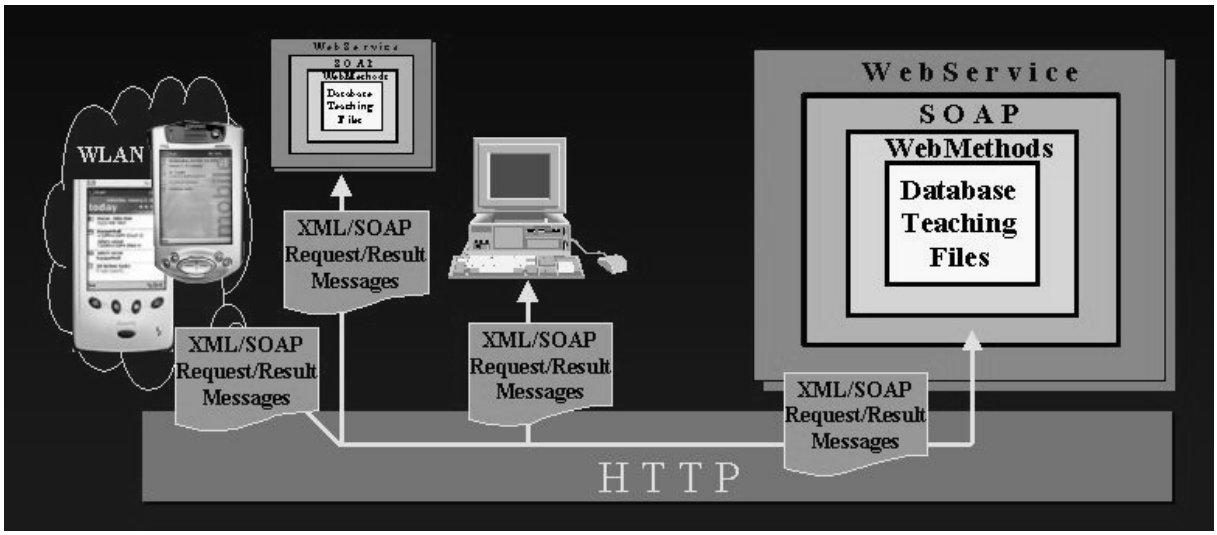

Abbildung 6 Schematischer Zusammenhang von Hard- und Software unter Berücksichtigung der Interoperabilität: von unterschiedlichen Geräteklassen und Plattformen kann mittels *SOAP*-Nachrichten über eine Internetverbindung mit dem *Web-Service* kommuniziert werden. Dies erfolgt im in der Abbildung dargestellten Beispiel auf Basis des *HTTP*-Protokolls. Selbstverständlich können auch *Web-Services* ihrerseits in der Position eines *Clients* die Funktionalität eines anderen Web-Services nutzen.

Im Folgenden werden die in Abbildung 6 gezeigten Elemente beschrieben, die den dargestellten *Web-Service* methodisch umsetzen.

4.1.1 SOAP-Adapter

Der *SOAP*-basierte Adapter interpretiert die eintreffenden Nachrichten und führt sie einer serverseitigen Verarbeitung durch entsprechend vorbereitete Methoden zu (vgl. Kapitel 4.1.2). Hierzu müssen diese Methoden bei der SOAPx4-basierten *Server*-Funktionalität registriert werden. Listing 3 zeigt den entsprechenden *Quellcode*-Abschnitt, mit dem die interoperablen Methoden exportiert werden.

Hierbei handelt es sich um die Funktionen

(1) GetURIMediaImageListByKeyword

(2) GetURIMediaHTMLTextListByKeyword

(3) GetURIMediaPPTListByKeyword.

```
// Eingabe: Diagnosen-Suchbegriff; Ausgabe: Liste von URI's der Images
$server->add_to_map(
             "GetURIMediaImageListByKeyword",
             array("string"),
             array("array")
      );
// Eingabe: Diagnosen-Suchbegriff; Ausgabe: Liste von URI's der textuellen Anmerkungen
$server->add_to_map(
             "GetURIMediaHTMLTextListByKeyword",
             array("string"),
             array("array")
      );
// Eingabe: Diagnosen-Suchbegriff
// Ausgabe: Liste von URI's der PowerPoint-Teaching-Files
$server->add_to_map(
             "GetURIMediaPPTListByKeyword",
             array("string"),
             array("array")
      );
```

Listing 3: Eintragung der Interoperabilitätsmethoden in den *SOAP*-Adapter. Diese Methoden implementieren die per Web-Service angebotenen Web Methods und finden in der *WSDL*-Definition ihre Entsprechung im Abschnitt „Operationen“. Die funktionale Beschreibung dieser Methoden folgt im anschliessenden Kapitel 4.1.2.

Die funktionale Beschreibung dieser im *SOAP*-Adapter registrierten Methoden schließt sich im folgenden Kapitel an.

4.1.2 Web Methods

Die über den *SOAP*-Adapter adressierten Methoden, die sogenannten *Web Methods*, bilden in der Implementierung die Kernfunktionalität. Der interoperable Zugriff erfolgt über Schlüsselworte auf Bild-, Anmerkungs- und Präsentationsdaten. Listing 3 zeigt die Deklarationsrümpfe dieser *Web Methods* bei deren Export über den *SOAP*-Adapter. Die Methoden bieten die Akquisition von Lehrmaterialien aus der radiologischen Lehrmittelsammlung entsprechend ihres Medientyps an. Im Bedarfsfall könnten im Regelbetrieb beliebige weitere *Web Methods* implementiert und der Menge an *Web Methods* problemlos hinzugefügt werden. Die definierten *Web Methods* entsprechen der Prämisse, nach einem durch die Radiologie definierten Standard, eindeutig dokumentierte, typisierte Daten an den Anforderer zu übermitteln.

Dementsprechend sind für jede Datentypklasse separate *Web Methods* definiert, die über die entsprechenden Verzeichniseinträge eindeutig und automatisierbar lokalisiert werden können.

4.1.3 WSDL-Dokument des Radiologischen WebService

Die Interoperabilität im Sinne eines *Web-Services* wird mittels einer *WSDL*-Beschreibung repräsentiert. Das Listing 4 stellt die neu erzeugte *WSDL*-Beschreibung des Web-Services der Radiologie dar. In diesem Listing sind demnach alle Definitionen, Datentypen und Methoden ausgewiesen, die für die Interoperabilität und die automatische Generierung von Zugriffsmethoden auf Seiten des *Clients* benötigt werden könnten. (Vgl. 3.6). Das Listing 4 ist somit eine Art Quellcode für Programme, die dieses *WSDL*-Dokument einlesen, analysieren und die *Web Methods* in den Funktionsraum der importierenden Applikation integrieren. Zum Standard konform, ist in Zeile 01 die verwendete Version des XML-Standards und die Kodierungsvorschrift (UTF-8) angegeben. Die Kodierungsvorschrift ist relevant, da für die Übertragung per *HTTP* üblicherweise lediglich die *ASCII*-Kodierung angenommen wird, und dementsprechend bei beiden Kommunikationsendpunkten eine Kodierung/Dekodierung vorgenommen werden muss. Die Definitionsbeschreibung wird in Zeile 02 mit dem „definition“-Schlüsselwort eingeleitet und umfasst den gesamten Bereich bis zum abschließenden „/definition“ in Zeile 83. Wesentlich zum Verständnis ist die Identifizierung der *Web Methods*, also jener Funktionen, die für die Interoperabilität zur Verfügung stehen. Diese bereits definierten, drei Methoden (vgl. Kapitel 4.1.1) werden im *WSDL*-Dokument mit dem Schlüsselwort „operation“ bezeichnet, (Zeilenbereich von Zeile 20 bis Zeile 32). Wie bereits im Grundlagenkapitel (Kapitel 3.6) ausführlich erläutert, sind diese *Web Methods* Teil einer „Port“-Definition (Zeile 19), die in der „Binding“-Sektion (ab Zeile 34 bis Zeile 74) als Typ referenziert wird. Die Zusammenführung der „Binding“-Sektion und der „Port“-Sektion erfolgt durch die Verschmelzung zu einem „Service“, definiert von Zeile 75 bis Zeile 81. Ebenfalls Teil der „Service“-Sektion ist die Definition der „soap adress location“ in Zeile 77, die den Zielpunkt der Kommunikationsanfragen festlegt, in dieser Implementierung entsprechend die *URL* des SOAP-Adapters.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="KnowledgeBase_Service-interface"
            targetNamespace="http://tf-server.ukbf.fu-
            berlin.de/Soapx4/UKBFMedicalKnowledgeBaseWebService.wsdl"
            xmlns="http://schemas.xmlsoap.org/wsdl/"
            xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns:tns="http://tf-server.ukbf.fu-
            berlin.de/Soapx4/UKBFMedicalKnowledgeBaseWebService.wsdl"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<message name="KeywordRequest">
  <part name="keyword" type="xsd:string"/>
</message>

<message name="URIListKeywordResponse">
  <part name="result" type="xsd:string"/>
</message>

<portType name="KnowledgeBase_Port">
  <operation name="GetURIMediaImageListByKeyword">
    <input message="tns:KeywordRequest"/>
    <output message="tns:URIListKeywordResponse"/>
  </operation>
  <operation name="GetURIMediaHTMLTextListByKeyword">
   <input message="tns:KeywordRequest"/>
    <output message="tns:URIListKeywordResponse"/>
  </operation>
  <operation name="GetURIMediaPPTListByKeyword">
    <input message="tns:KeywordRequest"/>
    <output message="tns:URIListKeywordResponse"/>
  </operation>
</portType>

<binding name="KnowledgeBase_Bind"
         type="tns:KnowledgeBase_Port">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetURIMediaHTMLTextListByKeyword">
    <soap:operation soapAction="urn:UKBFMedicalKnowledgeBaseWebService"/>
    <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:UKBFMedicalKnowledgeBaseWebService"
                 use="encoded"/>
    </input>
    <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:UKBFMedicalKnowledgeBaseWebService" use="encoded"/>
    </output>
  </operation>
```

```
49   <operation name="GetURIMediaImageListByKeyword">
50     <soap:operation soapAction="urn:UKBFMedicalKnowledgeBaseWebService"/>
51     <input>
52       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
53                  namespace="urn:UKBFMedicalKnowledgeBaseWebService"
54                  use="encoded"/>
55     </input>
56     <output>
57       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
58                  namespace="urn:UKBFMedicalKnowledgeBaseWebService" use="encoded"/>
59     </output>
60   </operation>
61   <operation name="GetURIMediaPPTListByKeyword">
62     <soap:operation soapAction="urn:UKBFMedicalKnowledgeBaseWebService"/>
63     <input>
64       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
65                  namespace="urn:UKBFMedicalKnowledgeBaseWebService"
66                  use="encoded"/>
67     </input>
68     <output>
69       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
70                  namespace="urn:UKBFMedicalKnowledgeBaseWebService" use="encoded"/>
71     </output>
72   </operation>
73 </binding>
74
75 <service name="KnowledgeBase_Service">
76   <port binding="tns:KnowledgeBase_Bind" name="KnowledgeBase_ServicePort">
77       <soap:address location="http://tf-server.ukbf.
78        fu-berlin.de/Soapx4/UKBFMedicalKnowledgeBaseWebService.php"/>
79       </port>
80
81 </service>
82
83 </definitions>
```

Listing 4: Die WSDL-Beschreibung des prototypischen WebService der Radiologie. Per Definition des Services « KnowledgeBase_Service" mit Angabe der Web Methods (Operationen), der Nachrichten (Messages) und der Protokoll-Bindungen können aus dieser Beschreibung vollautomatisch enstprechende Interoperabilitätsfunktionen für den Kommunikationspartner generiert werden.

4.1.4 UDDI-Publikation

Nach Erhebung aller entsprechend der *UDDI*-Spezifikation notwendigen Daten zur vollständigen Beschreibung des *Web-Services*, wurden diese im *UDDI*-Register [34] von IBM publiziert. Die folgende Abbildung 7 zeigt ein Bildschirmfoto der registrierten Daten, die neben

den deskriptiven Elementen, insbesondere eine *URL*-Referenz auf das *WSDL*-Dokument des radiologischen *Web-Service* beinhalten. Diese, im *XML-Tag* <*accessPoint*> angegebene Referenz (siehe (A) in Abbildung 7) dient der Generierung von Zugriffsmethoden auf Seiten des Clients, die aufgrund der vollständigen *WSDL*-Beschreibung vollautomatisiert erfolgen kann.

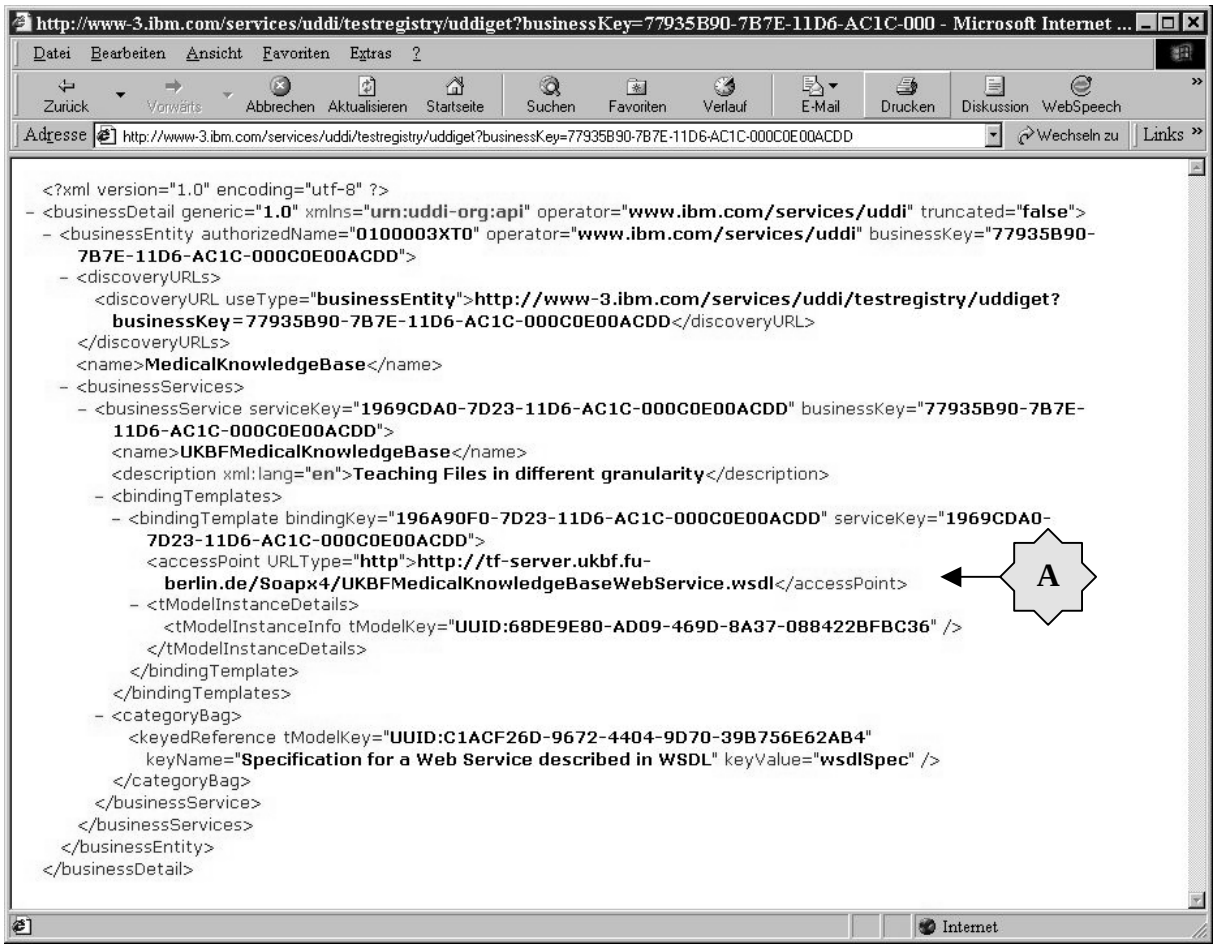

Abbildung 7: Auszug aus dem Eintrag des radiologischen Web-Services im öffentlichen UDDI-Register. Wesentlich zum Zwecke der Interoperabilität ist das Attribut <accessPoint> (siehe (A)) mit der Referenz auf das gültige WSDL-Dokument des Web-Services.

4.2 Arbeitsplatz-Software und Interoperabilität

Das Textverarbeitungssystem *Word2000*, als Quasi-Standard einer verbreiteten Software am Arbeitsplatz, lässt sich mittels einer speziellen Programmiersprache in seiner Funktionalität erweitern. Diese Programmiersprache wird mit *WordBasic* bezeichnet. Die Programmiersprache bietet dem Programmierer eine Schnittstelle zu den Textverarbeitungsfunktionen und zu dem in Bearbeitung befindlichen Textpassagen. Hierdurch wird die direkte programmtechnische Einflussnahme auf den Inhalt des Textdokuments ermöglicht, beispielsweise in Form

von Veränderung, Löschung oder Erweiterung. Entsprechend des in 5.1.1 angenommenen Szenarios besteht eine Zielsetzung in der Erweiterung eines Textdokuments durch radiologisches Referenzbildmaterial. Das Problem kann in sechs Einzelschritte unterteilt werden:

1. Identifizierung des Suchbegriffs durch Markierung im aktuellen Dokument
2. Instanziierung des *SOAP*-Schnittstellenobjekts
3. Festlegung der zu adressierenden *Web Method*
4. Sendung der *SOAP*-Nachricht an den *Web-Service*
5. Empfang und Dekodierung der *SOAP*-Antwortnachricht des *Web-Service*
6. Referenzierung und Integration der radiologischen Bilder in das Dokument

Im ersten Schritt wird das Schlüsselwort, das als Suchbegriff dem Web-Service übermittelt wird, als Parameter übergeben. Hierzu wird die *WordBasic*-Schnittstelle „*Selection*“ eingesetzt, um das im Augenblick des Funktionsaufrufs markierte Wort (vgl. Abbildung 10) im Textverarbeitungsprogramm zu identifizieren (Listing 5).

```
Dim selectedtext As String
    selectedtext = Selection.Text
    If Selection.Type <> wdSelectionNormal Then
        MsgBox "Markieren Sie zuerst das Diagnose-Keyword im Text!"
        Exit Sub
    End If
```

Listing 5: Deklaration des Parameters und Zuweisung des aktuell im Dokument markierte Wort. Im Falle einer nicht erfolgreichen Markierung erfolgt die Ausgabe eines Hinweis auf die mögliche Fehlerursache.

Im zweiten Schritt wird mittels der Bibliothek „*PocketSOAP*“ ein *SOAP*-Schnittstellenobjekt instanziiert, das sich aus einem Objekt für den *SOAP-Envelope* und einem Objekt für das *Transport-Binding* zusammensetzt (Listing 6):

```
Dim e, t
Set e = CreateObject("PocketSOAP.Envelope.2")
Set t = CreateObject("PocketSOAP.HTTPTransport.2)
```

Listing 6: Instanziierung der Objekte für den SOAP-Envelope und der SOAP-Transport-Bindung

Im dritten Schritt wird in der Instanz dieser *SOAP*-Schnittstelle als zu adressierende *Web Method* die Methodenbezeichnung „*GetMediaImageListbyKeyword*“ im radiologischen *Web-Service* festgelegt (Listing 7):

```
e.methodName = "GetURIMediaImageListbyKeyword"   'Web Method, befindlich im WebService
```

Listing 7: Festlegung der adressierten Web Method im radiologischen Web-Service

Desweiteren wird der als Schlüsselwort festgelegte Methodenparameter zugeordnet (Listing 8):

```
e.Parameters.Create "DiagnosisKeyword", selectedtext, "xsd:string"
```

Listing 8: Parametrisierung der Web Method mit dem im Dokument selektierten Schlüsselwort

Im vierten Schritt wird der Methodenaufruf durch sogenannte Serialisation in eine *SOAP*-Nachricht umgesetzt und über das an *HTTP* gebundene Transportobjekt an den *Web-Service* gesandt (Listing 9)

```
t.Send webserviceURI, e.serialize
```

Listing 9: Serialisation in eine SOAP-Nachricht und Versand dieser Nachricht über das http-Transportobjekt an den Web-Service

Nach Verarbeitung der Nachricht im *Web-Service* sendet dieser eine *SOAP*-Antwortnachricht über die erzeugte Instanz der *HTTP*-Transport-Schnittstelle und führt zum fünften Schritt, in dem die Nachricht empfangen und per *XML-Parsing* decodiert wird:

```
e.parse t   'XML-Parsing des Empfangenen
Dim resarr()
    resarr = e.Parameters.Item(0).Value

    If UBound(resarr) = -1 Then MsgBox "Aktuell kein Eintrag für >" + Selection.Text +
"< vorhanden." + Chr(13) + "Bitte versuchen Sie es später einmal wieder..."
```

Listing 10: Empfang der Antwort-SOAP-Nachricht vom Web-Service und Dekodierung der Nachricht per XML-Parsing. Die Ergebniselemente werden in einer Ergebnismenge (resArray) gespeichert. Im Falle einer leeren Ergebnismenge wird ein Hinweistext erzeugt und dem Benutzer mittels Hinweis-DialogBox-Funktion (MsgBox) sichtbar gemacht.

Im letzten Schritt werden die referenzierten Bilder aus der Ergebnismenge mittels verschachtelter Funktionsaufrufe von „*insertWinWordPicture*“ und „*dumpArraytoDoc*“ in das aktuelle Dokument eingefügt (Listing 11).

```
    Call dumpArrayToDoc(resarr)

Function dumpArrayToDoc(dumparr())
    For idx = LBound(dumparr()) To UBound(dumparr())
        insertWinWordPicture (dumparr(idx))
    Next
End Function
Function insertWinWordPicture(fromurl)
With Selection
    '.Collapse Direction:=wdCollapseStart
    .Collapse Direction:=wdCollapseEnd
    .Fields.Add Range:=Selection.Range, Type:=wdFieldIncludePicture, _
        Text:=fromurl, PreserveFormatting:=False
End With
End Function
```

Listing 11: Definition der Hilfsfunktionen und Aufruf dieser zur Integration der referenzierten Bilder in das aktuelle Dokument

4.3 Der radiologische Teaching-File-Server als zu integrierende Kommunikationsplattform

Als Plattform der Gruppeninteraktion und –kommunikation findet die in Kapitel 3.9.3 genannte *PHProjekt*-Implementierung Verwendung. Diese Kommunikationsplattform bildet die Interaktionsanforderungen der radiologischen Benutzer eines Teaching-File-Servers, der unter eigener, maßgeblicher Beteiligung entwickelt wurde, ab. Die vorhandenen Module werden in ihrem Funktionsumfang nachfolgend beschrieben.

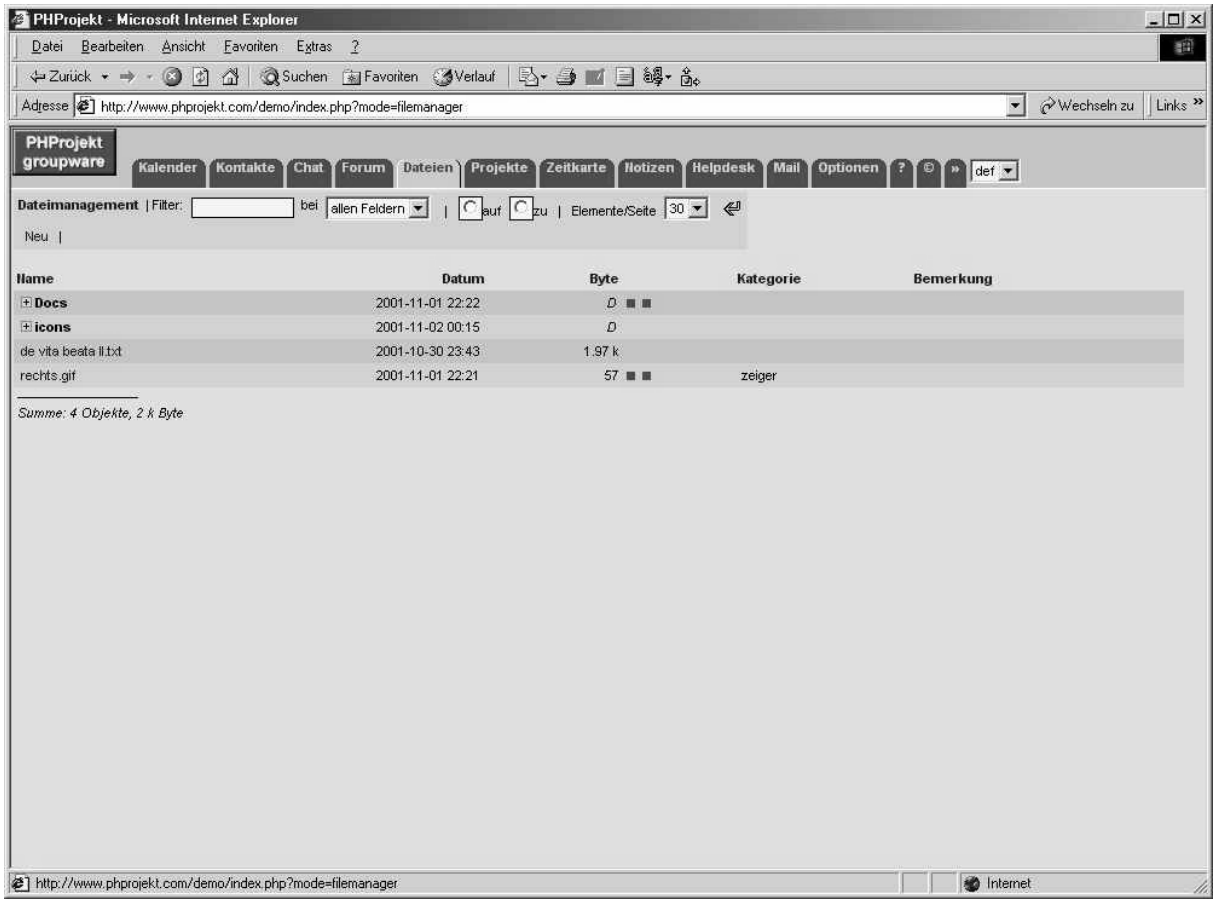

Abbildung 8: Die Benutzeroberfläche von PHProjekt mit aktiver File-Sharing-Auswahl. Die Präsentation der Funktionsmodule erfolgt über eine Karteikarten-Metapher: mit Mausauswahl eines Karteikartenreiters wird die Karte in den Vordergrund gesetzt und zeigt die zur Funktion zugehörigen Interaktionselemente.

4.3.1 Benutzerverwaltung

Mittels eines separaten, administrativen Zugangs, über eine *HTML* basiertes *Interface* lassen sich Benutzer und Benutzergruppen definieren, sowie das Attribut „Teamleiter“ für bestimmte Benutzer setzen. Ein Teamleiter kann beispielsweise in der Terminverwaltung gemeinsame Gruppentermine festlegen. Weitere Attribute wie das Zugangs-Passwort der Benutzer lassen sich ebenso editieren, und schließlich können über die angebotene Funktionalität auch Benutzer entfernt werden. Die Benutzerverwaltung trägt durch die Gruppeneinteilung der Benutzer dazu bei, die üblichen Organisationsstrukturen von Lerngruppen oder Tutorien abzubilden. Zudem sind dadurch Regeln definierbar, die den Zugriff einzelner Benutzer auf vorhandenen Ressourcen bestimmen.

4.3.2 Terminverwaltung

Die Terminverwaltung wird über den Karteikartenreiter „Kalender“ aktiviert. Mit der Terminverwaltung lassen sich Vorlesungstermine bekannt geben, an denen mittels der Integration von Video-Direktübertragung (vgl. 4.3.8) eine Teilnahmemöglichkeit über das Internet besteht. Termine lassen sich für jeden Benutzer separat verwalten, Gruppentermine können jedoch auch von den Teamleitern für seine Teammitglieder eingetragen werden. Als Terminansicht stehen Tageslisten und Wochenlisten zur Verfügung, im Fall von Arbeitsgruppen, können die Termine der Gruppenmitglieder gemeinsam dargestellt werden. Beim Eintrag von neuen Terminen findet eine Überprüfung auf freie Zeitressourcen statt, sodass Überschneidungen vermieden werden.

4.3.3 File-Sharing

Die File-Sharing-Funktion kann genutzt werden, um im Vorlauf von Lehrveranstaltungen, Fachdokumente an die Teilnehmer eines Kurses zu verteilen, oder erarbeitete Dokumente der eigenen Arbeitsgruppe zur Verfügung zu stellen. Beliebige lokale Dateien können vom Benutzer zum Server übertragen werden und die Sichtbarkeit bzw. der Zugriff von anderen Benutzern. Zudem lassen sich Verzeichnisse errichten, um die zentralen Datenressourcen auch logisch organisieren zu können. Diese Funktion ist die aktive Auswahl in Abbildung 8 mit aktivierten Karteikartenreiter „Dateien“.

4.3.4 Chats

Zum spontanen, textbasierten, Meinungsaustausch dient die sogenannte *Chat*-Funktion, auswählbar mit dem gleichnamigen Karteikartenreiter. Die Funktion realisiert einen virtuellen Raum, in den die Benutzer eintreten können, und damit die Kommunikation der übrigen Teilnehmer verfolgen als auch aktiv mit Beiträgen teilhaben können. Die Anzahl der parallel zugeschalteten Teilnehmer ist unbegrenzt, allen Benutzern wird über eine ständig aktualisierte Teilnehmerliste aufgezeigt, welche Mitglieder sich in dem virtuellen Besprechungsraum „aufhalten“, d.h. die *Chat*-Funktion aktiviert haben.

4.3.5 Diskussions-Foren

Das Diskussionsforum wird über den Karteikartenreiter „Forum“ (Abbildung 8) aktiviert. Jeder Benutzer kann eine Diskussion / schriftlichen Meinungsaustausch mit einem Beitrag im

Diskussionsforum initiieren um den fachlichen Dialog aus Tutorien oder anderen Lehrveranstaltungen oder Aufgaben zu vertiefen. Aufgrund der asynchronen Kommunikationsform, ergänzt diese Funktion die übrigen Modul, die zum Meinungsaustausch angeboten werden. Die Beiträge und Meinungsäußerungen der teilnehmenden Benutzer sind allgemein einsehbar, die Funktion entspricht damit weitestgehend den im Alltag geläufigen Anschlagsbrettern.

4.3.6 Projekt/Aufgabenplanung

Aufgaben können mit Ressourcen und Benutzern gekoppelt werden und können zu Projekten zusammengefasst werden. Diese Projekte können in einer Fortschrittsgrafik visualisiert werden, als auch in detaillierter Übersicht der beteiligten Aufgaben und Personen gelistet werden. Damit unterstützt dieses Modul die Planung und Durchführung der Lehrveranstaltungen und koordiniert die Gruppenarbeit. Konkrete Aufgaben und Projekte können von den Kursleitern definiert werden und zur Bearbeitung an einzelne Benutzer oder Benutzergruppen vorgegeben werden. Gekoppelt mit fallbasierten Lehrmaterialien, eröffnet sich die technische Unterstützung für das Lernen nach dem Muster des „Problemorientierten Lernens“. Aktiviert wird dieses Funktionsmodul über den Karteikartenreiter mit der Bezeichnung „Projekte“ (Abbildung 8).

4.3.7 Integration der Teaching-File-Server-HCI

Eine Integration des *Human-Computer-Interface* (*HCI*) als funktionale Erweiterung bestehender Eigenschaften der Kommunikationsplattform erfordert eine visuell nahtlose Einbettung unter Beibehaltung der bestehenden Benutzerführung. Die modulare Struktur der Plattform ermöglicht eine Eingliederung als „Extension“. Für jede Extension wird ein anwählbarer Karteikartenreiter erzeugt, der ein Umschalten auf die Funktion mit dem darin enthalten *HCI* ermöglicht. Die visuelle Einbettung ist in Abbildung 9 als Bildschirmfoto dargestellt, mit aktiviertem Karteikartenreiter „TeachingServer“. Aufgrund der Einbettung der verfügbaren radiologischen Lehrbildsammlung kann der Benutzer diese zur fachlichen Vor- und Nachbereitung nutzen, ohne die Kommunikationsplattform verlassen zu müssen. Ein schneller Wechsel zwischen dieser Funktion und den Kommunikationsmöglichkeiten mit der Gruppe ist damit sichergestellt.

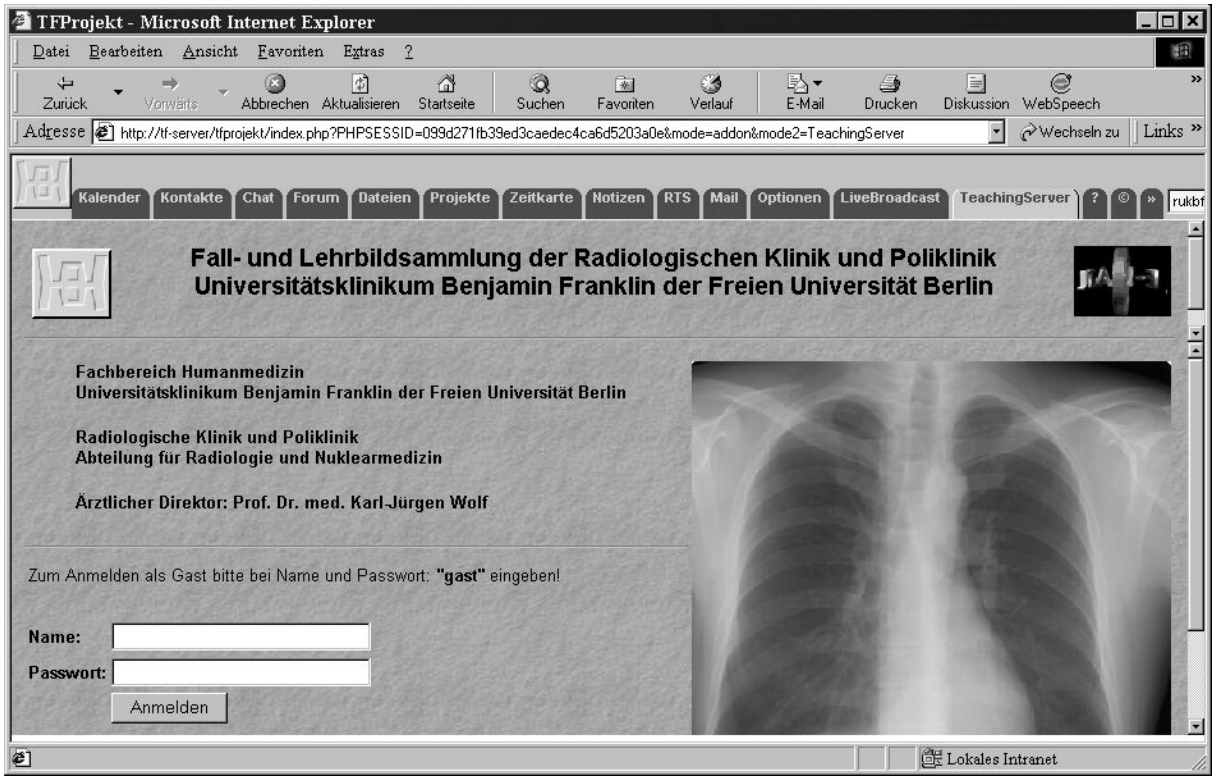

Abbildung 9: Die Einbettung der Benutzer-Interaktions-Schnittstelle zum radiologischen Teaching-File-Server. Ausgewählt durch den aktiven Karteikartenreiter „TeachingServer“, der aufgrund der Einbettung als Extension vom PHProjekt-Rahmenwerk generiert wurde.

4.3.8 Integration von Video-Direktübertragung

Die Integration von Audio/Video-Übertragungen basiert auf einer Kopplung mit einem installierten RealVideo-Server. Die visuelle Integration ist, wie bereits bei der *Teaching-Server-HCI*-Integration, per *Extension*-Einbettung realisiert. Zur Anzeige des *Videostreams* ist beim entsprechenden *HTML-Browser* ein kostenlos verfügbares *Plug-In* des Herstellers des *Real-Video-Servers* zu installieren. Da diese *Plug-In*-Komponente und die damit visualisierbaren *Streams* bei vielen Internetanwendern weit verbreitet ist, wird der Benutzer nur in seltenen Fällen diese Komponente nachträglich installieren müssen. Für diesen seltenen Fall kann automatisch die Komponente über das Internet geladen und installiert werden. Die Videodirektübertragung gestattet den vom Standort unabhängigen Besuch einer Vorlesung oder eines Seminars. Aufgrund der übrigen Funktionen der Kommunikationsplattform sind die Voraussetzungen geschaffen, den Teilnehmer in den Ablauf der Veranstaltung zu integrieren. Beispielsweise kann eine synchrone Verbindung über die *Chat*-Funktion hergestellt werden,

Lehrmaterialien können über die *File-Sharing*-Funktion übertragen werden, oder auch Termine, Projekte und Aufgaben in direkter Abstimmung aller Teilnehmer, vereinbart werden.

4.3.9 Kommunikationsplattform und *Web-Service* - Interoperabilität

Die Definition des Begriffs Interoperabilität (vgl. Kapitel 1) setzt eine sinnvolle Ergänzung der Funktionalitäten der Teilsysteme voraus. Die Interoperabilität zwischen der Kommunikationsplattform und dem radiologischen *Web-Service* führt zu einer konkreten, neuen Erweiterung des bislang auf bestimmte Funktionalität ausgerichteten Verwendungszwecks, zu einem umfassenden, integrierten Kommunikations-, Koordinations- und Lehrsystems. Die radiologische Lehrbildsammlung, nun gekapselt vom *Web-Service* mit seinen interoperablen Schnittstellenfunktionen, steuert die medizinischen Inhalte bei, die beispielsweise für die Lehre relevant sind. Um eine fachliche Diskussion einer Benutzergruppe zu realisieren, bietet sich als Ausgangspunkt das *Chat*-Modul an. Entsprechend der für die übrige Funktionalität der Kommunikationsplattform eingesetzten Programmiersprache, wird zur Integration der vom *Web-Service* angebotenen *Web Methods* eine *PHP*-basierte Bibliothek zur Nutzung des *SOAP*-Protokolls verwendet. Deswegen wird die *SOAPx4*-basierte *Client*-Funktionalität zur Kommunikation mit dem Web-Service genutzt. Die Kopplung der Kommunikationsplattform mit dem *Web-Service* ist für den Benutzer transparent. Daher werden die Eingaben vom Benutzer, die für die Parametrisierung der Interoperabilität notwendig sind, aus der bestehenden Benutzerschnittstelle abgeleitet. Hierzu bedarf es einer konsistenten Informationsquelle der aufgetretenen Ereignisse. Eine robuste Informationsquelle für die Verarbeitung chronologischer Ereignisse stellt eine Protokolldatei dar. Die *Chat*-Funktionalität der Kommunikationsplattform führt einen Mitschnitt der Eingaben aller Teilnehmer als gespeicherte Datei auf dem *Server*. Diese Dateien werden für die Diskussionen separat geführt und lassen sich den jeweiligen Gruppen eindeutig zuordnen.

Die Methode zur Schaffung der Interoperabilität zwischen der Kommunikationsplattform und dem *Web-Service* umfasst acht Phasen:

(1) Zyklisches Laden der Protokolldatei
(2) Identifizierung des Suchbegriffs in einer Befehlzeile
(3) Instanziierung des *SOAP*-Schnittstellenobjekts
(4) Festlegung der zu adressierenden *Web Method*

(5) Sendung der *SOAP*-Nachricht an den *Web-Service*

(6) Empfang und Dekodierung der *SOAP*-Antwortnachricht des *Web-Service*

(7) Referenzierung der radiologischen Bilder

(8) Integration der radiologischen Bilder in den gemeinsamen Sichtbereich aller Teilnehmer und zyklische Aktualisierung des gemeinsamen Sichtbereichs

In Phase (1) wird die Protokolldatei der jeweiligen Diskussionsgruppe bestimmt und in einem Datenarray geladen (Listing12):

```
// Im Gruppenmodus die entsprechende Gruppen-Datei auswaehlen:
01       if ($groups) {
02         $alivefile = $user_group."_".$alivefile;
03         $chatfile = $user_group."_".$chatfile;
04         $discussfile = $user_group."_".$discussfile;
05         $lastscanlinefile = $user_group."_".$lastscanlinefile ;
06       }
07         $lines = file($chatfile);
```

Listing 12: Identifizierung des Dateinamens der Protokolldateien auf dem *Server* entsprechend der jeweiligen Gruppe (Zeilen 01-06). Laden der Datei in ein Datenarray (Zeile 07). Dieses Datenarray ($lines) wird weiter verarbeitet in Listing 13

Die Phase (2), implementiert in Listing 13, analysiert die Datenzeilen sequentiell und vergleicht dabei jede Zeile mit dem Suchmuster „#“ , welches den Inhalt als potentielle Befehlszeile ausweist. Sofern das Kommandowort („KEY“) erkannt ist, wird zur Phase (3) verzweigt. Andernfalls wird erneut Phase (1) fortgesetzt.

```
01         for ($i = count($lines); $i >=0; $i--) {
02              $line=strip_tags($lines[$i]);
03              $pos = strpos ($line, "#");
04              if ($pos) {
05                     $rowparts = explode (":",$line);
06                     $commandparts= explode ("#", $rowparts[1]);
07                     switch (trim($commandparts[0])) {
08                        case ("KEY"):
09                         $url_arr=interopwebservicebykeyword ($commandparts[1]);
10                         $success=echodiscussimagetablefile ($url_arr, $discussfile);
11                         break;
12                        }}}
```

Listing 13: Die Befehlszeile wird durch eine Suchen nach dem Sonderzeichen „#“.identifiziert. Nach erfolgreicher Suche (Zeilen 04-14) wird eine Auftrennung der Befehlszeile (Zeilen 05-06) durchgeführt. Anschließend wird geprüft (Zeilen 07;08), ob es sich um ein gültiges Schlüsselwort handelt. Die nachfolgende Kommunikation (vgl. Listing 14) mit dem *Web-Service* wird mittels Suchbegriff (Zeile 09) parametrisiert. Die passenden Referenzen werden vom *Web-Service* als Rückgabewert der Funktion „interopwebservicebykeyword“ (Zeile 09) zur Darstellung mittels Funktionsaufruf „echodiscussimagetabelfile“ (Zeile 10) gebracht, die in Listing 15 aufgeführt ist.

Die Umsetzung der Phasen (3) bis (7) ist untrennbar miteinander verbunden und wird in der ummantelnden Funktion „interopwebservicebykeyword“ (Listing 14) realisiert. Die darin aufgerufene *Web Method* „GetURIMediaImageListbyKeyword“ (Zeile 07) liefert eine Referenzliste der radiologischen Bilder zum übertragenen Suchbegriff:

```
01       function interopwebservicebykeyword ($interopkeyword) {
02         global $ws_endpoint;
03         $url_arr = array ();
04         $ws_endpoint="http://tf-server.ukbf.fu-
berlin.de/Soapx4/UKBFMedicalKnowledgeBaseWebService.php";
05         $method_params["GetURIMediaImageListbyKeyword"]["keyword"] = $interopkeyword;
06         $soap = new soap_client($ws_endpoint);
07         $method="GetURIMediaImageListbyKeyword";
08         $soap_message = new soapmsg ($method,$method_params["GetURIMediaImageList-
byKeyword"] ,$method_namespace);
09         $return_val = new soapval;
10         $return_val = $soap->send($soap_message,"");
11         $decoded_val= $return_val->decode();
12         foreach($decoded_val as $k => $v){
13             foreach($v as $a => $b){
14                 $url_arr[]=$b;
15                 }
16             }
17       return (array) $url_arr;
18       }
```

Listing 14: Herstellung der Interoperabilität durch Instanziierung des *SOAP*-Schnittstellenobjekts (Zeile 06), parametrisiert mit dem Kommunikationsendpunkt (Zeile 04) des *Web-Services* und dem Suchbegriff als Übergabewert (Zeile 05). Die adressierte *Web Method* wird in Zeile 07 spezifiziert und geht als Parameter in die Instanzierung der *SOAP*-Nachricht ein. Die Sendung der *SOAP*-Nachricht an den *Web-Service* wird in Zeile 10 gestartet. Die als Rückgabewerte der Versendung empfangenen Daten werden in Zeile 11 dekodiert und die Referenzen auf die Bilder identifiziert (Zeilen 12-16). Schliesslich wir die Aufzählung der Referenzen an die aufrufende Funktion übermittelt (Zeile 17).

Die Darstellung und Aktualisierung der Bilder in einem für alle Teilnehmer sichtbaren Bereich (Phase (8)) erzielt die Funktion „echodiscussimagetablefile“ (Listing 15) mit Hilfe von *HTML*-Kommandos, die bei jedem Teilnehmer eine Tabelle mit den referenzierten Bilder darstellt:

```
01      function echodiscussimagetablefile ($url_arr, $discussfile) {
02        global    $chatfile,    $chatfreq,    $max_lines,    $css_style,    $bgcolor3,
               $HTTP_USER_AGENT;
03        echo  "<html><head>";
04        $chatfreq = $chatfreq/1000;
05        echo      "<meta      http-equiv=\"refresh\"      content=      \"$chatfreq;
        URL=discuss.php?mode = scancommand\">\n";
06        echo "<body>\n";
07        echo   "<P><TABLE   Class='images'   BORDER=2   CELLSPACING=2   CELLPADDING=2
        align=center>\n";
08        $count=0;
09        $col=1;
10        foreach($url_arr as $url_elem){
11              // schreibe img_src
12              if ($count==0){
13                echo "<tr Class='images'>";
14                }
15                echo "<td Class='images'>";
16                $rest = substr($url_elem, -3, 3);
17              if ($rest == "jpg") {
18                   echo "<img src='$url_elem'>";
19              }
20              $count++;
21              echo "</td>";
22              if ($count==$col){
23                  echo "</tr>";
24                  $count=0;
25                  }
26             }
27              echo "</table><P>";
28              echo "</body></html>\n";
29      return true;
30      }
```

Listing 15: Erzeugung der *HTML*-basierten Tabelle, in der die referenzierten Bilder angezeigt werden. Die zyklische Aktualisierung der Anzeige erfolgt über den Paramewter „refresh“ in Zeile 05. Nach Ablauf der Zeitfrequenz („$chatfreq“, Zeile 05), wird automatisch erneut die Funktion „scancommand“ aus Listing 13 aktiviert, und damit der Zyklus zur Phase 1 geschlossen.

Um den Zyklus bestehend aus Phase (1) bis Phase (8) zu schließen, wird mittels des *HTML*-Kommandos „refresh“ in Zeile 05 nach Ablauf einer einstellbaren Zeitdauer („$chatfreq“ in Zeile 05), erneut die Funktionalität der Phase (1) aufgerufen.