Integration eines Anwendungssystems für die Tumordokumentation in ein klinisches Informationssystem

Inaugural-Dissertation
zur Erlangung des Grades eines Doktors der Humanbiologie
des Fachbereichs Humanmedizin
der Justus-Liebig-Universität Gießen

vorgelegt von Iris Stolte

aus Frankenthal/Pfalz

Gießen 2001

Aus dem Medizinischen Zentrum für Ökologie
Institut für Medizinische Informatik

Leiter: Prof. Dr. J. Dudeck
des Universitätsklinikums Gießen

Gutachter: Prof. Dr. J. Dudeck

Gutachter: Prof. Dr. H. von Lieven

Tag der Disputation: 28.8.2001

1. Einleitung

1.1 Hintergrund

Krebserkrankungen haben eine vielseitige und komplexe Problematik. Daher ist eine interdisziplinäre und überregionale Zusammenarbeit in allen Bereichen der Vorsorge, Diagnostik, Therapie, Früherkennung, Nachsorge und Rehabilitation bei Krebserkrankungen zu gewährleisten. Für den Austausch und sinnvollen Vergleich der erfaßten Daten zwischen verschiedenen Tumorzentren sind bestimmte Standardisierungen unumgänglich. Dazu gehört eine einheitliche Nomenklatur und Klassifikation.

Eine Datenerfassung bei Tumorpatienten, die an das internationale Klassifikationssystem angelehnt ist, geht in Deutschland bis auf die Mitte der 70er Jahre zurück.
Mit Unterstützung der Bundesregierung wurde seit 1978 die Basisdokumentation für Tumorkranke durch die Arbeitsgemeinschaft deutscher Tumorzentren (ADT) und durch das Deutsche Krebsforschungszentrum (DKFZ) in den folgenden Jahren in verschiedenen Versionen veröffentlicht. Die fünfte Auflage der Basisdokumentation für Tumorkranke ist 1999 erschienen.

Seit 1991 wurde ein EDV-System für Tumorkranke, das Gießener Tumordokumentationssystem (GTDS), mit dem Ziel entwickelt, ein Werkzeug für die Erfassung und Verarbeitung der Daten der revidierten Basisdokumentation bereitzustellen. Inzwischen ist GTDS in über 30 Zentren - vor allem in den neuen Bundesländern - im Einsatz. Weiterentwickelt und gepflegt wird es vom Institut für Medizinische Informatik der Justus-Liebig-Universität, teilweise in Zusammenarbeit mit dem Informationszentrum für Standards in der Onkologie (ISTO).

Gegenwärtig wird das GTDS hauptsächlich von Dokumentaren in Klinischen Krebsregistern an Tumorzentren und Onkologischen Schwerpunkten und weniger direkt vom klinischen Personal verwendet. Das primäre Ziel dieser Einrichtungen ist, Information über den gesamten Verlauf dieser Krankheit, einschließlich Nachsorgebehandlung, zu sammeln, und die fachübergreifende Zusammenarbeit aller an der Versorgung von Krebskranken beteiligten medizinischen Disziplinen mittels eines umfassenden und zeitgerechten Informationsaustausches zu gewährleisten. Einheitlich geführte Krebsregister haben darüber hinaus eine gesundheitspolitische Bedeutung, da sie zentrumsübergreifende Datenerhebungen und -aus-

wertungen zu Fragen der Qualität und Effizienz der Krebsdiagnostik und -therapie ermöglichen [Dudeck 1994].

GTDS liefert spezifische Funktionalitäten, um die Aufgabe der Tumorzentren zu unterstützen. Obgleich von Anfang an Funktionen und Dienste integriert waren, um spezifische klinische Anforderungen gerecht zu werden, um so die individuelle Behandlung von onkologischen Patienten zu unterstützen, kann ein Registersystem wegen der unterschiedlichen Ausbildung, Arbeitsmethoden und Interessen von Ärzten im Vergleich zum Personal in Tumorzentren, nicht einfach an ein Krankenhausnetzwerk angeschlossen werden.

Dokumentare in Tumorzentren sind speziell qualifizierte Fachkräfte. Sie besitzen eine interdisziplinäre Ausbildung in Dokumentation, Medizin und Informatik und sind darauf spezialisiert, medizinische Daten zu erfassen, auszuwerten und zu archivieren. Demgegenüber sind Ärztinnen und Ärzte bisher eher sporadische Benutzer, d.h. keine Spezialisten für Dokumentationssysteme, so daß sie eine höher entwickelte Führung und Unterstützung benötigen.

Ärzte und Dokumentationspersonal verwenden Daten außerdem sehr unterschiedlich. Während Ärzte Daten während der Behandlung aufnehmen und überblicken, sammelt Registerpersonal Daten z.B. von gedruckten Dokumentationsformularen meistens nach dem die Diagnose, Behandlung oder Nachsorgeuntersuchungen abgeschlossen wurden. Diese retrospektive Erfassung schränkt den Umfang der zu dokumentierenden Daten wegen des nicht unwesentlichen zusätzlichen Aufwandes erheblich ein. Insbesondere für Aufgaben des Qualitätsmanagements, die in der Tumordokumentation immer stärker in den Vordergrund treten, ist es notwendig, umfangreiche, an Diagnose und Therapie der Patienten orientierte Daten zu erheben. Diese können verläßlich aber nur von den an der Behandlung beteiligten Ärzten und Schwestern während des Krankheitsverlaufs, d.h. in der Klinik, erhoben werden.

Im Gegensatz zu Dokumentaren, die eher allgemeine statistische Untersuchungen durchführen, hat das klinische Personal noch ein weitreichenderes wissenschaftliches Interesse beim Einsatz von Dokumentationssystemen. Durch individuelle, auf die klinischen Belange abgestimmte Erweiterungen können daher auch sehr differenzierte Erkenntnisse gewonnen werden.

Tumordokumentationssysteme wie das GTDS sind krankheitsspezifische Systeme (disease specific systems), die Daten von verschiedenen medizinischen Fachrichtungen sammeln. Durch den krankheitsspezifischen Fokus beschränken sie sich naturgemäß auf bestimmte Daten von Patienten mit onkologischer Relevanz. Ihre spezielle Funktionalität ist für gewöhnlich nicht auf allgemeine abteilungsspezifische Bedürfnisse ausgerichtet.

Einer weit verbreiteten Verwendung des GTDS im klinischen Umfeld standen daher die folgenden speziellen Hindernisse im Wege:

- die unzulängliche Zugangsmöglichkeit auf das Tumordokumentationssystem für das klinische Personal
- ein erhöhter Einarbeitungsaufwand
- die Beschränkung auf onkologische Patienten
- das Fehlen von abteilungsspezifischer Funktionalität und
- die Notwendigkeit teilweise redundante Daten einzugeben

1.2 Zielsetzung

Da das ärztliche Interesse an direkten Online Verwendungen solcher Anwendungssysteme für die Tumordokumentation wachsend ist, war es Ziel der Arbeit, GTDS zu modifizieren, zu erweitern und in das bereits bestehende Informationssystem des Universitätsklinikums Gießen zu integrieren. Zur Erreichung des Ziels wurde daher ein Verfahrensweg eingeschlagen, der sich durch die folgenden Punkte charakterisieren läßt:

1. Weiterentwicklungen zur Verbesserung der Benutzerakzeptanz

   Das GTDS hatte eher generischen Charakter und war daher nicht auf individuelle, benutzerspezifische Bedürfnisse ausgerichtet. Es sollte daher ein System entwickelt werden, das die immer wiederkehrenden urologischen Untersuchungen systematisch und lückenlos erfaßt sowie deren Verlauf dokumentiert. Die Anforderungen sollten mit der geeignetsten (Internet-)technologie in elektronische Formulare umgesetzt und sukzessive an die Wünsche der klinischen Anwender angepaßt werden. Dabei war eine enge Zusammenarbeit mit den zukünftigen Anwendern notwendig, um die Hindernisse, die dem Einsatz des GTDS in einer urologischen Abteilung entgegenstanden, zu beseitigen.
2. Evaluierung geeigneter (Internet-)techniken für die Realisierung

   Lösungen sollten, im Gegensatz zu lokal installierten Programmen, eine hohe Verfügbarkeit und leichte Zugangsmöglichkeit für das ärztliche Personal bieten. Daher sollten zunächst bestehende Internettechnologien für die Realisierung gegenübergestellt, Vor- und

Nachteile der in Frage kommenden Anwendungen gegeneinander abgewogen sowie mögliche Programmierumgebungen bezüglich ihrer Funktionalität praktisch erprobt werden.

3. Integration bereits vorhandener klinischer Daten und Funktionen

   Es sollte erreicht werden, daß die bereits von anderen Institutionen erfaßten und für das System relevanten Daten bereitgestellt und nicht erneut eingegeben werden müssen. Über eine HL7-Schnittstelle sollte die Konsistenz der Daten gewährleistet werden.

4. Prototypische Einführung des erweiterten GTDS-Systems in das klinische Informationssystem und dessen Bewertung im klinischen Einsatz

   Das System sollte in der Urologischen Ambulanz zur Verfügung gestellt und während der praktischen Nutzung über einige Monate hinweg analysiert werden, ob die urologischen Untersuchungen den Erfordernissen entsprechend erfaßt werden können und die gewünschte abteilungsspezifische Funktionalität integriert worden ist. Weiterhin sollte betrachtet werden, ob die Bereitstellung von bereits erfaßten Daten zuverlässig und korrekt funktioniert und ob das entwickelte System insgesamt übersichtlich und unmißverständlich, ob es intuitiv und ohne größere Einarbeitungszeit verwendbar ist.

5. Berücksichtigung gewisser Sicherheitskriterien bei der Entwicklung und Implementierung des Systems

6. Skizzierung, wie die aus der Erfahrung gemachten Probleme, unter Verwendung der gegenwärtig in Entwicklung befindlichen und zukünftig zur Verfügung stehenden Technologien, zu beseitigen sind

Kapitel 2: Das Gießener Tumordokumentationssystem (GTDS)

2.1 Die Basisdokumentation für Tumorkranke

Die Basisdokumentation für Tumorkranke umfaßt einen Mindestdatensatz der Dokumentation von Diagnose, Therapie, Verlauf und Abschluß der Tumorerkrankung, der bei jedem Patienten nach festgelegten Regeln erhoben wird. Diese Daten bilden die Grundlage für den Aufbau einer umfassenden Dokumentation von Tumorpatienten in Klinischen Krebsregistern an Tumorzentren und Onkologischen Schwerpunkten [Dudeck 1994].

Während epidemiologische Krebsregister der Morbiditäts- und Mortalitätsstatistik dienen, sollen Klinische Krebsregister die Betreuung von Krebspatienten verbessern helfen, indem die Zusammenarbeit zwischen den betroffenen medizinischen Fachbereichen bei der Krebsbehandlung erleichtert sowie die einheitliche Qualität der Versorgung von Tumorpatienten im gesamten Bundesgebiet gewährleistet wird. Dabei verfolgt die Erfassung und Speicherung der Daten der Tumorbasisdokumentation nach [Dudeck 1999] folgende Ziele:

- Dokumentation des individuellen Krankheitsverlaufs jedes einzelnen Patienten in einheitlicher und vergleichbarer Form
- Unterstützung der ärztlichen Tätigkeit und der Langzeitbetreuung des Patienten
- Unterstützung der Qualitätssicherung: Eine interne Qualitätskontrolle wird durch die Möglichkeit zur Bestimmung von Remissions- und Heilungsraten sowie Überlebenszeiten unterstützt. Eine externe Qualitätskontrolle wird durch Vergleiche mit Daten anderer Zentren und Angaben aus der Literatur ermöglicht.
- Durchführung deskriptiver und analytischer statistischer Auswertungen: Deskriptiv können Patientenkollektive nach verschiedenen Schichtungen wie Alter, Geschlecht, Tumorstadien, Therapie etc. ausgewertet werden. Mit Methoden der analytischen Statistik sind z.B. Vergleiche der Überlebenszeiten der Patienten verschiedener Gruppen möglich.

Bei allen statistischen Auswertungen muß jedoch beachtet werden, daß Daten Klinischer Krebsregister bezüglich der Zusammensetzung der Patientenkollektive in der Regel gewissen Selektionsmechanismen (z.B. der Spezialisierung der Zentren auf Therapieschwerpunkte) unterliegen, die die Verallgemeinerungsfähigkeit der Ergebnisse einschränken können. Bevölkerungsbezogene Aussagen wie Änderung der Häufigkeiten von Neuerkrankungen (Inzidenzen) sind aus Daten Klinischer Krebsregister nur in Ausnahmenfällen zu erhalten.

Seit dem Erscheinen der 1. Basisdokumentation für Tumorkranke im Jahre 1978 wurde diese einiger inhaltlicher Überarbeitungen unterzogen. So wurden in den nachfolgenden Neuauflagen jeweils Erfahrungen aus der Praxis eingearbeitet. Ab der 4. Auflage (1994) wurden zusätzlich die wesentlich weitgehenderen Möglichkeiten der modernen Datenverarbeitung berücksichtigt. Die Dokumentationsinhalte wurden so gestaltet, daß sie über eine patientennahe, behandlungsorientierte Erfassung der Daten stärker in den Behandlungsablauf integriert werden können und die Merkmale nicht mehr nachträglich, sondern behandlungsaktuell erfaßt und gespeichert werden können, so daß sie in weit stärkerem Maße als bisher zur Unterstützung der täglichen klinischen Tätigkeit (Behandlungsübersichten, Arztbriefschreibung) sowie zur Kommunikation zwischen den behandelnden Ärzten, Kliniken und Registern genutzt werden können. Weiterhin wurden die zu dokumentierenden Merkmale an die internationale Entwicklung der Tumordokumentation angepaßt.

Auch die 4. Auflage wurde kritisch evaluiert und die Ergänzungsvorschläge in die 5. Auflage (1999) eingefügt. Dazu gehören die Einführung neuer Schlüssel, Datensätze und Dokumentationsstandards. Neben den Wünschen der Zentren brachte das 1994 in Kraft getretene Krebsregistergesetz die Notwendigkeit der Überarbeitung der Basisdokumentation mit sich. Eine der Folgen dieses Gesetzes wird die verstärkte Zusammenarbeit zwischen klinischen und epidemiologischen Krebsregistern sein. Deshalb mußten die im Gesetz aufgezählten, von den bevölkerungsbezogenen Registern zu erfassenden Items, soweit sie bisher nicht schon erhalten waren, in die Basisdokumentation neu aufgenommen werden. Andere Items wie z.B. die Raucheranamnese wurden gegenüber der 4. Auflage gestrichen.
Um Kommunikation und Datenaustausch der klinischen Krebsregister untereinander sowie zwischen klinischen und epidemiologischen Registern auf elektronischem Wege zu unterstützen, wurde 1998 vom Zentralinstitut für Kassenärztliche Versorgung (ZI) in Köln standardisierte BDT- (Behandlungsdatenträger-) Schnittstellen an die Erfordernisse der Krebsdokumentation angepaßt. Die Satzarten und Felddefinitionen des onkologischen BDT sind mit der Basisdokumentation für Tumorkranke kompatibel [Dudeck 1999].

Die zu dokumentierenden Merkmale der Basisdokumentation lassen sich nach folgenden Gesichtspunkten ordnen [Dudeck 1999]:

- Allgemeine Identifikationsdaten (inkl. Patientenstammdaten)
- Daten über die Erkrankung und den Zustand des Patienten
  - in der prätherapeutischen Phase (Diagnosedaten)
  - im Verlauf (Verlaufsdaten)
  - bei Abschluß der Betreuung (Abschlußdaten)
  - bei Autopsie bei Tod des Patienten (Autopsiedaten)
- Daten über vorgesehene und durchgeführte Maßnahmen (Therapiedaten, Schmerzdokumentation)
- organisatorische Daten (Einbestelltermin etc.)
- Daten zum sozio-ökonomischen Status
- Daten zur Lebensqualität
- Daten zur palliativ-onkologischen Versorgung

Die einheitliche Struktur des Datensatzes soll die Vergleichbarkeit der an den Zentren erhobenen Daten gewährleisten. Über diesen Mindestdatensatz hinaus kann jedes Zentrum weitere Daten erheben, die für organisatorische Aufgaben und wissenschaftliche Problemstellungen benötigt werden.

2.2 Funktionsumfang des Gießener Tumordokumentationssystems

Der Datenstandard der „Basisdokumentation für Tumorkranke“ diente zunächst als inhaltliche Grundlage für das Datenmodell des Gießener Tumordokumentationssystems. GTDS wird an Klinischen Krebsregistern und den mit diesen über Netzwerk verbundenen klinischen Abteilungen an Tumorzentren und Onkologischen Schwerpunkten unterschiedlicher Größe eingesetzt. Folgende Funktionen sind nach [Altmann 1999] im GTDS realisiert:

Dokumentation

- Diagnose
- Therapie
  - operativ
  - Bestrahlung
  - internistisch
- Verlauf
- Abschluß

- abteilungs- oder organspezifische Erweiterungen
- Lebensqualität und Schmerzdokumentation

Follow-up und Nachsorgemanagement

- Definition von Nachsorgeschemata
- Terminplanung
- Erinnerungsbriefe
- Individuelle Dokumentationsbögen
- Rücklaufkontrolle
- Nachfragen bei Hausarzt oder Meldeamt

Registerorganisation

- Statistiken
- Vergütung

Datenaustausch

- Gemeinsames Krebsregister
- Melanomregister
- andere Register, Praxissysteme (BDT)
- Klinikinformationssysteme (z.B. HL7)

Um die angestrebte klinische Unterstützung zu erreichen, wurden jedoch erhebliche Erweiterungen vorgenommen. So wurde die zusätzliche Eingabemöglichkeit von Freitexten zur Verfügung gestellt, um der häufig nicht schematisierbaren klinischen Situation gerecht werden zu können oder die Möglichkeit zur Definition eigener Merkmalskataloge gegeben, um nicht standardisierte Verlaufsparameter (wie zum Tumormarker) erfassen zu können [Altmann 1998].
GTDS bietet zusätzlich zahlreiche Funktionen, die die Routinearbeit der onkologischen Versorgung unterstützen [Altmann 1999]:
Briefschreibung

- Übersichtsberichte
- Arztbriefe

Therapiemanagement

- Definition von Protokollen
- Berechnung angepaßter Einzeldosen
- Übersicht über verabreichte Gesamtmengen
- Definition von Therapieplänen

Statistiken

Studienmanagement

- Studiendaten (Beschreibungen)
- Adressen
- Studienarme und -schichten
- Eingangskriterien
- teilnehmende Patienten

Sofern ein Netzwerkzugriff möglich ist, können die angeschlossenen Abteilungen Daten selbst eingeben und die damit mögliche Funktionalität direkt nutzen, so daß die Integration der Tumordokumentation in die klinischen Routine unterstützt wird und damit deren Akzeptanz gesteigert wird [Altmann 1998].

3. Evaluierung geeigneter Internettechniken für die Realisierung

3.1 Problematik

Das klassische GTDS ist durch eine Client/Server-Architektur gekennzeichnet, was zur Folge hat, daß jeder potentielle Benutzer von GTDS umfangreiche Oracle Tools (Developer/2000) auf seinem Rechner installiert und Zugriff auf eine Datenbank haben muß. Dies bedeutet, daß nicht nur erhebliche systemtechnische Voraussetzungen für ein solches System vorhanden sein mußten, sondern auch ein erheblicher Zeit- und Kostenaufwand für die Installation, Wartung, Upgrading etc. pro Benutzer zu leisten sind.

Zudem sind die Computersysteme der verschiedenen Kliniken sehr heterogen, so daß die Verbreitung eines bestimmten Anwendungsprogramms auf die verschiedenen Rechner und Betriebssysteme häufig sehr schwierig und nur unter erheblichem Anpassungsaufwand möglich ist. Im Extremfall kann dies dazu führen, daß das Programm für die Installation auf einem anderen System komplett neu kodiert werden muß, oder aber daß eine minimale Anpassungsentwicklung Wochen dauern kann. So ist GTDS z.B. für MS Windows-Systeme optimiert.

Diese Problematik läßt sich vermeiden, indem man eine effiziente Entwicklungsumgebung aufbaut, die bewußt auf plattformunabhängige Standards setzt.
Mit den Möglichkeiten des World Wide Web (WWW) und entsprechenden Internet- und Intranet-Systemarchitekturen können die Anforderungen für den Betrieb des GTDS dramatisch verringert werden.
Durch die Verwendung eines graphischen Browsers als einfach zu bedienende, homogene Benutzerschnittstelle wird eine plattformunabhängige Programmausführung über das Web möglich. Gleichzeitig entfällt der Aufwand für die Distribution und Installation der Software.

Darüberhinaus bietet das Internet eine hohe Verfügbarkeit und leichte Zugangsmöglichkeit für das klinische Personal. Die Tumordokumentation soll an allen klinischen Arbeitsplätzen erfolgen können und die vom GTDS gebotenen unterstützenden Hilfsmittel für die ärztliche Tätigkeit, wie Berichterstellung, Generierung von Arztbriefen und Benachrichtigungen etc. sollen dort ebenfalls verfügbar sein.

3.2 Anforderungen an eine Web-DB-Anbindungsarchitektur

Das System bzw. dessen Entwicklung und Wartung muß gewissen Anforderungen entsprechen. Diese sollen zunächst genau spezifiziert werden. Da für die Realisierung prinzipiell mehrere verschiedene Web-DB-Anbindungsarchitekturen, also Architekturen, die den Zugriff auf ein Datenbanksystem über das Inter- oder Intranet ermöglichen, zur Verfügung stehen, sollen die Möglichkeiten bezüglich der unten genannten Anforderungen in den folgenden Abschnitten evaluiert werden.

- **Hohe Produktivität bei der Anpassung**

  Damit effiziente Adaptationen der Applikationen und damit eine rasche Anpassung an neue Erkenntnisse aus der Forschung und Erfahrungen aus der Praxis möglich sind, sollte der Programmieraufwand nicht zu hoch sein.

- **Hohe Portabiliät**

  Es sollte keine oder nur eine geringe Abhängigkeit vom Browser und vom Betriebssystem bestehen, bzw. sollte Browser und Betriebssystem sehr weit verbreitet sein. Um eine leichte Distribution und Installation des Systems zu erreichen, setzt man bewußt auf plattformunabhängige Tools.

- **Volle Funktionalität einer Programmiersprache**

  Die Eingabevalidierung und Verarbeitung der Daten aus einer Datenbank bedeutet, daß eine Programmiersprache mit einer hoch entwickelten Funktionalität verwendet werden muß.

- **Performance**

  Für den Benutzer sollten keine langen Wartezeiten beim Laden der Anwendung entstehen bzw. sollte das Navigieren innerhalb der Applikation ohne zeitliche Verzögerungen durchgeführt werden können.

- **Das System muß hohen Sicherheitskriterien genügen**

  Webbasierte Anwendungen sind aus verschiedenen Gründen mit einem höheren Sicherheitsrisiko verbunden als Anwendungen auf Rechnern, welche nicht mit einem Netzwerk verbunden sind. Sie verwenden komplexe Dienste und Sicherheitslücken in diesen Diensten wirken sich direkt auf die Anwendungen aus.

3.3 WWW-Grundlagen

Das Internet in seiner technischen Grundform existiert schon seit den 60er Jahren. Ursprünglich vom US-Militär geplant und als ARPA-Net realisiert begann die wissenschaftliche und damit zivile Nutzung etwa zehn Jahre später. Markant für das Internet war die fehlende zentrale Kontrolle. Der Verbund einzelner Rechner zu Netzen führte zu einem schnellen Wachstum des Internets. Rechnernetze wurden im Verbund mit anderen Netzen zu Subnetzen. Die technische Basis dieser plattformunabhängigen Rechnerkommunikation regelt sein Anbeginn das Protokoll TCP/IP (Transmission Control Protocol/Internet Protocol). Häufig werden die Begriffe WWW und Internet synonym verwandt [Rahm 1999]. Dabei umfaßt das Internet eine Reihe von Diensten, die es bereits vor dem WWW gab, wie beispielsweise FTP (File Transfer Protocol), für den einfachen Austausch von Dateien von Computer zu Computer, Telnet für den Zugriff auf andere Rechner über einen lokalen Computer, Gopher oder NetNews. Das Internet hat sich erst seit wenigen Jahren vom universitären Netz in ein weltumspannendes Gewebe aus Information, Unterhaltung und Kommerz entwickelt. Das WWW ist aufgrund seiner einfachen Handhabung, graphischen Browser und dem einheitliche Zugang zu vielen Online-Informationsquellen mittlerweile der populärste Dienst im Internet.

Zentrales Element des WWW ist das multimediale Dokument. Die auf vielen Hardware-Plattformen verfügbaren Browser formatieren ein mit HTML (Hyptertext Markup Language) beschriebenes Dokument, auf das über einen Verweis (Uniform Resource Locater, URL) zugegriffen wird. Ein URL hat die Form Dienst://Server/Verzeichnis/Datei. Im Hypertext enthaltene Verweise auf andere Dokumente oder Ressourcen vernetzen Dokumente miteinander. Da im URL auch viele andere Internetdienste verwendet werden können, wurde so eine einheitliche Schnittstelle zum Internet geschaffen.

WWW-Klienten kommunizieren mit einem Web Server über das Protokoll HTTP (Hyper Text Transfer Protocol). Der Server liest die angeforderte Information aus einem Dateisystem und überträgt die Datei. Sie wird von einer Formatbeschreibung mit MIME (Multipurpose Internet Mail Extensions) angeführt. Der Browser entscheidet daraufhin, ob er die gesendete Bytefolge selbst interpretiert (z.B. MIME-Type text/html) oder andere Hilfsanwendungen (Helpers) und Zusätze (Plugins) zur Interpretation aufruft.

Durch Programme unterhalb des Web Servers können anstelle statischer Dateien veränderliche Verweise oder dynamische Dokumente erzeugt werden [Benn 1998].

3.4 Einteilung der Web-DB Anbindungsarchitekturen

In den folgenden beiden Abschnitten soll eine Gegenüberstellung der bei webbasierten Datenbankanwendungen in Betracht kommenden Techniken vorgenommen werden. Die einzelnen Techniken werden kurz beschrieben, deren Vor- und Nachteile dargestellt sowie anhand des Forderungskatalogs bewertet. Darüberhinaus werden die mit diesen einhergehenden Sicherheitsprobleme diskutiert. Die Möglichkeiten weiterer Entwicklungen soll ebenfalls angesprochen werden.

Web-DB-Anbindungsarchitekturen können in Abhängigkeit ihres Ausführungsortes in zwei Gruppen unterschieden werden: serverseitige (HTTP-basierte) und clientseitige (siehe Abbildung 1). Serverseitige Anwendungen sind für die Erzeugung bzw. Bereitstellung der Daten verantwortlich. Clientseitige Anwendungen sind für die Darstellung der Daten und die Benutzerinteraktionen zuständig [Turau 1999].

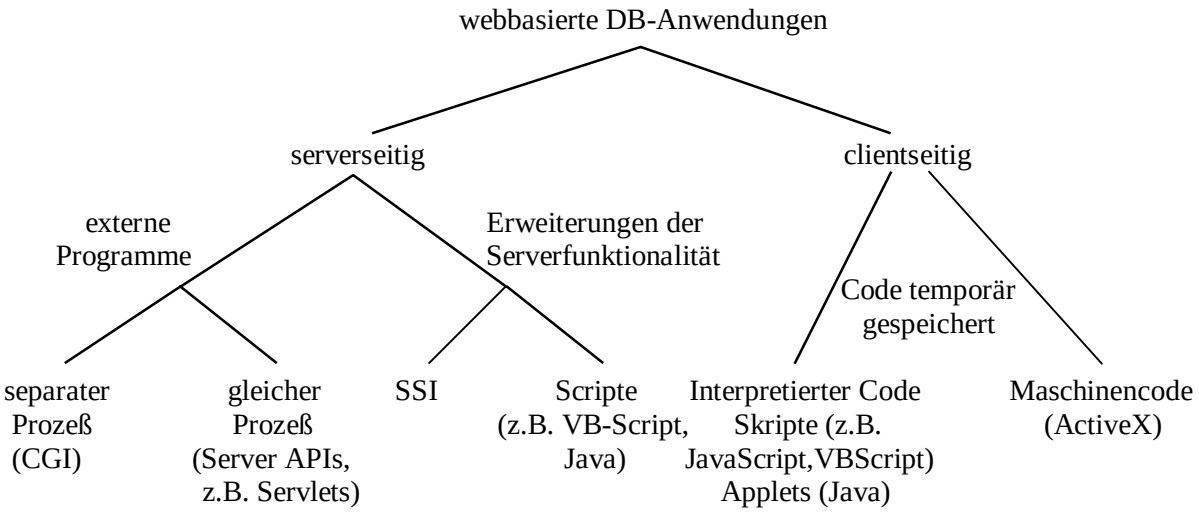

Abbildung 1: Klassifikation webbasierter DB-Anwendungen
Web-DB-Anbindungsarchitekturen können in Abhängigkeit ihres Ausführungsortes in zwei Gruppen unterschieden werden: serverseitige (HTTP-basierte) und clientseitige. Serverseitige Anwendungen werden dabei wiederum nach externen Programmen, welche von Servern gestartet werden, und Erweiterungen der Serverfunktionalität unterteilt.

Es sollen zunächst die serverseitigen, danach die clientseitigen DB-Anbindungsarchitekturen betrachtet werden.

3.5 Serverseitige Anwendungen (HTTP-basierte Ansätze)

Web Server sind für die Bereitstellung der Daten zuständig. Ursprünglich waren alle Daten in Dateien abgespeichert. Serverseitige Anwendungen ermöglichen den Zugriff auf Daten, welche nicht statisch vorliegen und nicht mit HTML ausgezeichnet sind. Die Daten können dynamisch erzeugt werden oder eben aus komplexen Informationsquellen (wie z.B. Datenbanken) entnommen werden. Bei serverseitigen Anwendungen muß zwischen externen Programmen, welche von Servern gestartet werden, und Erweiterungen der Server-funktionalität unterschieden werden [Turau 1999].

3.5.1 Externe Programme

3.5.1.1 Common Gateway Interface (CGI)

CGI ist eine Schnittstelle für die Kommunikation zwischen Web Servern und Anwendungs-programmen. Die vom Benutzer in einem HTML-Formular eingegebenen Parameter werden mit Hilfe des CGI an ein Anwendungsprogramm, ein CGI-Programm, das in einer beliebigen serverseitig unterstützten Programmier- oder Scriptsprache (z.B. Perl, Shell-Skripte, C/C++) implementiert ist, auf dem Web Server übergeben. Dieses Programm wird als Prozeß auf dem Web Server gestartet und kann nun als DB-Client auf die Datenbank zugreifen [Rahm 1999].
Während der Laufzeit des Programms ist nun ein vollständiges HTML-Dokument zu erzeugen, das an den wartenden Web Server zurückgegeben wird. Dieser reicht schließlich das Dokument als Ergebnis der HTTP-Anfrage zur Anzeige an den Web Browser zurück [Loeser 1998]. Danach wird der Prozeß beendet, d.h. es wird für jede Anfrage ein neuer Prozeß gestartet.

Der Vorteil einer selbstentwickelten, ausprogrammierten Lösung liegt in der Möglichkeit, das jeweilige Programm direkt auf spezifische Anforderungen zuschneiden zu können. Ein weiterer Vorteil ist auch die Unterstützung durch alle Web Server, Web Browser und Betriebssysteme. Allerdings sind mit diesen Vorteilen zahlreiche Nachteile verbunden. So ist die Programmerstellung recht aufwendig, da keine CGI-Entwicklungsumgebung existiert, das Debugging problematisch und die Parameterübergabe umständlich. Auch ist die Formatierung des Dokuments erst nach der Programmausgabe überprüfbar [Rahm 1998].

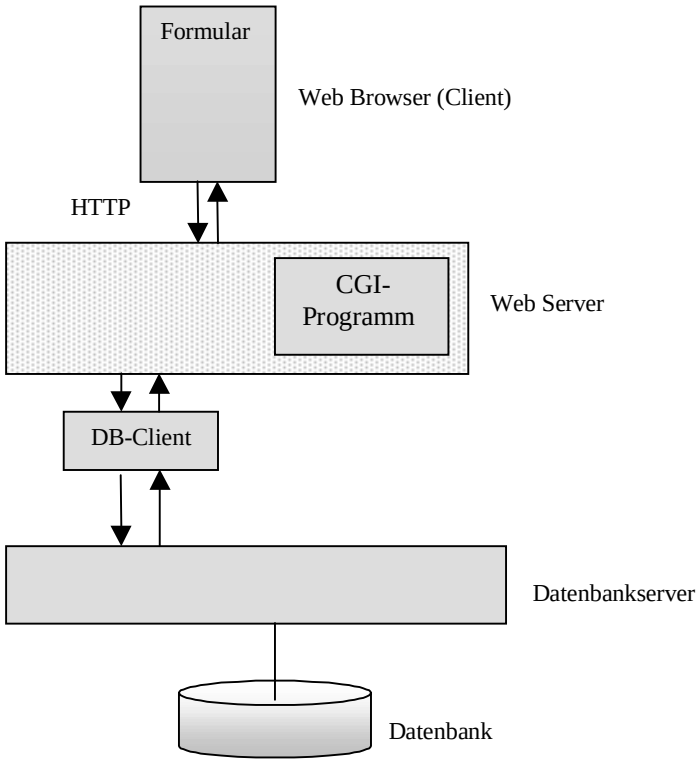

Abbildung 2: Ablauf einer CGI-Datenbankanwendung
Eine CGI-Schnittstelle realisiert den einfachsten Übergang zwischen einem Web Server und einer Datenbank. Die vom Benutzer in ein HTML-Formular eingegebenen Parameter werden mit Hilfe des CGI an ein Anwendungsprogramm, ein CGI-Programm, auf dem Web Server übergeben. Dieses Programm wird als Prozeß auf dem Web Server gestartet und kann nun als DB-Client auf die Datenbank zugreifen. Die Datenbank ist eben eine Datenquelle, aus der durch eine einzelne Anfrage eine Menge von Daten geholt, für die Ausgabe nach HTML formatiert und zum Browser geschickt werden. Danach wird der Prozeß beendet, d.h. es wird für jede Anfrage ein neuer Prozeß gestartet. Änderungen der angezeigten Daten entsprechen eigenständigen Anfragen.

Desweiteren erzeugt der Web Server für jede CGI-Anfrage einen eigenen Prozeß. Der mit jeder Anfrage einhergehende Verbrauch von Betriebssystemressourcen kann sich negativ auf die Verfügbarkeit des Servers auswirken [Turau 1999]. Bei jedem Prozeß wird eine neue Verbindung zur Datenbank aufgebaut. Zudem werden logische Formular-Eingabefehler erst im CGI-Programm erkannt und erfordert zeitintensive Serveraktion. Auch sind selbst programmierte Lösungen (anfangs) in der Regel fehlerhafter als „Standardwerkzeuge“ für die Webanwendung.

Fazit: CGI-Programme sind eigentlich nur für den gezielten, transaktionslosen DB-Zugriff bei geringem Datenaufkommen und kurzer CGI-Prozeßlaufzeit geeignet [Rahm 1998].

3.5.1.2 Server APIs

Application Programming Interfaces (APIs) integrieren das CGI-Konzept in den Web Server. Ähnlich dem Prinzip der Dynamic Link Libraries (DLL) werden funktionale Erweiterungen zur Laufzeit in den Web Server eingebunden. Dadurch wird verhindert, daß wie bei CGI-Programmen ein eigener Prozeß gestartet wird. Diese Erweiterungen sind auch gegenüber CGI-Programmen funktional nicht eingeschränkt.
Der Web Server muß anhand der URL entscheiden, ob für die Abarbeitung der Client-Anfrage eine Erweiterungsfunktion aufgerufen werden muß. Dieser Aufruf geschieht als prozeßinterner Funktionsaufruf, wodurch ein Performancegewinn gegenüber der CGI-Lösung erreicht wird. Ein weiterer Vorteil ist, daß Servererweiterungen nach der HTTP-Verbindung nicht wie CGI-Prozesse terminiert werden und somit weiterlaufen können. Damit können Datenbankverbindungen permanent offengehalten werden [Rahm 1999].

Nachteilig wirkt sich aus, daß der Aufwand zur Entwicklung von Server API Anwendungen oft höher ist als der für CGI-Anwendungen. Die Programmerzeugung ist umständlich, das Debugging problematisch. Auch besteht eine Abhängigkeit von API-kompatiblen Web Servern. Darüberhinaus besteht die Gefahr des "Hängenbleibens" offener Datenbanktransaktionen durch noch nicht beendete Sitzungen.

Etabliert haben sich Internet Server API von Microsoft (ISAPI) sowie die Netscape Server API (NSAPI), wobei beide nicht kompatibel sind.

3.5.1.3 Persistente Prozesse

Eine Alternative zu den von Server APIs verwendeten dynamisch ladbaren Bibliotheken sind persistente Prozesse. Ein Web Server startet beim ersten Aufruf einen Prozeß für die Anwendung. Im Gegensatz zu CGI-Anwendungen wird dieser Prozeß nach Abarbeitung der Anfrage aber nicht beendet, sondern kann für weitere Anfragen genutzt werden. Man erzielt so eine bessere Performance als reine CGI-Anwendungen und hat den Vorteil der Prozeßisolation. Für die Kommunikation wird ein eigenes Protokoll benötigt. Fast CGI ist eine Realisierung dieser Technik [Turau 1999].
Das Problem der Performance wurde mit diesem Ansatz gelöst, doch bleiben die anderen Nachteile des reinen CGI bestehen.

3.5.1.4 Servlets

Die Java-Servlet API wurde als Java-Pendant zu den oben beschriebenen APIs entwickelt. Neben der Plattformunabhängigkeit von Java bieten Servlets noch andere Vorteile. Die Entwicklung von Anwendungen wird erleichtert, da die wichtigsten Strukturen als Objekte zur Verfügung stehen. Da Java Klassen dynamisch geladen werden, funktionieren Servlets ähnlich wie dynamisch ladbare Bibliotheken. Sie werden beim ersten Aufruf geladen und verbleiben dann im Speicher. Es kann eine beliebige Anzahl von Anfragen gleichzeitig bearbeitet werden. Das Erstellen von Servlets ist relativ einfach, da das Servlet API bereits viele Basisklassen enthält, welche das gleichzeitige Ausführen mehrerer Anfragen unterstützen. Servlets bilden eine Alternative zu CGI Anwendungen und zu Server Side Includes (siehe weiter unten). Eine Einbettung von Servlets erfolgt mit dem Auszeichnungselement <SERVLET>. Dieses Element erlaubt u.a. die Angabe einer URL eines Servlets. Der Server parst ein HTML-Dokument, führt referenzierte Servlets aus und ersetzt die Auszeichnungselemente durch die entsprechende Ausgabe.
Ein Vorteil von Servlets gegenüber Server API-Anwendungen ist, daß der Code von Servlets auch von einem anderen Rechner geladen werden kann. Servlets werden mittlerweile von einigen Web Servern unterstützt [Turau 1999].

3.5.2 Erweiterungen der Serverfunktionalität

Es werden spezielle Anweisungen in die HTML-Syntax integriert, die vom Server ausgewertet werden [Rahm 1999]. Die einzelnen Verfahren unterscheiden sich in der Mächtigkeit der verwendeten Sprache für die Anweisungen. Die einfachste Variante ist SSI (Server Side Includes). Zwar ist die Bandbreite der Anweisungen in der Regel relativ klein und hängt vom Web Server ab, da jedoch Aufrufe von Betriebssystemkommandos möglich sind, können beliebige Anwendungen gestartet werden. Server Side Includes bilden die einfachste Art von Server-Erweiterungen, denn es können nur einzelne Anweisungen eingebettet werden. Komplexe Anwendungen erlauben die Verwendung von vollständigen Programmiersprachen (z.B. VBScript, JavaScript oder Java). Diese werden dann als serverseitge Skripte oder auch als aktive Server-Seiten bezeichnet. Ein Web Server enthält in diesen Fällen einen Interpreter, welcher in HTML-Dokumente eingebettete Programme interpretiert und die Ausgabe in die Dokumente einfügt. Solche Anwendungen hängen stark von der Server-Software ab und sind oft für dezidierte Aufgaben wie den Zugriff auf Datenbanken konzipiert.
Da das Analysieren der Dokumente die Leistung des Servers negativ beeinflußt, werden solche Einfügungen in Dokumente nur auf Anforderung durchgeführt. Ein Server entscheidet

an der Endung eines Dateinamens, ob der Inhalt eines angeforderten Dokumentes auf Anweisungen analysiert wird.
Von Vorteil ist die einfache Handhabung. Server Side Includes und serverseitige Skripte können direkt in ein HTML-Dokument eingebettet werden. Alle Anwendungen dieser Kategorie belasten jedoch den Server und verzögern die Auslieferung von Dokumenten [Turau 1999].

3.5.3 Sicherheit serverseitiger Anwendungen

Bei der Sicherheit sind die Übertragungssicherheit, d.h. die Abhörsicherheit, und der Zugriffschutz des Web Servers zu unterscheiden.
Um einen eingeschränkten Zugriff auf bestimmte Inhalte eines Web Servers zu realisieren, können die sich ergänzenden Verfahren der HTTP-Authentisierung sowie das Einschränken des Verbindungsrechtes auf bestimmte Internet-Adressen bzw. Domains angewendet werden. Bei letzterem kann über eine entsprechende Konfiguration des Web Servers festgelegt werden, von welchen Internetadressen aus auf den Web Server oder bestimmte Unterverzeichnisse zugegriffen werden darf bzw. von welchen Internet-Adressen aus dies generell verboten ist (allow/deny), um so die Zugriffsmöglichkeit auf eine Abteilung oder ein Bereich zu begrenzen und einen Intranet-Server zu realisieren. Damit indirekt verbunden ist auch die Einschränkung eines potentiellen webbasierten Angriffs auf DB-Server.
Mit dem Verfahren der HTTP-Authentisierung kann der Zugriff auf Unterverzeichnisse oder den ganzen Server auf bestimmte Benutzer eingeschränkt werden. Greift ein Benutzer auf geschützte Bereiche (Domain) eines Web Servers zu, so muß er sich über Benutzernamen und Paßwort authentisieren.
Da die Übertragung von Paßwort-Informationen und anderen vertraulichen Daten unter Nutzung des HTTPs nicht sicher ist, bedarf es des Einsatzes eines Verschlüsselungsverfahrens. Existierten anfangs SHTTP (Secure HTTP) und SSL (Secure Socket Layer) so hat sich wegen seiner Praktikabilität SSL durchgesetzt und ist heute de facto das alleinige Verfahren. SSL basiert auf dem symmetrischen RSA-Verfahren und benötigt auf Seiten des Web Servers ein elektronisches Zertifikat. Ist der Aussteller dem Browser nicht bekannt, z.B. weil sich der Betreiber des WWW-Servers selbst zertifiziert, so wird der Benutzer in der Regel gefragt, ob er dem Anbieter trauen möchte. Lehnt er dies ab, wird keine verschlüsselte Verbindung zum Web Server aufgebaut.
Entscheidender Nachteil von SSL sind die aufgrund der US-amerikanischen Exportrestriktionen kurze Schlüssellängen, die den Einsatz für Anwendungen mit besonderen

Sicherheitsanforderungen außerhalb der US unbrauchbar macht. Deshalb sind Ergänzungen der im Browser vorhandenen Sicherheitsmaßnahmen nötig, um solche Anwendungen zu realisieren (siehe weiter unten) [Loeser 1998].

Serverseitige Anwendungen verarbeiten häufig Benutzereingaben aus HTML-Formularen. Hierbei besteht die Gefahr, daß Anwendungen diese Eingaben nicht korrekt verarbeiten und dazu verführt werden, Kommandos auszuführen, die eigentlich nur autorisierten Benutzern vorbehalten sind. Eine von vielen Quellen für dieses Problem liegt in Programmen, bei denen es zu Überläufen von Puffern beim Einlesen von Benutzereingaben kommt. Wird die Größe eines solchen Puffers nur an der zu erwartenden Eingabe eines idealen Benutzers orientiert, so kann eine weit über diese Größe hinausgehende Eingabe die Anwendung zum Absturz bringen. Unter Umständen hinterläßt ein solcher Absturz einen Rechner in einem Zustand, welcher die Eingabe beliebiger Kommandos erlaubt. Sicherheitslücken, die auf derartige Programmierfehler entstehen, sind nicht auf serverseitige Anwendungen beschränkt.
Ein andere potentielle Gefahr bei allen serverseitigen Anwendungen liegt in der Übergabe von unkontrollierten Benutzereingaben an Systemkommandos. Enthalten solche Eingaben anstelle von den erwarteten Daten Betriebssystemkommandos, so kann dies die Sicherheit des Servers gefährden [Turau 1999].

3.5.4 Bewertung der serverseitigen Web-DB Anbindungsarchitektur

Der Vorteil der HTTP-basierten Architekturen liegt in den einfachen HTML-Gestaltungsmöglichkeiten für die Datenpräsentation, und darin, daß kleine Anwendungen schnell entwickelt werden können. Aufgrund des breiten Spektrums an bereits vorhandenen Web-DB-Anwendungen, kann hier auf einen großen Erfahrungsschatz zurückgegriffen werden [Rahm 1999].

Allerdings bestehen Verbindungen zwischen Client und Web Server nur für die Zeit der HTTP-Anfragebearbeitung, d.h. HTTP-Transaktionen sind zustandslos. Eine Anfrage weiß nichts von den vorherigen Anfragen des gleichen Clients. Außerdem kostet der Neustart von CGI-Prozessen und das Wiederöffnen von Datenbankverbindungen Ressourcen und führt daher zu Performanceverlusten. Mehrschritt-Arbeitsgänge sowie die Realisierung von Zuständen können nur über Zusatzmaßnahmen wie z.B. Formularvariablen, HTTP-Cookies oder HTTP-Authentisierung erreicht werden. Die in den letzten Abschnitten diskutierten Techniken erleichtern solche Anwendungen. Sowohl Servlets als auch Server API Anwendungen haben einen internen Zustand und ermöglichen somit komplexe kooperative

Anwendungen. So können z.B. Verbindungen zu Datenbanken über einzelne Anfragen hinweg offen gehalten und Anwendungen zur Verfügung gestellt werden. Erst durch die Kombination von client- und serverseitigen Techniken werden umfangreiche sitzungsorientierte Anwendungen ermöglicht [Turau 1999].

Ein generelles Problem, das allen Verfahren zur Realisierung von Zuständen anhaftet, ist die Frage nach geeigneten Timeout-Werten zur serverseitigen Unterbrechung von Web-Sitzungen bei Untätigkeit des Benutzers. Ein zeitgebundener serverseitiger Abbruch ist wichtig, um nicht benötigte Ressourcen (z.B. Kontextdaten in der DB) wieder freizugeben. Die Timeout-Werte hängen dabei sehr stark von der jeweiligen Anwendung ab [Loeser 1998].

3.6 Clientseitige Anwendungen

Bei serverseitigen Anwendungen wird ein Browser im Prinzip nur zur Ein- und Ausgabe verwendet. Zur Reduktion der Netzbelastung und Verbesserung der Antwortzeiten ist z.B. eine Validierung von Eingaben auf Clientseite wünschenswert. Mit HTML ist dies nicht zu erreichen. Aus diesem Grund wurden Techniken für clientseitige Anwendungen entwickelt.

3.6.1 Interpretierter Code

3.6.1.1 Scripte

Die Skripte clientseitiger Anwendungen können sowohl in HTML-Dokumente eingebettet oder auch separat übertragen werden. Ein Browser führt ein Script entweder direkt beim Laden des Dokumentes oder aufgrund eines vom Browser erzeugten Ereignisses aus (z.B. Abschicken eines Formulares oder Anforderung eines Dokumentes). Der HTML 4.0 Standard ist unabhängig von speziellen Skriptsprachen gehalten, es wird lediglich die Form der Einbettung von Skripten und eine Liste von Ereignissen festgelegt.
Eine der wichtigsten Anwendungen von Skripten besteht in der Validierung von Benutzereingaben in Form-Elemente. Dies erleichtert serverseitige Anwendungen, denn eine eventuell zusätzliche Kommunikation mit dem Server bei fehlerhaften Eingaben entfällt. Die Möglichkeit von Skript-Sprachen gehen aber viel weiter als die Validierung von Eingaben. Sie können den Inhalt, die Struktur und die Präsentation von Dokumenten dynamisch verändern.
JavaScript ist eine einfache objektorientierte Sprache basierend auf Prototypen. Es ist nicht möglich, auf Dateien zuzugreifen oder Netzwerkverbindungen aufzubauen.

Neben JavaScript werden noch VBScript und Tcl häufig für clientseitige Anwendungen verwendet. Entweder sind die zugehörigen Interpreter in die Browser integriert oder sie werden durch Plug-ins realisiert [Turau 1999].

3.6.1.2 Applets

Java Applets werden ähnlich wie HTML-Dokumente, in die sie eingebettet sind, auf einem Web Server bereitgestellt und vor der Programmausführung als Teil einer HTML-Seite in einen Java-fähigen Web-Browser eingeladen [Loeser 1998]. Innerhalb von HTML-Dokumenten steht einem Applet ein rechteckiger Bereich zur graphischen Ausgabe zur Verfügung. Viele Browser enthalten mittlerweile Java-Interpreter und die Klassen der Standard-APIs. Java-Quellcode wird von einem Compiler in ein Binärprogramm (Bytecode) übersetzt. Das Binärprogramm einer übersetzten Java-Anwendung wird in einer Datei mit der Endung .class abgelegt. Dieses wird von einem Interpreter ausgeführt, wobei die verwendeten Klassen analog zu Scriptsprachen dynamisch geladen werden (von der Festplatte über das Netzwerk). Um die Ausführungsgeschwindigkeit zu steigern, werden sogenannte just-in-time Compiler eingesetzt. Diese übersetzen den Bytecode vor der Ausführung in Maschinencode [Turau 1999].
Durch die beim Kompilieren erfolgende Überführung jedes Interface und jeder Klasse in je eine Class-Datei, muß beim Laden eines Applets über das Netz daher für jede von der Anwendung benötige Klasse bzw. Interface die entsprechende Class-Datei angefordert werden. Um den dazu jeweils erforderlichen (teuren) Aufbau einer Netzwerkverbindung zu vermeiden, wurden mit der Version 1.1 von Java die sog. JAR Dateien (JavaARchive) eingeführt. Sie können mehrere oder alle Class-Dateien eines Applets enthalten und werden anstelle der einzelnen Dateien geladen, wodurch sich die Applet-Ladezeiten deutlich reduzieren [Loeser 1998].

Im Gegensatz zu JavaScript ist Java eine vollständige Programmiersprache mit der auch eigenständige Anwendungen realisiert werden können. Der Aufwand zum Erstellen von Applets ist allerdings in der Regel auch wesentlich höher als der für Scripte. Dafür ermöglichen sie Web-Anwendungen mit komplexen graphischen Benutzeroberflächen.
Durch die Möglichkeit, Anwendungen direkt im Browser ausführen zu lassen, stehen nun neben einem Programm mit Zustandsspeicher auch komplexe Datenauswertungsfunktionen zur Verfügung. Verbunden mit der Grafikfähigkeit von Java können neben den von HTML

nur unterstützten alphanumerischen „Geschäftsdaten“ in Tabellenform auch komplexe Daten visualisiert werden.
Während bei HTTP-basierten Verfahren durch die Übertragung des HTML-Dokuments neben den Daten auch die gesamte Markup-Information geliefert wird, kann bei Java-basierten Lösungen eine Beschränkung auf die eigentlichen Daten erfolgen. Mußte bei HTTP-basierten Verfahren z.B. zur Darstellung eines Kurvenverlaufs serverseitig die Grafik generiert und als eingebettetes Bild übertragen werden, kann die Datenaufbereitung im Browser durch das Applet erfolgen [Loeser 1998].
Darüberhinaus kann der Datenbankzugriff über eine eigene Verbindung realisiert werden. Der Web Server ist nur noch für das Zustandekommen der initialen, allgemeinen Netzwerkverbindung via HTTP und das Übertragen des Applets zuständig. Nach dem Laden des Codes kommuniziert der Client mit dem Anwendungsserver direkt. Dieser wiederum hält die Verbindung zum DB-Server [Benn 1998].
Auf diese Art kann die Zustandslosigkeit des Protokolls HTTP überwunden werden. Mehrschritt-Interaktionen, die über das HTTP-Protokoll nur über Umwege realisierbar waren, sind direkt realisierbar. Es können beliebig lange Transaktionen unter Nutzung von zweiphasigen Commit-Protokollen durchgeführt werden. Hierzu werden Techniken wie JDBC (Java DataBase Connectivity), SQLJ, RMI (Remote Method Invocation) oder Corba eingesetzt. Einige Browser enthalten zu diesem Zweck schon einen in Java implementierten ORB (Object Request Broker).
Im Gegensatz zu CGI-Anwendungen, bei denen alle Parameter als Strings übergeben werden, können Parameter beliebigen Typs übergeben werden. Weiterhin kann ein Server jederzeit auf Änderungen der von einem Applet dargestellten Daten reagieren und die Daten übertragen (Server Callbacks).
Java unterstützt eine automatische Freispeicherverwaltung und abstrahiert den Zugriff auf Betriebssystemfunktionen durch APIs für Ein-/Ausgabe, Netzwerkprogrammierung, Nebenläufigkeit und für graphische Oberflächen. Diese Eigenschaften erlauben den plattformübergreifenden Einsatz im Web [Turau 1999].

Die Möglichkeiten der Benutzerinteraktion und der Datenrepräsentation müssen programmiert werden. Nachteilig wirkt sich dabei aus, daß die Benutzerschnittstelle sowie die Präsentation der Daten in Java implementiert werden müssen. Dabei verliert man den Vorteil der einfachen Gestaltung- und Strukturierungsmöglichkeiten in HTML, welche auch durch zahlreiche Tools unterstützt werden [Rahm 1999].

Eine clientseitige Unterstützung, z.B. eine JVM (Java Virtual Machine) ist nun notwendig. Eine JVM entspricht dabei einem Interpreter, der die Befehle des plattformunabhängigen Java Bytecode auf die Maschinenbefehle des jeweiligen Systems abbildet [Loeser 1998].
Die Verlagerung der Anwendungslogik auf die Clientseite wirkt sich insofern nachteilig aus, da 1. Know-how (Code) freigegeben wird und 2. die Netzwerkbelastung durch den Transport großer Applets, sogenannter fat Clients, erhöht wird [Rahm 1999].

Insgesamt stellt aber Java im Vergleich zu HTTP-basierten Lösungen ein viel breiter gefächertes Lösungspotential zur Verfügung. Dies reicht von den in einer Programmiersprache inhärenten Realisierungsmöglichkeiten über die bereits in der Sprache enthaltenen Funktionsbibliotheken bis hin zu den grafischen Möglichkeiten, die auch die aufbereitete Ausgabe komplexer Daten oder die Schaffung speziell angepaßter Benutzerschnittstellen erlaubt. Daher eignen sich Applet-basierte Ansätze besonders für Anwendungen mit besonderen Anforderungen.

3.6.2 Maschinen Code

ActiveX ist die Bezeichnung für eine von Microsoft entwickelte Sammlung von Techniken, Protokollen und APIs zur Realisierung von netzwerkweiten Anwendungen. Die ActiveX Technik erlaubt z.B. die Einbindung von ausführbaren Programmen in Web-Seiten ähnlich Applets. Aber die Bedeutung von ActiveX geht weiter als die von Applets: ActiveX stellt ein Software-Komponentenmodell zur Verfügung. Das Modell stützt sich auf andere Microsoft-Techniken wie COM und OLE. ActiveX-Steuerelemente sind Objekte, die in Web-Seiten und in jede andere Anwendung eingefügt werden können, die ein ActiveX-Steuerelement-Container ist.
Die Funktionalität von solchen in Web-Seiten eingebetteten Elementen ist mit der von Applets vergleichbar, die verwendeten Techniken unterscheiden sich aber wesentlich. Ein großer Unterschied besteht darin, daß ActiveX Anwendungen gegenüber Applets plattformabhängig sind, da der Maschinencode verteilt wird. Dadurch können sie im Gegensatz zu Applets in jeder Sprache realisiert werden. Zur Zeit ist diese Technik nur unter Windows und mit einem Com-fähigen Web-Browser nutzbar. Ein weiterer Unterschied zwischen ActiveX und Applets besteht in dem zugrunde liegenden Sicherheitskonzept.
Die bei clientseitigen Anwendungen verwendeten Sprachen wie Java und JavaScript sind noch jung und werden noch weiterentwickelt. In relativ kurzen Abständen gibt es neue Versionen. Zum Teil werden an Sprachen auch proprietäre Änderungen vorgenommen.

Verschiedene Browser unterstützen unterschiedliche Versionen von JavaScript. Die Integration der Interpreter in die Browser und die Verbreitung der neuen Browser-Software erfolgt nur mit zeitlicher Verzögerung. Die sich daraus ergebenden Inkompatiblitätsprobleme stellen eine Gefahr für die Akzeptanz clientseitiger Anwendungen dar [Turau 1999].

3.6.3 Sicherheit clientseitiger Anwendungen

Clientseitige Anwendungen basieren oft auf Zusatzprogrammen wie z.B. Plug-ins oder Interpretern, welche Fehler enthalten können. Diese Programme verarbeiten Daten, die von Benutzern ohne weiteres über das Netz geladen werden. Die Kombination von unbekannten Programmen und unbekannten Eingaben ist für Benutzer nicht mehr einfach zu kontrollieren. Die von Entwicklern solcher Programme entworfenen Sicherheitsvorkehrungen haben sich in der Vergangenheit als unzureichend erwiesen.

Die diskutierten Anwendungen Java und ActiveX haben ganz unterschiedliche Sicherheitskonzepte. Java schränkt die Aktionen, welche ein Applet ausführen kann, auf ein Minimum ein (Sandkastenprinzip). So sind weder Zugriffe auf das lokale Dateisystem noch Netzwerkverbindungen zu beliebigen Rechnern erlaubt. Zudem enthält ein Java-Interpreter einen sogenannten Bytecode-Verifier, dieser versucht sicherzustellen, daß der Code von einem korrekten Compiler erzeugt wurde. Ein Klassenlader überwacht das Laden der Klassen und stellt unter anderem sicher, daß keine Systemklassen überschrieben werden.

ActiveX Steuerelemente unterliegen keinen solchen Einschränkungen. Stattdessen tragen sie eine digitale Signatur des Autors, welche durch Zertifikate von Zertifizierungsstellen verifizierbar gemacht werden sollen. Diese Konzept soll das anonyme Verteilen und das nachträgliche Ändern von ActiveX Anwendungen verhindern. Dabei wird aber keine Aussage über das Verhalten solcher Programme gemacht.

Ein Nachteil des Sandkastenprinzips ist, daß der Einsatz von Applets stark eingeschränkt wird. In vielen Anwendungen wäre es z.B. sinnvoll, wenn ein Applet Dateien im lokalen Dateisystem erzeugen dürfte. Aus diesem Grund wurde in Version 1.1 und 1.2 des Java Developer Kits (JDK) signierte Applets (signed Applets) eingeführt. Das Konzept erlaubt die Konfiguration der von einem Applet ausführbaren Aktionen durch die Benutzer. Einem Applet von einem als vertrauenswürdig eingestuften Autor können bestimmte Rechte z.B. Schreibrechte in einem lokalen Verzeichnis eingeräumt werden. Ein signiertes Applet ist dabei ein in eine JAR-Datei zusammengefaßtes und mit einer digitalen Signatur versehenes Applet.

Durch die digitale Unterschrift soll sichergestellt werden, daß das Applet bzw. die Class-Dateien seit der Erstellung des Archives nicht verändert wurden und vom Unterschreiber stammen. Eine Verfälschung oder ein Austausch nach der Erzeugung der JAR-Datei, z.B. auf einem Web Server durch einen nicht vertrauenswürdigen Mitarbeiter oder auf dem Übertragungsweg durch einen „Angreifer“, wird so verhindert. Signierte Applets haben im Vergleich zu normalen Applets und JAR-Dateien den weiteren Vorteil, daß sie persistent auf dem Rechner des Anwenders mittels einer expliziten Installation gespeichert werden können.
Hierdurch ist ein Neuladen über das Netz vor der Ausführung des Applets nicht mehr notwendig, und die Ladezeiten und Startkosten werden, insbesondere bei großen Applets, drastisch reduziert.
Wie auch bei ActiveX-Anwendungen sagt eine digitale Signatur natürlich noch nichts über das wirkliche Verhalten eines Applets aus. Man weiß höchstens, wer ein ungewolltes Verhalten verursacht hat.
Die Realität hat gezeigt, daß zumindest die Umsetzung, wenn nicht sogar das ganze Sicherheitskonzept von Java keinen vollständigen Schutz garantiert. Ein Beleg hierfür sind die in regelmäßigen Abständen entdeckten Sicherheitslücken. Erhebliche Zweifel an der Vertrauenswürdigkeit kamen auf, als signierte ActiveX Steuerelemente auftauchten, welche in der Lage waren, die Festplatte eine Benutzers komplett zu löschen [Turau 1999].

3.7 Zusammenfassung der Anforderungen an server- und clientseitige Anwendungen

Die Tabelle stellt die Anforderungen an die einzelnen Web-DB-Anbindungsarchitekturen noch einmal gegenüber.

	Programmier-aufwand	Abhängigkeit vom Browser	Abhängigkeit vom Betriebs-system	Volle Funk-tionalität einer Programmier-sprache	Performance	Sicherheit
CGI	mittel bis hoch	keine	keine	ja, beliebige Pro-grammiersprache	mittel	*
Fast CGI	mittel bis hoch	keine	keine	dito	besser als bei reinen CGI-Anwendungen	*
Server APIs	hoch	keine	mittel	ja, an Prgram-miersprache der Server API gebunden, meistens C/C++	besser als bei reinen CGI-Anwendungen	*
z.B. Servlets	mittel	mittel	keine	ja, nur Java	besser als bei reinen CGI-Anwendungen	*
Erweiterung d. Serverfunktio-nalität:						
- SSI	sehr gering	keine	gering	nein	mittel	*
- Scripte	mittel	keine	mittel	ja	mittel	*
Interpretierter Code:						
- Scripte (JavaScript, VBScript,Tcl)	mittel	mittel	keine	nein	hoch	unsicherer als Java
- Applets	hoch	gering	gering	ja, nur Java	hoch	**
Maschinen Code:						
ActiveX	hoch	hoch	extrem	ja, in beliebiger Sprache realisier-bar	hoch	digitale Signatur

Tabelle 1: Gegenüberstellung der fünf Anforderungen
Der obere Teil der Tabelle zeigt die serverseitigen (HTTP-basierten), der untere Teil die clientseitigen Web-DB-Anbindungsarchitekturen.

* Sicherheit HTTP-basierter Anwendungen:
1. Zugriffssicherheit des Web Servers:
   - HTTP-Authentisierung (Benutzername/Paßwort)
   - Einschränkung des Verbindungsrechts auf bestimmte Internetadressen bzw. Domains

2. Übertragungssicherheit (Abhörsicherheit)
   - SSL

**Sicherheitsmaßnahmen von Java
- **Laufzeit-Überprüfung**: Wenn der Computer ein Java-Programm ausführt, wird dieses Programm Zeile für Zeile untersucht, um sicherzugehen, daß es nur gültige Java-Anweisungen enthält. Diesen Anweisungen ist nämlich das Eingreifen in andere Programme oder in die Grundeinstellungen des Computers verboten.
- **Eingeschränkte Interaktivität**: Java hat eingebaute Beschränkungen zur Einrichtung von Verbindungen zu anderen Computern. Solange der Anwender keine speziellen Rechte vergibt, kann ein Java-Applet nur Verbindungen zu dem Ort im Internet aufnehmen, von dem es gekommen ist.
- **Eingeschränkte Zugriffsrechte**: Java hat ebenfalls eingebaute Beschränkungen für die Möglichkeit des Programms beim Lesen und Schreiben von Daten. Das Programm kann nur auf die Teile des Computers zugreifen, für die es vorher die entsprechende Erlaubnis erhalten hat.

In diesem Kapitel wurden Konzepte und Techniken für die Realisierung webbasierter DB-Anbindungen vorgestellt, ihre jeweiligen Stärken und Schwächen angesprochen sowie anhand eines Forderungskatalogs untersucht.

Bei den vorangestellten Verfahren, die zunächst nach serverseitigen (HTTP-basierten) und clientseitigen unterschieden wurden, wurden serverseitige Lösungen auf Basis von CGI-Programmen, Server APIs und Servererweiterungen als für Standardanwendungen und die Entwicklung kleinerer Applikationen geeignet eingestuft.

Die clientseitigen Verfahren wurden noch in Interpretierten Code (wie z.B. JavaScript und Java) und Maschinen Code (ActiveX) eingeteilt. Dabei ist JavaScript im Vergleich zu Java eine loosely-typed Sprache, die zwar viel einfacher als Java zu erlernen ist, sich aber in der Komplexität auf einer ähnlichen Eben wie BASIC bewegt. Java dagegen ist eine komplexe, objektorientierte Programmiersprache wie C++ oder Eiffel.
ActiveX, vom Programmieraufwand und von der Funktionalität mit Applets vergleichbar, bietet zum einen nicht die Sicherheitsvorkehrungen wie Java, zum anderen sind dessen Applikationen im Gegensatz zu Applets plattformabhängig.

Für Anwendungen mit besonderen Anforderungen hinsichtlich Sicherheit, Kommunikation und grafischer Interaktion sollten deshalb Applet-basierte Lösungen herangezogen werden, die wegen ihrer Java-Nutzung für clientseitige Datenaufbereitung, Benutzerinteraktion und Kommunikation flexiblere Möglichkeiten bieten. Außerdem stehen in Java mit JDBC und SQLJ „genormte“ DB-Schnittstellen zur Verfügung.

3.8 Realisierung der Anwendung mit dem Developer/2000

3.8.1 Einführung

Aus der bisherigen Darstellung und der Tabelle, die die Ergebnisse der Gegenüberstellung zusammenfaßt, wird ersichtlich, daß JavaApplets für die Realisierung einer GTDS-Erweiterung vor allem wegen seiner Plattformunabhängigkeit und Sicherheit, zumindest zum Zeitpunkt der Entscheidungsfindung, die beste Wahl darstellte. Der größte Nachteil einer Entwicklung mit Java war jedoch der zu erwartende beträchtliche Aufwand für die Programmerstellung, der eine flexible Gestaltung der Applikation zunächst schwierig erscheinen ließ. Eine geeignete Lösung stellte daher der Developer/2000 von Oracle dar, der sich für diese Aufgabe vor allem durch zwei Merkmale anbot [Muller 1997]:

- Produktivität bei Client/Server- und Webentwicklung durch RAD (Rapid Application Development) - Techniken, Objektorientierung und eine vereinheitlichte Client-, Applikationsserver- und Datenbankserverarchitektur
- Einmal erstellte Developer-Anwendungen können sowohl im Client/Server-Betrieb auf Character Mode Endgeräten, als auch (durch den Developer Server) im Web auf einem Thin Client mit java-fähigem Web Browser eingesetzt werden – ohne Änderungen am Programmcode. Damit entfällt die Notwendigkeit, Runtime und Applikationssoftware auf jedem Client zu installieren, wodurch die Einsatz- und Wartungskosten gemindert werden.

Der Oracle Developer/2000 ist ein Applikationsgenerator der Anwendungen vor allem durch deklarative Spezifikation der Anwendung statt durch prozedurale Programmierung erzeugt.
Ein wichtiger Aspekt einer solchen Entwicklung ist Rapid Prototyping. So nennt man den Prozeß, schnell einen Anwendungsentwurf zu erstellen, bei dem möglicherweise einige Merkmale fehlen, sonst aber die substantielle vollständige Performance, die die Basisfunktionalität demonstrieren kann, zur Verfügung steht. RAD versucht die Produktivität zu steigern, um somit die Geschwindigkeit der Auslieferung zu verbessern. Es gibt zwei Ansätze, um die Produktivität zu steigern: Reduzierung des Arbeitsaufwands, um eine gewisse Funktionalität zum Laufen zu bekommen, und Verbesserung der Qualität um die Notwendigkeit der Überarbeitung des Systems in einer späten Projektphase zu reduzieren. RAD fokussiert auf drei Mechanismen, um höhere Produktivität und Qualität zu erreichen [Muller 1997]:

- es verwendet Werkzeuge um den Umfang des Codes, den man schreiben muß, zu reduzieren
- es verwendet so viel Code wie möglich wieder
- es verwendet Prototyping, iterative Entwicklung und Risikomanagement, um die Qualität zu verbessern

Die Betonung liegt nicht auf „quick and dirty“, sondern darauf „quick and clean“ Code Prototypen zu produzieren. Es gibt relativ wenige Lebenszyklusmodelle für RAD. Einige interessante Modelle, die sich auf Iteration und Wiederverwendung fokussieren, treten in einem anderem, dem objektorientierten Programmparadigma auf.

Das objektorientierte Paradigma

Obgleich das Schreiben von objektorientierten Programmen technologisch unterschiedlich vom RAD ist, so ist doch die Fokussierung auf Iteration und Wiederverwendung sehr ähnlich. Objektorientierte Lebenszyklen sind z.B. das Springbrunnenmodell (B. Henderson-Sellers und Julian M. Edwards, „The Object-oriented Systems Life Cycle“) und das rekursive-parallele Modell (Edward V. Berard, „Essays on Object-oriented Software Engineering“).
Objektorientierte Lebenszyklen brechen das Projekt in Abschnitte auf und parallelisieren die Produktarchitektur. Man kann somit verschiedene parallele Teile haben, die auf unterschiedliche Arten während eines Projektes iterieren, und die alle auf eine ultimative Integration als ein Arbeitssystem zielen; indem man inkrementellen Progreß hinzufügt, erhält man die Möglichkeit auf verschiedene unterschiedliche Dinge zu schauen, die alle auf einer bestimmten Ebene arbeiten und alle im Zustand voranschreiten.
Statt die Programmierung und Testphase wie z.B. im Wasserfallmodell in das letzte Stadium des ganzen Projektes zu verschieben, haben objektorientierte Modelle die Programmierung und das Austesten in die Prototypen verlegt.
Wie im Spirallebenszyklus wechseln sich Risikoanalyse und Prototypisierung in jeder Iterationsphase nacheinander ab. Dabei untersucht die Risikoanalyse die Prozesse um technische, planerische und Kostenrisiken zu identifizieren.

Die andere Schlüsseltechnologie in OO Lebenszyklen, die für RAD Projekte von Interesse ist, ist die Fokussierung auf Wiederverwendung. Wiederverwendung ist die Verwendung eines Systems in Aufgaben, die unterschiedlich von der Aufgabe sind, für die das System erstellt wurde. So können z.B. Teile eines Formulars, eines Berichts oder Komponenten einer

Objektbibliothek wiederverwendet werden. Wiederverwendung hat die Wirkung, daß man weniger Code schreibt, weniger Fehler produziert und somit die Produktivität erhöht.
Die Wiederverwendbarkeit eines Systems erreicht man durch eine geringe Verbindung zu anderen Teilen des Systems und durch dessen Parametrisierung. Parametrisierung z.B. ermöglicht, daß man das Verhalten eines Systems verändert indem man unterschiedliche Argumente liefert und es dadurch flexibler und somit wiederverwendbarer macht [Muller 1997].

Der Oracle Developer beinhaltet Werkzeuge zur Erstellung von Masken, Berichten, Grafiken, Abfragen, Datenbankobjekten und Prozeduren. Diese Entwicklungstools laufen auf allen gängigen Plattformen (Windows, Macintosh, UNIX).
Moderne Anwendungen bestehen allgemein aus einer graphischen Benutzeroberfläche mit Menüs, Toolbars, Dialogboxen und Fenstern, die die Anwendungsobjekte darstellen. Die zwei Arten von Objekten, die hier von Interesse sind, sind Masken und Berichte.
Masken sind elektronische Formulare, die ein interaktives Werkzeug für die Verwaltung von Daten liefern. Eine Formularanwendung präsentiert Daten in einem Online-Format, das aus einer Serie von Feldern besteht, die über ein oder mehrere Fenster verteilt sind. Sie bieten eine gute Möglichkeit die Information, die die Anwendung enthält, zu überblicken, neue Daten einzugeben und zu ändern.
Ein Bericht dagegen ist eine seitenorientierte Darstellung der Daten. Eine große Menge von Daten soll in einer lesbaren Art und Weise dargestellt werden.

Die Webarchitektur von Formularen und Berichten soll in den nächsten beiden Abschnitten kurz dargestellt werde.

3.8.2 Die Webarchitektur von Formularen

Es gibt zwei Hauptunterschiede zwischen einer Formularanwendung in einer Client/Server Architektur und einer Web Implementation [Oracle Developer/2000 1997].

- Client/Server

  Die Forms Runtime Maschine und die gesamte Anwendungslogik werden auf der Desktop Maschine des Endbenutzers installiert. Obgleich eine Anwendung datenbankseitige Trigger und Logik beinhalten kann, finden typischerweise alle Benutzerschnittstellen- und Triggerprozesse auf der Clientmaschine statt.

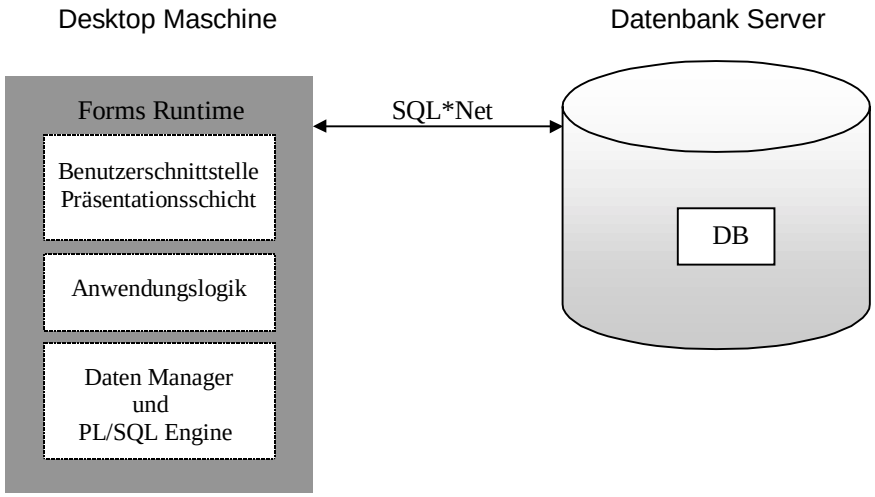

Abbildung 3: Developer/2000: Client/Server Architektur
Die Forms Runtime Maschine und die gesamte Anwendungslogik werden auf der Desktop Maschine des Endbenutzers installiert. Obgleich eine Anwendung datenbankseitige Trigger und Logik beinhalten kann, finden typischerweise alle Benutzerschnittstellen- und Triggerprozesse auf der Clientmaschine statt. Übertragungen zwischen dem Forms Server und dem Datenbank Server erfolgen dabei über das SQL*Net.

- Web: Die Forms Server Runtime Maschine und die gesamte Anwendungslogik sind auf dem Anwendungsserver und nicht auf der Clientmaschine installiert. Alle Triggerprozesse finden auf der Datenbank und auf dem Anwendungsserver statt, während die Benutzerschnittstellenprozesse auf dem Forms Client stattfinden.

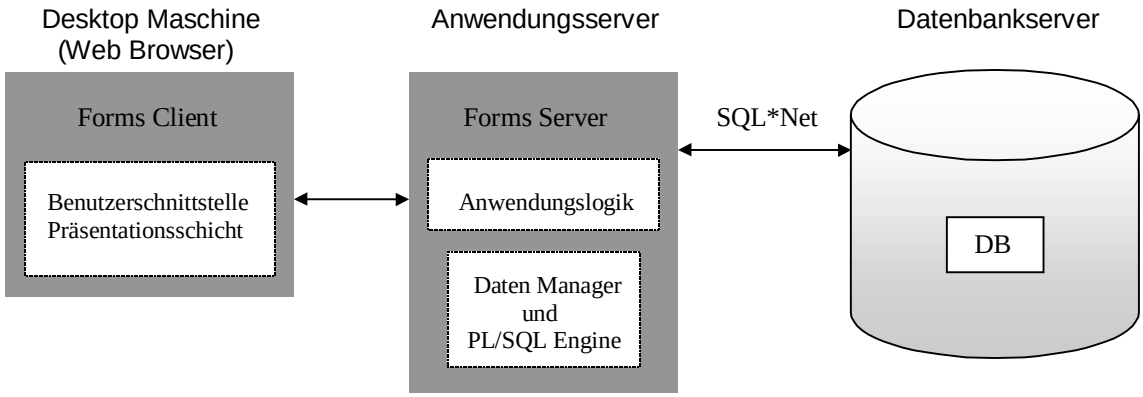

Abbildung 4: Developer/2000: Web Architektur
Die Forms Server Runtime Maschine und die gesamte Anwendungslogik sind auf dem Anwendungsserver und nicht auf der Clientmaschine installiert. Alle Triggerprozesse finden auf der Datenbank und auf dem Anwendungsserver statt, während die Benutzerschnittstellenprozesse auf dem Forms Client stattfinden. Übertragungen zwischen dem Anwendungsserver und dem Datenbank Server erfolgen dabei über das SQL*Net.

Der Developer/2000 unterstützt somit eine 3-Schichten Architektur.

Schicht	Hardware
Front-end	Beliebige Anzahl von Client Desktop Maschinen Desktopmachinen sind mit javafähigen Webbrowsern oder Applet Viewern ausgestattet und werden vom Endanwender verwendet, um die Anwendung auszuführen
Mitte	Ein oder mehrere Anwendungsserver Auf diesen Servern läuft die Webserver Software und eine oder mehrere Developerkomponenten (Forms, Reports, Graphics)
Back-end	Ein oder mehrere Datenbankserver

Der Vorteil eines 3-Schichtenmodells liegt in der klaren Abgrenzung der generischen anwendungsunabahängigen Funktionalität des Anwendungsservers, der anwendungsspezifischen Logik und der Schnittstellenelemente.

Die Forms Webkomponenten des Developer/2000 bestehen aus dem Forms Client und dem Forms Server [Oracle Developer/2000 1997].

Forms Client

Der Forms Client ist ein Java Applet - zur Laufzeit von einem Anwendungsserver zum Web Browser eines Endbenutzers heruntergeladen - das die Schnittstelle des Formsbenutzers darstellt und die Interaktion zwischen Endbenutzern und Forms Server verarbeitet. Der Forms Client erhält „Bündel“ von Schnittstellenbefehlen vom Forms Server und übersetzt diese (in Mengen) in Schnittstellenobjekte für den Endbenutzer.

Der Forms Server

Der Forms Server besteht aus zwei Komponenten:

- Listener: Der Forms Server Listener initiiert die Forms Server Laufzeitsitzung und etabliert eine Verbindung zwischen dem Forms Client und der Forms Server Runtime Maschine.
- Runtime Maschine: Die Forms Server Runtime Maschine ist eine modifizierte Version der Forms Runtime Maschine bei der die Benutzerschnittstellenfunktionalität zum Forms

Client umgelenkt wird. Sie bearbeitet die gesamte Formularfunktionalität mit Ausnahme der Benutzerschnittstelleninteraktionen, einschließlich Trigger- und Ausführungsprozesse, Record Management und allgemeine Datenbankinteraktionen.

Um eine Formularanwendung zu starten, zeigen Anwender mit einem javafähigen Browser oder einem Applet Viewer auf einen URL, der eindeutig mit einer Anwendung verbunden ist. Die Listenerkomponente des Web Servers, die auf Kommunikationsanfragen vom Client wartet, nimmt diese Anfrage entgegen, holt die HTML-Datei vom Betriebssystem und schickt sie zum Client Desktop. Der Web Listener verwendet HTTP um mit dem Client zu kommunizieren und er kann Verbindungen von einer oder mehreren Adressen/Ports annehmen.
Die HTML-Seite enthält alle Informationen, die notwendig sind, das Forms Client Applet herunterzuladen, wie die Portnummer, durch die das Applet mit dem Forms Server Listener kommunizieren wird, und der Name des Webmodules, das das Applet ausführen wird, ebenso wie die zusätzlichen Parameter, die dem Modul übergeben werden.
Nachdem die oben beschriebene HTML-Seite heruntergeladen wurde, wird das Forms Client Applet gemäß den Instruktionen in dieser HTML-Datei ebenfalls heruntergeladen. Das Applet schickt eine Kommunikationsanfrage zusammen mit dem Modulenamen und allen zusätzlichen Parameter, die in der HTML Datei spezifiziert wurden, an einen Port, der ebenfalls in der Datei spezifiziert wurde, und an dem der Forms Server Listener auf Anfragen wartet.
Der Forms Listener kontaktiert die Forms Server Runtime Maschine und verbindet sich mit einem Forms Server Runtime Prozeß (entweder indem er einen neuen Prozeß startet oder indem er sich mit einem existierenden Prozeß verbindet). Die Parameter, die vom Applet an den Listener weitergereicht wurden, werden nun an den Forms Server Runtime Prozeß übergeben.
Der Forms Listener etabliert eine direkte Socketverbindung mit der Forms Runtime Maschine und schickt die Socketinformation zum Forms Client. Der Forms Client etabliert dann eine direkte Socketverbindung mit der Forms Runtime Maschine. Der Forms Client und die Forms Runtime Maschine kommunizieren dann direkt und befreien den Listener um Startanfragen von anderen Endbenutzern zu akzeptieren. Der Forms Client stellt die Benutzerschnittstelle der Anwendung in einem Appletfenster außerhalb des Hauptfensters des Webbrowsers dar. Erfordert das Modul Datenbankinteraktionen, erstellt die Forms Server Runtime Maschine

eine SQL*Net Verbindung mit dem Datenbankserver und kommuniziert mit ihm wie in einer Client/Server Implementierung [Lulushi 2000]

Das Ausführen von Formularanwendungen im Web

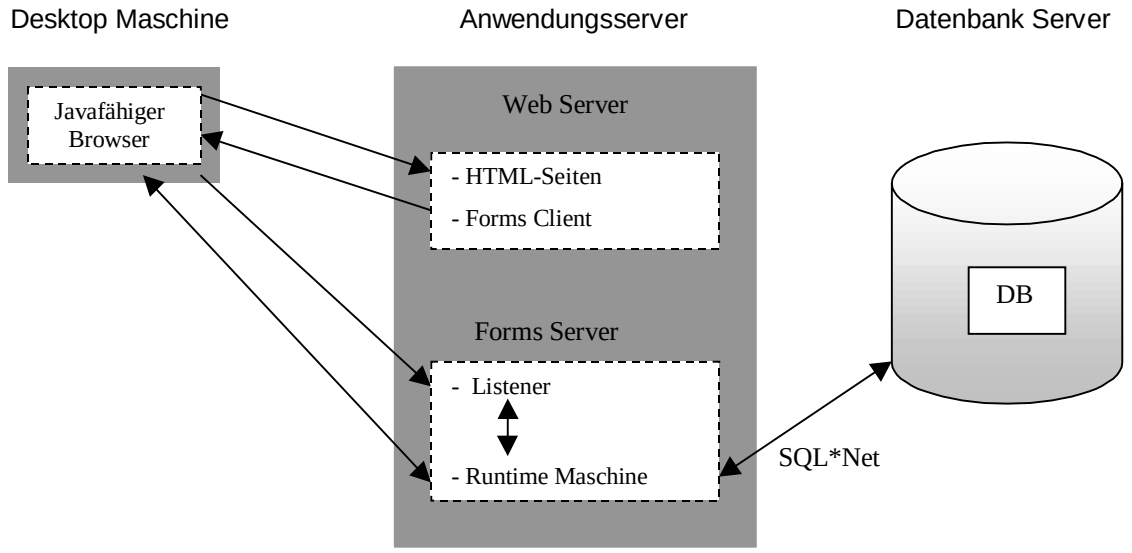

Abbildung 5: Ausführen von Formularanwendungen auf dem Web: Prozeßfluß
Um eine Formularanwendung zu starten, zeigen Anwender mit einem javafähigen Browser auf einen URL, der eindeutig mit einer Anwendung verbunden ist. Die Listenerkomponente des Web Servers nimmt diese Anfrage entgegen. Eine HTML-Datei und dann das Forms Client Applet werden vom Anwendungsserver zum Browser des Anwenders heruntergeladen. Das Applet schickt eine Kommunikationsanfrage an einen Port, an dem der Forms Server Listener auf Anfragen wartet. Der Forms Listener kontaktiert die Forms Server Runtime Maschine und verbindet sich mit einem Forms Server Runtime Prozeß. Die Parameter, die vom Applet an den Listener weitergereicht wurden, werden nun an den Forms Server Runtime Prozeß übergeben. Der Forms Listener etabliert eine direkte Socketverbindung mit der Forms Runtime Maschine und schickt die Socketinformation zum Forms Client. Der Forms Client etabliert dann eine direkte Socketverbindung mit der Forms Runtime Maschine. Der Forms Client und die Forms Runtime Maschine kommunizieren dann direkt und befreien den Listener um Startanfragen von anderen Endbenutzern zu akzeptieren. Der Forms Client stellt die Benutzerschnittstelle der Anwendung in einem Appletfenster außerhalb des Hauptfensters des Webbrowsers dar. Erfordert das Modul Datenbankinteraktionen, erstellt die Forms Server Runtime Maschine eine SQL*Net Verbindung mit dem Datenbankserver und kommuniziert mit ihm wie in einer Client/Server Implementierung.

Ist einmal eine direkt Netzwerkverbindung zwischen dem Forms Client und dem Forms Server etabliert, kommunizieren diese beiden Komponenten über eine Serie von Anfragen und Antworten - via komprimierten Nachrichten, die über ein Netzwerk weitergereicht werden [Oracle Developer/2000 1997].

Anfragen vom Forms Client sind Ereignisse:

- High-level Operationen (wie z.B. Annehmen oder Streichen eines Dialoges)
- Operationen (wie z.B. Ankreuzen einer Checkbox oder Navigieren zwischen Feldern) die Validierungsprozesse zur Folge haben und das Auslösen voreingestellter und benutzer-definierter Trigger verursachen

Antworten vom Forms Server sind eine Serie von Veränderungen an die Benutzerschnittstelle (wie z.B. Wertveränderungen und Hinzufügen und Entfernen von Komponenten), alle die der Forms Client in Darstellungsobjekte umwandelt.

Sicherheit und Verschlüsselung
Daten, die zwischen Datenbank, dem Forms Server und dem Forms Client ausgetauscht werden, werden automatisch vor und nach der Übertragung durch die folgenden Protokolle verschlüsselt.

- RSA RC4 40-Bit Verschlüsselung (für Übertragungen zwischen dem Forms Client und dem Forms Server)
- SQL*Net SNS/ANO (für Übertragungen zwischen dem Forms Server und dem Datenbank Server.

Dabei wird Verschlüsselung durch die Voreinstellung geliefert, kann aber jederzeit aufgehoben werden.

3.8.3 Die Webarchitektur von Berichten

Eine Methode der Webpublizierung ist, statische HTML oder PDF Ausgaben zu generieren, sie in einer Datei auf dem Server zu plazieren und sie mit einer Webseite zu verbinden. Diese Methode ist sehr sinnvoll, wenn man Berichte in regelmäßigen Abständen ausführt und die vorhergehenden Versionen der Ausgabe für einen gewissen Zeitraum beibehält.
Für andere Daten, wie z.B. Arztbriefe oder Gesamtberichte, ist es aber wichtig, daß man die aktuelle Information zur Verfügung hat. Die dynamische Webberichterstellung generiert die Berichtausgabe genau dann, wenn man auf den URL klickt. Ein Bericht wird ausgeführt und die Ausgabe an den Web Browser zurückgeliefert. Durch diese Methode ist sichergestellt, daß man immer die aktuellsten Daten betrachtet.

Für die dynamische Webberichterstellung verwendet man den Reports Server in Verbindung mit der Reports Web Cartridge oder dem Web CGI. Der Reports Server ermöglicht einem, Berichte auf einem entfernten Anwendungsserver laufen zu lassen. In Verbindung mit einer Reports Web Cartridge oder einem Web CGI, gibt er Anwendern auch die Möglichkeit Berichte von einem Web Browser aus unter Verwendung der Standard URL Syntax zu generieren und die Ausgabe auf dem Browser des Client im HTML- oder Adobe´s PDF-Format zu sehen.

Die Reports Web CGI liefert dabei eine Standardverbindung zwischen einem Web Server und einem Reports Server. Die Reports Web Cartridge ist eine Library und führt dieselben Funktionen aus wie die Reports Web CGI, hat aber den Vorteil einer nativen Integration mit dem Oracle Web Application Server. Einmal installiert ist sie immer bereit. Die CGI Executable muß dagegen jedesmal gestartet werden [Oracle Developer/2000 1997].

Um einen Bericht auf dem Web zu starten, zeigen die Endbenutzer, wie oben schon erwähnt, aus ihrem Browser heraus auf einen URL. Die folgende Sequenz tritt dann auf [Oracle Developer/2000 1997]:

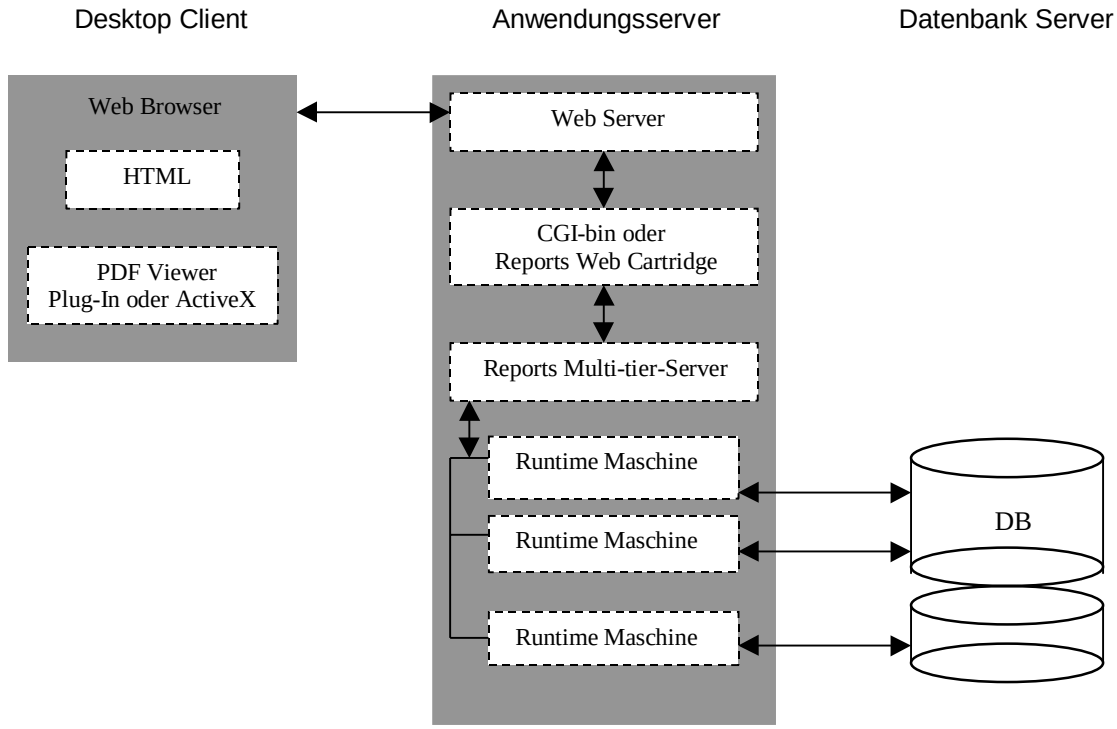

Abbildung 6: Developer/2000: Die Webarchitektur von Berichten
Für die dynamische Web Berichterstellung kann man den Reports Server in Verbindung mit der Reports Web Cartridge oder der Web CGI verwenden. Klickt man auf einen Link, wird die URL von der Reports Web Cartridge oder der Web CGI ausgeführt und die Jobanfrage an den Reports Server geschickt. Der wiederum gibt den Job zwecks Ausführung an eine Runtime Maschine weiter. Wenn die Berichtausgabe fertig ist, kann diese vom Web Server zurück zum Web Browser gesendet werden.

1. Der Web Browser gibt den URL an den Web Server weiter und der Web Server sendet die Anfrage an die Reports Web Cartridge oder das Web CGI.
2. Die Reports Web Cartridge oder das Web CGI analysiert die Anfrage und wandelt sie in eine Kommandozeile um, die durch den Reports Server ausgeführt werden kann.

3. Der Reports Server fügt die Jobanfrage der Web Cartridge oder dem Web CGI in eine Warteschlange und sobald einer der Runtime Maschinen verfügbar ist, wird der Bericht ausgeführt. Wenn die Jobs in der Warteschlange steigen, kann der Reports Server mehrere Runtime Maschinen starten bis er das maximale Limit erreicht hat, das spezifiziert wurde, wenn der Serverprozeß gestartet wurde. In gleicher Weise werden untätige Maschinen geschlossen, nachdem sie für länger als eine spezifische Zeitperiode untätig waren.
4. Die Reports Web Cartridge oder das Web CGI erhält den Namen der Berichtsausgabe vom Reports Server und erstellt eine HTTP Nachsendung zur Ausgabedatei.
5. Der Web Server führt die HTTP-Nachsendung aus, gibt den neuen URL an den Web Browser zurück und der Bericht wird auf dem Desktop Client dargestellt.

3.9 Praktische Evaluierung der Anwendung

Praktische Erfahrungen konnten zunächst mit der ersten webfähigen Probeversion (Trialversion V 1.4) des jetzige Developer/2000 (V 1.6) gemacht werden. Dieser war zusammen mit einem Oracle Web Server 3.0 auf einem NT-Rechner im Institut, der von nun an als Anwendungsserver diente, installiert worden. Der Webclient wurde von einem zu diesem Zeitpunkt noch im Klinikum weit verbreiteten Apple Macintoshsystem gestartet.
In dieser Umgebung zeigten sich jedoch einige aus der Oracledokumentation nicht zu ersehende Probleme:

- der Serverprozeß lief in dieser Version nicht stabil
- als schwerwiegendstes Hindernis für eine praktische Einsetzbarkeit erwiesen sich die langen Wartezeiten für das Herunterladen des Applets vom Anwendungsserver über das Kliniknetz zum Webclient, die ohne weiteres über 10 Minuten betrugen. Auch die Navigation von einer Maske zur anderen lief nur mit erheblicher zeitlicher Verzögerung, so daß nur ein sehr geduldiger und nicht unter Zeitdruck stehender Anwender damit gearbeitet hätte.
- zusätzlich viel negativ auf, daß sich das Bildschirmlayout auf einem Macintosh anders darstellte als in einer Windowsumgebung, so gab es einige farbliche Differenzen sowie erhebliche Unterschiede in Schriftbild- und größe. Die Lesbarkeit und Übersichtlichkeit der Masken ging dabei etwas verloren.

Eine Nutzung des System unter diesen Umständen wäre zu aufwendig gewesen, da man Masken mit unterschiedlichem Layout für beide Betriebssystemumgebungen hätte entwickeln und warten müssen.

Die Umgebung und insbesondere der Developer/2000 waren zu diesem Zeitpunkt noch nicht reif genug. Da bald darauf auch NT-Rechner im Klinikum eingeführt wurden, wurde auf die klassische Client/Serverinstallation zurückgegriffen.

Allerdings sind inzwischen mit der neuen Developer/2000 Version von Oracle (zur Zeit liegt die Version 2.0 vor) die Probleme der langen Ladezeit und des verzögerten Navigierens durch die Masken gelöst. Während das Wechseln von einer Maske zur anderen ohne zeitliche Verzögerung abläuft, so als ob das GTDS lokal installiert sei, ist beim Starten des Systems über das Web mit eine kurzen Wartezeit von etwa 1 bis 2 Minuten zu rechnen. Steht ein geeigneter Rechner zur Verfügung, der ständig als Server in Betrieb sein kann, könnte das Programm auch als Web-Anwendung im Kliniknetz mit dem Vorteil laufen, daß keine lokale Laufzeitumgebung installiert werden muß.

4. Spezielle Anpassungen des GTDS an die Urologische Klinik

4.1 Problematik

Bisher waren Computer und damit auch das GTDS für dokumentarische Zwecke fast nur in den klinischen Registern verfügbar. Die Eingabe der Daten konnte deshalb nur retrospektiv nach Abschluß der Behandlung erfolgen. Eine direkte Nutzung der erfaßten Daten für klinische Zwecke war nicht oder nur sehr eingeschränkt möglich.

Das GTDS hat als Basisdokumentationssystem eher generischen Charakter und ist daher nicht auf individuelle, benutzergruppenspezifische Bedürfnisse ausgerichtet. Jede Abteilung in einer Klinik hat aber wegen unterschiedlicher Ausrichtung und Schwerpunktbildung und den daraus resultierenden unterschiedlichen Funktionen zusätzlich zu erhebende und zu dokumentierende Daten (primär krankheitsorientiert bzw. behandlungsorientiert).
Spezielle urologische Untersuchungen konnten bisher nur in Papierform erfaßt werden, da das GTDS für die Beobachtung des Krankheitsgeschehens in einer nicht rein onkologischen Abteilung nicht vorgesehen war.

Dem Einsatz des GTDS in einer urologischen Abteilung stand also entgegen, daß

- Daten, die für eine vollständige Erfassung der zu behandelnden Patienten wichtig sind, nicht mit dem System erfaßt werden konnten,
- bestimmte Daten, z.B. Patientenstammdaten, mehrfach erfaßt werden mußten, und
- die Zusammenführung onkologischer und nicht tumorbezogener Daten nur schwer möglich und mit einem erheblichen zusätzlichen Aufwand verbunden war.

Die Erweiterung des GTDS für die urologische Klinik ist keine Anpassung im Sinne der von der ADT (Arbeitsgemeinschaft Deutscher Tumorzentren) herausgegebenen und im Dezember 1995 erschienenen „Organspezifischen Tumordokumentation“. Eine inhaltlich verbreiterte und vertiefte Spezialdokumentation der Organtumoren, die das bewußt beschränkt gehaltene Minimalprogramm der Basisdokumentation erweitert, findet in der Praxis, zum jetzigen Zeitpunkt jedenfalls, keine Anwendung und wurde auch von Seiten der zukünftigen Anwender nicht gewünscht. Die Gründe liegen wohl in der sehr aufwendigen und zeitintensiven Bearbeitung, für die die notwendigen Ressourcen fehlen, sowie in der Betrachtung anderer wissenschaftlicher Fragestellungen.

Das GTDS wurde dagegen so erweitert, daß urologische Untersuchungen praxisnah dokumentiert werden können.

Da bundesweit kein anerkannter Standard für die Dokumentation von urologischen Untersuchungen besteht, konnten nicht einfach schon bereits existierende, bedruckte Dokumentationsformulare als Vorlage für die Entwicklung der elektronischen Formulare verwendet werden. Die Realisierung spezieller abteilungsspezifischer Wünsche und klinischer Anforderungen konnte dagegen nur in enger Zusammenarbeit mit den späteren Anwendern erarbeitet werden. Sie beruht auf jahrelanger klinischer Erfahrung.
Als wichtiger Faktor stellte sich dabei die Struktur der zu erfassenden Daten heraus. Als Hauptstruktur wurden die möglichen Untersuchungsarten gewählt, die dann weiter nach Organen unterteilt wurden. Die detaillierten Anforderungen wurden dann in einer Art Trial and Error Methode verfeinert. In einer iterativen Verfahrensweise wurde die Struktur der Masken, der inhaltliche Aufbau und die notwendige Anzahl der Items festgelegt. Dabei waren mehrere Durchläufe der Maskenerstellung erforderlich.
Ein wichtiges Designkriterium war dabei, die Masken möglichst übersichtlich zu gestalten. Die auf jeder Maske wiederkehrenden Patienten identifizierenden Daten und gleichen Funktionen sollten immer an der selben Stelle plaziert werden. Auch sollten die Items eines Organs nicht auf mehrere Masken verteilt sein. In einzelnen Fällen kann ein Organ über ein Dutzend Items haben, die jeweils wiederum mehrere detailliertere Beschreibungen zur Auswahl bieten und nicht ohne weiteres auf eine Maske passen. Daher mußten andere Möglichkeiten der Darstellung genutzt werden, wie z.B. konditionale Untermenüs und Pop-up Fenster, die nur bei bestimmten Selektionen erscheinen.
Im Laufe der Entwicklung hat sich auch gezeigt, daß es an einigen Stellen gar nicht möglich ist alle Detailschreibungen wie z.B. nähere Charakterisierungen als Auswahlkriterium aufzuführen. Felder in denen freitextliche Anmerkungen gemacht werden können, erschienen hier als die beste Lösung. Auch besteht prinzipiell die Möglichkeit eigene Bemerkungen zu jeder Untersuchungsart zu ergänzen.

Gemäß seinem späteren Einsatzort in der Urologischen Ambulanz der Urologischen Klinik im Universitätsklinikum Gießen wurde das System spezifisch auf deren Bedürfnisse zugeschnitten. Bei der Übertragung auf andere Urologische Kliniken wäre also mit weiteren Adaptationen zu rechnen.

4.2 Anforderungen an die GTDS-Erweiterung

Für die urologischen Untersuchungen wurde ein System benötigt, das die immer wiederkehrenden Untersuchungen systematisch und lückenlos erfaßt und den Verlauf dokumentiert. Dabei sollte eine enge Kopplung durch echte Interoperabilität zwischen den urologischen Untersuchungsmasken und dem grafischen GTDS realisiert werden. Patienten, die sich nach der urologischen Untersuchung als Tumorpatienten herausstellen, müssen auf Knopfdruck ins GTDS übernommen werden können. Umgekehrt muß eine Möglichkeit bestehen, vom GTDS leicht in den urologischen Teil der Dokumentation zu wechseln.
Der zunehmende Ausbau klinikweiter Netze mit PCs an mehr und mehr Arbeitsplätzen erlaubt es nun, die Dokumentation der Daten in den Behandlungsablauf zu integrieren und die Merkmale nicht nachträglich, sondern durch den behandelnden Arzt zu erheben und zu speichern, so daß sie in weit stärkerem Maße als bisher zur Unterstützung der täglichen klinischen Tätigkeit genutzt werden können. Je umfassender dies gelingt, desto besser wird die Qualität der erfaßten Daten.

Der Mehraufwand für den behandelnden Arzt muß sich dabei in engen und für ihn vertretbaren Grenzen halten. So darf die Einarbeitungszeit in das Dokumentationssystem nicht zu aufwendig, müssen die Untersuchungsmasken übersichtlich und selbsterklärend, das System generell benutzerfreundlich sein und eine hohe Performance aufweisen, sowie der Ablauf der Dokumentation intuitiv und effizient durchführbar sein.

4.3 Ablauf der Dokumentation

Der Ablauf der Dokumentation wird in Abbildung 7 schematisch dargestellt. Am Beginn steht die *Übersicht über alle Patienten der Klinik*.
Die Selektion des Patienten erfolgt über die Eingabe der Patienten_Id oder des Namens bzw. eines Teil des Namens. Zeichen können auch durch sogenannte SQL-Wildcards ersetzt werden. „%“ steht für beliebig viele Zeichen, „_“ für genau ein Zeichen. Über eine Suchfunktion (Drücken der Return- oder Tabulator-Taste bzw. Knopf *Suchen*) werden die entsprechenden Patienten mit der zugehörigen Patienten_Id, gegebenenfalls dem Geburtsnamen, dem Geburtsdatum sowie des Geschlechts und des Wohnortes in einer Tabelle dargestellt. Die Genauigkeit der Suche kann dabei variiert werden. Voreingestellt ist die *phonetische Suche*. Sie sucht am ungenauesten und setzt dabei phonetisch verwechselbare Zeichen wie „N“ und „M“ gleich. *Standardisierte Namen* berücksichtigen Groß- und Kleinschreibung sowie Umlaute nicht.

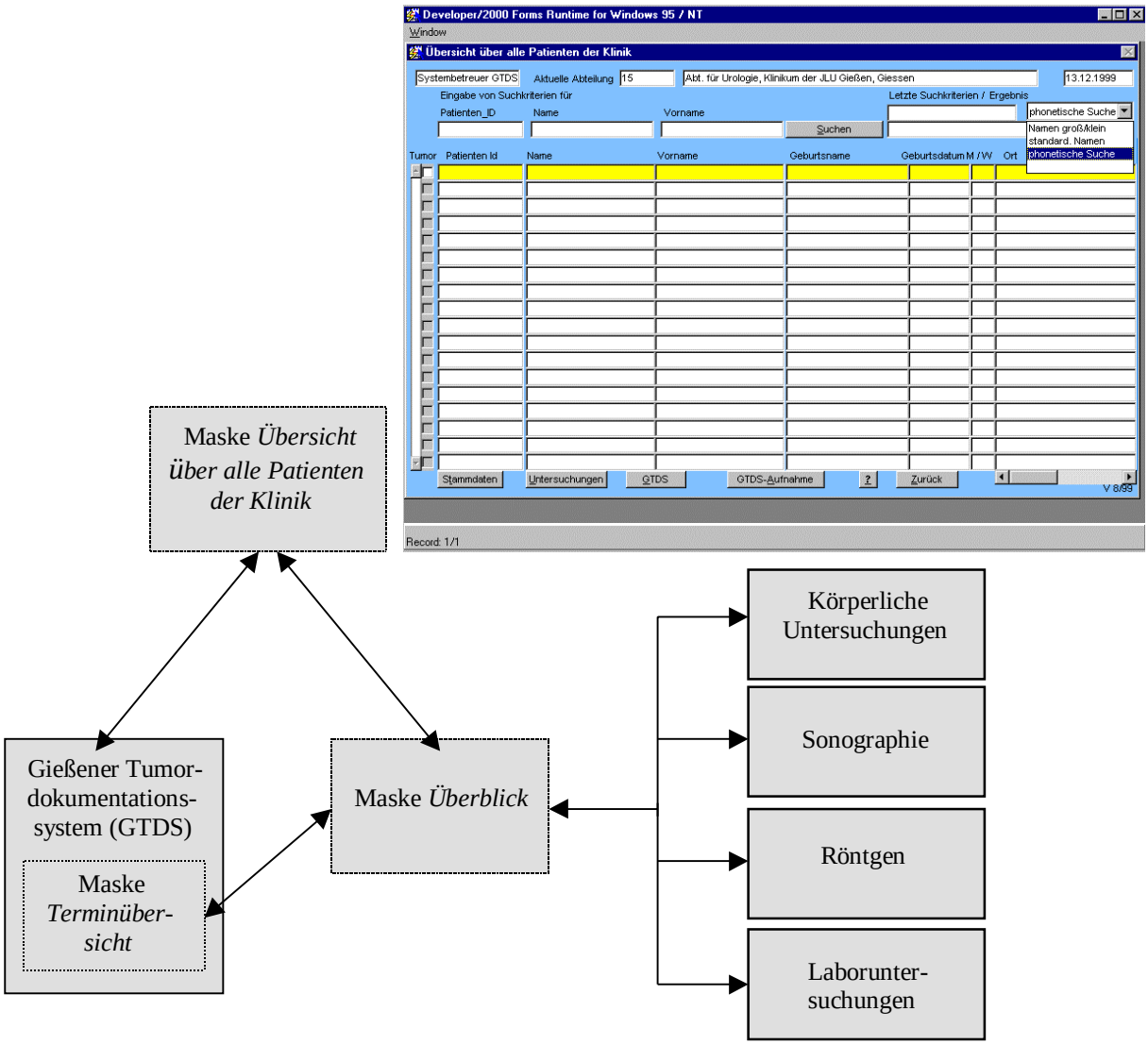

Abbildung 7: Ablauf der Dokumentation

Loggt man sich in das Dokumentationssystem ein, gelangt man zunächst zur Maske *Übersicht über alle Patienten*. Nach der Selektion eines Patienten kann man entweder zum *GTDS* oder zu den (urologischen) *Untersuchungen* (Maske *Überblick*) verzweigen. Dabei besteht ein nahtloser Übergang zwischen der Maske *Überblick*, dem Ausgangspunkt der urologischen Dokumentation, und dem grafischen GTDS. Patienten, die sich nach der urologischen Untersuchung als Tumorpatienten herausstellen, können auf Knopfdruck ins GTDS übernommen werden. Umgekehrt besteht die Möglichkeit vom GTDS (der Maske *Terminübersicht*) leicht in den urologischen Teil der Dokumentation zu wechseln. Aus der Maske *Überblick* gelangt man zu den vier verschiedenen Untersuchungsgruppen *Körperliche Untersuchungen*, *Sonographie*, *Röntgen* und *Laboruntersuchungen*.

Ist der Patient dem System noch nicht bekannt, so können über die Eingabe einer 6- bzw. 7-stelligen Patienten_Id die zugehörigen Stammdaten aus dem PILS (**P**atienten **I**nformations- und **L**eit-**S**ystem), einem ADT Subsystem des WING (siehe Abschnitt 5.5) für die Aufnahme (**a**dmission), Entlassung (**d**ischarge) und Verlegung (**t**ransfer) von Patienten, über die gleichen Mechanismen wie oben beschrieben, geholt werden. Der behandelnde Arzt entnimmt dabei die jeweilige Patienten_Id aus den von der Verwaltung ausgestellten Zetteln bzw. Etiketten, die jeder Patient vor seiner Behandlung erhält und auf dem die persönlichen Daten bereits datentechnisch erfaßt wurden.

Um eine detailliertere Beschreibung des Patienten zu erhalten, kann über die Auswahl von *Stammdaten* in die entsprechende Maske verzweigt werden. Auf ihr werden für den Patienten spezifische Identifikationsdaten (z.B. Name und Vorname(n), Geburtsdatum, Anschrift) sowie organisatorische Daten (wie z.B. SV-Nummer, Hauptversicherungsname und -geburtsdatum) vermerkt.
Die Daten der urologischen Untersuchung werden zunächst bei allen potentiellen Tumorpatienten erhoben. Über *Untersuchungen* gelangt man zur Maske *Überblick* (siehe Abbildung 8) und damit zum Ausgangspunkt für die urologische Dokumentation. Stellt sich heraus, daß eine Tumorerkrankung vorliegt, wird der Patient über Auswahl von *GTDS-Aufnahme* ins GTDS übernommen und mit einer Tumormarkierung vor seiner *Patienten_Id* versehen. Beim wiederholten Erscheinen des Patienten in der Urologischen Ambulanz können aus der Übersichtsmaske heraus jeweils die Daten der Urologischen Dokumentation sowie bei Vorliegen einer Tumorerkrankung die des GTDS (über *GTDS*) ergänzt bzw. erweitert werden. In umgekehrter Richtung können die auch im GTDS vorhandenen Terminfunktionen (Maske *Terminübersicht*) genutzt werden, um im Rahmen der Tumorsprechstunde in die urologischen Daten zu verzweigen.

Die Maske *Überblick* (Abbildung 8) zeigt neben den in jeder Maske wiederkehrenden Kopfdaten (Patientenname und -vorname, Geburtsdatum und Patienten_Id), Verlaufsinformationen zu einem, soweit vorhanden, bereits im GTDS erfaßten Tumor. Dazu gehören die im GTDS zugeordnete *PAT_ID*, das *Untersuchungsdatum*, die über eine Liste auswählbare *Verlaufsnummer* sowie ein *Freitext*.
Die Untersuchungen sind jeweils nach Gruppen geordnet. Durch Anwählen von *Neuer Befund* hinter dem jeweiligen Untersuchungsnamen wird in die entsprechende Untersuchungsmaske verzweigt. Über *Freitext* kann eine freitextliche Anmerkung zu jeder Untersuchungsgruppe ergänzt werden.
Ist ein neuer Datensatz angelegt und abgespeichert worden, erscheint der Name der Untersuchung, einschließlich deren Erfassungsdatum, ID und zugehöriger LfdNr in der Tabelle *Vorhandene Befunde*. Die LfdNr wird dabei automatisch aus dem Feld *Verlaufsnummer* übernommen. Über *F9* kann einer Untersuchung aus einer Liste bereits vorhandener Verlaufsnummern eine andere zugeordnet bzw. eine neue Verlaufsnummer für eine Untersuchung angelegt werden.

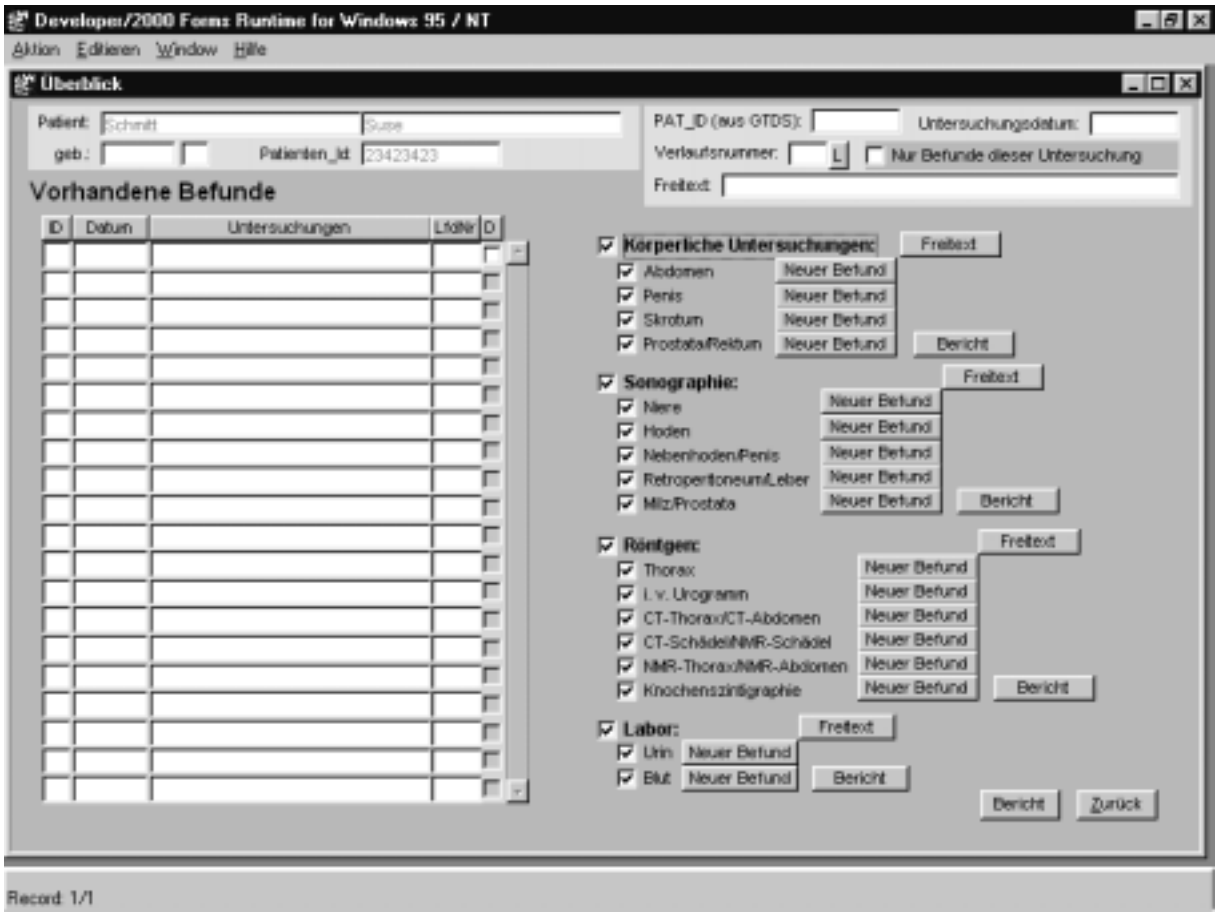

Abbildung 8: Maske *Überblick*

Die *Maske Überblick* ist der Ausgangspunkt der urologischen Dokumentation. Sie zeigt neben den in jeder Maske wiederkehrenden Kopfdaten Verlaufsinformationen zu einem, soweit vorhanden, bereits im GTDS erfaßten Tumor. Die Untersuchungen sind jeweils nach Gruppen geordnet. Durch Anwählen von *Neuer Befund* hinter dem jeweiligen Untersuchungsnamen wird in die entsprechende Untersuchungsmaske verzweigt. Über *Freitext* kann eine freitextliche Anmerkung zu jeder Untersuchungsgruppe ergänzt werden. Ist ein neuer Datensatz angelegt und abgespeichert worden, erscheint der Name der Untersuchung, einschließlich deren Erfassungsdatum, ID und zugehöriger LfdNr in der Tabelle *Vorhandene Befunde*.

Eine markierte Checkbox in der Spalte *D* (für Drucken) der Tabelle *Vorhandene Befunde* bedeutet, daß der jeweilige Untersuchungsbefund in den tabellarischen Gesamtbericht mit aufgenommen wird und über *Bericht* überprüft und ausgedruckt werden kann.

Neben dem Anlegen neuer Befunde können vorhandene Befunde durch Doppelklicken auf eines der Items in der Tabelle angesehen bzw. ergänzt und erweitert werden. Die Untersuchungen lassen sich über die jeweiligen Überschriften nach *ID*, *Datum*, *Untersuchungen*, *LfdNr* und *D* ordnen.

Um die Selektion von bestimmten Untersuchungen in der Tabelle *Vorhandene Befunde* zu erleichtern, können durch Markieren bzw. Demarkieren der entsprechenden Checkboxen einzelne oder eine ganze Gruppe von Untersuchungen angezeigt bzw. ausgefiltert werden.

Die GTDS-Erweiterung umfaßt Datensätze zur Dokumentation von *Körperlichen Untersuchungen*, *Sonographie*, *Röntgen* und *Labor*, die während der Untersuchung erhoben werden.

Struktur:

Körperliche Untersuchungen
• Abdomen • Penis • Skrotum • Prostata/Rektum

Sonographie
• Niere • Hoden • Nebenhoden/Penis • Retroperitoneum/Leber • Milz/Prostata

Röntgen
• Thorax • i.v. Urogramm • CT-Thorax/CT-Abdomen • CT-Schädel/NMR-Schädel • NMR-Thorax/NMR-Abdomen • Knochenszintigraphie

Labor
• Urin • Blut

4.4 Prinzipieller Aufbau einer Maske/Navigation innerhalb einer Maske

4.4.1 Aufbau einer Maske

Zur einfachen Orientierung ist jede Maske strukturell gleich aufgebaut. Sie besteht aus

- einer Menüleiste
- einem gleichlautenden Kopfteil
- dem eigentlichen Haupt- oder Dokumentationsteil
- sowie einer Knopfleiste.

4.4.2 Menüleiste

Ein Menüsystem stellt bestimmte Funktionen zur Verfügung. Es gibt ein Hauptmenü mit vier Menünamen (*Aktion*, *Editieren*, *Window* und *Hilfe*), die eine unterschiedliche Anzahl von Menüitems enthalten. Navigiert man mit der Maus auf einen Menünamen bzw. drückt eine bestimmte Tastenkombination (Alt + Anfangsbuchstabe des Menünamens), so klappt ein Menü herunter (Pull-down-Technik) aus dem ein Menüitem, wieder mittels Maus oder Tastatur, ausgewählt werden kann.

Unter *Aktion* finden sich die Funktionen *Speichern* und *Zurück*. Mit diesen werden Dateninhalte in der Maske gespeichert bzw. zum Ausgangspunkt, der Maske *Überblick*, zurücknavigiert. Der Menüname *Editieren* enthält Menüitems zur Bearbeitung (*Ausschneiden*, *Kopieren*, *Einfügen* und *Editieren*) von Textitems. Mit dem vom Developer/2000 voreingestellten Menünamen *Window* können die Masken diagonal überlagert (cascade) oder nebeneinander dargestellt (tile) werden. Unter *Hilfe* können Eingabehilfen zum jeweils aktuellen Item (Menüitem *Hilfe*) und bei Auftreten eines Fehlers zum aktuellen Fehler (Menüitem *Fehler anzeigen*) eingesehen werden.

4.4.3 Kopfteil

Wie im GTDS hat das Programm jeweils einen Patienten im „Kontext“, der in der Maske *Übersicht über alle Patienten in der Klinik* ausgewählt wurde. Der Kopfteil dient zur eindeutigen Identifikation dieses Patienten. In ihm werden Name und Vorname des Patienten, Geburtsdatum sowie dessen Patienten_Id und Fall _Nummer angezeigt. Ein Feld steht für das Eingabe- und das Änderungsdatum zur Verfügung. Die Benutzer_Id zeigt an, welcher behandelnde Arzt für die Eingaben verantwortlich ist. Diese Felder werden automatisch gefüllt und sind für den Anwender, bis auf das Feld *Datum der Eingabe*, nicht veränderbar und daher in grauer Schrift. Das Eingabedatum kann individuell eingestellt werden, falls die

Dokumentation der Untersuchung aus zeitlichen oder organisatorischen Gründen nicht am selben Tag, sondern um ein oder mehrere Tage verzögert erfolgt.

4.4.4 Hauptteil

Hier findet die eigentliche Dokumentation der Untersuchungen statt.
Grundsätzlich ist vorgesehen, daß bei vielen der einzelnen Felder zugelassene Merkmalsausprägungen in Merkmalskatalogen im System hinterlegt werden, so daß die Codierung über ein Auswahlmenü unmittelbar bei der Eingabe der Daten erfolgen kann.
Freitextliche Ergänzungen sollen im Dokumentationssystem überall dort vorgesehen sein, wo Auswahlmenüs nicht möglich sind und/oder zusätzliche Beschreibungen und Anmerkungen hilfreich sein können.

Die Daten werden je nach Anforderung mit unterschiedlichen Itemtypen erfaßt.

Viele der Merkmale lassen sich mit *Nein* (meistens die Voreinstellung) oder *Ja* beantworten. Bei einer relativ kleinen Zahl sich gegenseitig ausschließender Wahlmöglichkeiten werden Radioknöpfe verwendet. Wenn der Anwender einen Knopf anwählt, wird der gegenwärtig selektierte Knopf deselektiert und der selektierte ausgewählt. Ist dies aus irgendwelchen Gründen nicht eindeutig zu beantworten, sollte dies mit der Markierung des *F.A* (fehlende Angaben)-Radioknopfes dokumentiert werden.
Ist eine detailliertere Beschreibung - z.B. eine nähere Ortsangabe, eine genaue Ausprägungsform, eine Grad- oder Richtungsangabe - erforderlich, so wird eine Liste mit Wahlmöglichkeiten oder eine Auswahl an Checkboxen zur Verfügung gestellt. Ein Liste ist dann sinnvoll, wenn nur ein Element in Frage kommt, während Checkboxen einem die Möglichkeit geben mehrere Items auszuwählen.

Können keine inhaltlichen Vorgaben gemacht werden, werden Textitems zur Verfügung gestellt z.B. bei Größen-, (Breite-, Länge-, Durchmesser-,) Anzahl-, Volumenangaben oder Laborwerten.
Um den Aufwand für den Anwender möglichst gering zu halten, sind Einheiten grundsätzlich schon vorgegeben; sie werden zusammen mit dem dazugehörigen Wert abgespeichert. Da es sich bei Einheiten um unveränderliche Werte handelt, werden diese in Displayitems dargestellt.

Noch zu erwähnen sind die Felder für die ID und die Verlaufsnummer der jeweiligen Untersuchung, die sich im Hauptteil rechts oben befinden. Ist die Maske noch nicht abgespeichert worden, wird im ID-Feld zunächst *NEU* angezeigt. Bei jedem Abspeichern eines neuen Befundes wird die ID automatisch hochgezählt. Im Feld Verlaufsnummer wird die in der Überblicksmaske zugeordnete und damit im Kontext befindliche Verlaufsnummer dargestellt.
Da die ID und die Verlaufsnummer wie oben erwähnt neben Datum und Untersuchungsname in der Maske *Überblick* in der Tabelle *Vorhandene Befunde* angezeigt wird, hat der Anwender immer eine Übersicht darüber, wie oft und wann er die gleiche Untersuchung in welchem Verlaufskontext durchgeführt hat.

4.4.5 Knopfleiste

Sie besteht in den meisten Fällen aus einem *Speichern*-, *Drucken*- und einem *Zurück*-Knopf. *Speichern*- und *Zurück*-Knopf haben dieselbe Funktion wie die gleichlautenden Menüitems unter dem Menünamen *Aktion* und dienen der Navigation zwischen den Masken. Die Markierung von *Drucken* bewirkt, daß der Inhalt der Maske in den tabellarischen Gesamtbericht mit aufgenommen wird. Die Knopfleiste kann wie die Menüleiste mittels Maus oder Tastatur (Alt + unterstrichener Buchstabe) bedient werden.

Innerhalb der Maske kann ebenfalls sowohl mit der Maus als auch über die Tastatur navigiert werden. Dabei sind bestimmte nähere Angaben (z.B. Ort) nur anwählbar, wenn vorher durch Auswählen des richtigen Radioknopfes, die Detailangabe zugänglich geworden ist.

4.5 Beschreibung der einzelnen Masken

Im folgenden soll der Inhalt der Masken kurz dargestellt werden.

Am Beginn steht die Erfassung der *Körperlichen Untersuchungen.* Sie beinhaltet Abdomen, Penis, Skrotum sowie Prostata/Rektum.

Maske Körperlich Abdomen

Die Maske *Körperlich Abdomen* enthält Eingabefelder zur Dokumentation der körperlichen Untersuchung des Abdomens.

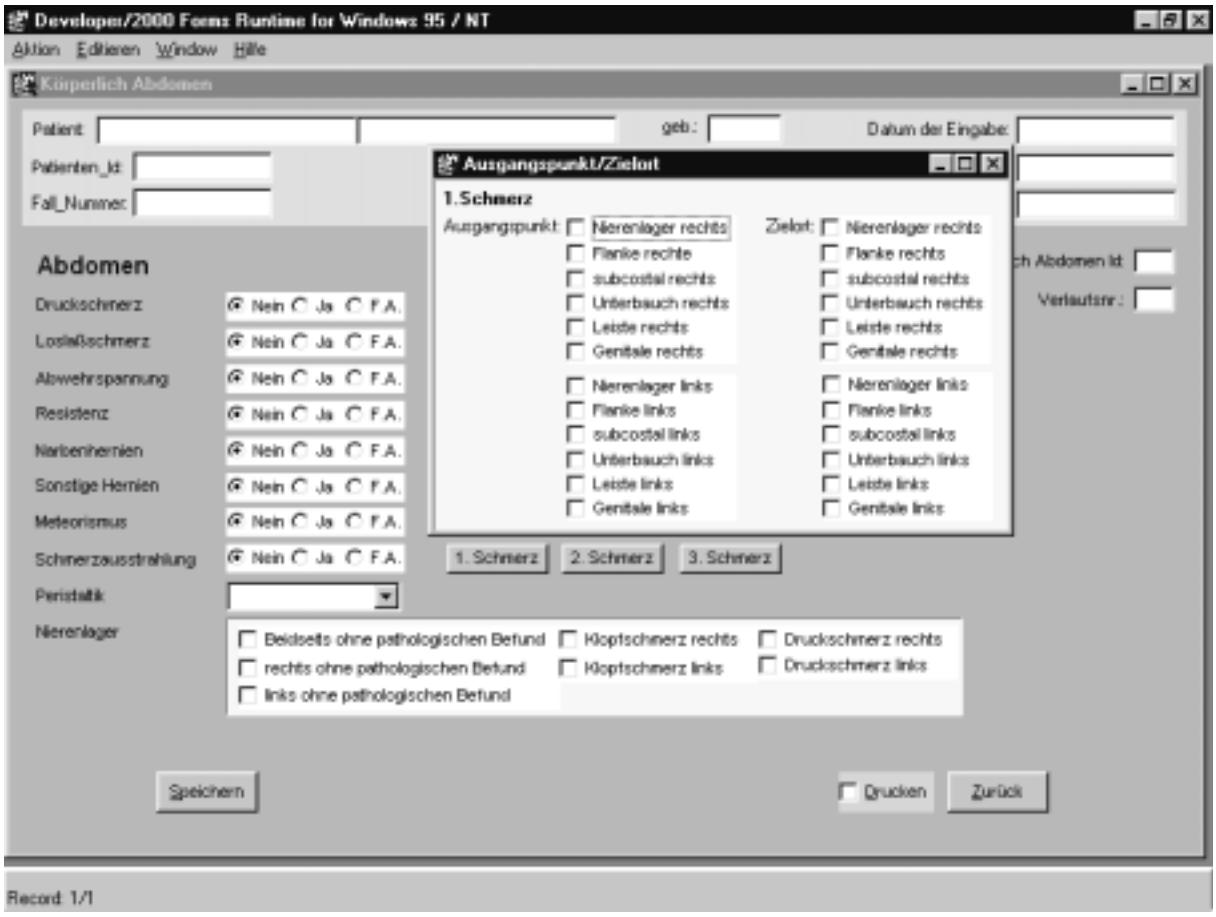

Abbildung 9: Maske *Körperlich Abdomen*

Zu erfassen sind insgesamt 10 Merkmale: *Druckschmerz*, *Loslaßschmerz*, *Abwehrspannung*, *Resistenz*, *Narbenhernien*, *Sonstige Hernien*, *Meteorismus*, *Schmerzausstrahlung*, *Peristaltik* und *Nierenlager*. Muß bei den Merkmalen *Druckschmerz*, *Loslaßschmerz*, *Abwehrspannung*, *Resistenz* und *Narbenhernien* mit *Ja* geantwortet werden, so öffnet sich jeweils ein Fenster und es kann der genauere Ort (*Oberbauch rechts, mitte, links, Nabelregion, Unterbauch*

rechts, *mitte*, *links* bei *Druckschmerz*, *Loslaßschmerz*, *Abwehrspannung* bzw. noch zusätzlich *Nierenlager rechts, links* bei *Resistenz* und *Narbenhernien*) durch Anklicken einer oder mehrerer Checkboxen bestimmt werden. Beim Merkmal *Sonstige Hernien* besteht die Möglichkeit, zwischen 3 verschiedenen Hernienarten (*Nabelhernie*, *Leistenhernie rechts*, *links*) auszuwählen bzw. eine freitextlich zu ergänzen. *Schmerzausstrahlung* differenziert zwischen drei Schmerzen, deren Ausgangspunkt und Zielort (jeweils 12 Auswahlmöglichkeiten) genau spezifiziert werden können.

Beim Merkmal *Peristaltik* kann aus einer Liste von 5 Elementen (*normal*, *rege*, *klingend*, *abgeschwächt*, *stumm* bzw. *keine Angaben*) ausgewählt werden. *Nierenlager* läßt die Auswahl eines oder mehrerer Orte zu.

Maske Körperlich Penis

Diese Maske erfaßt die körperliche Untersuchung des Penis.

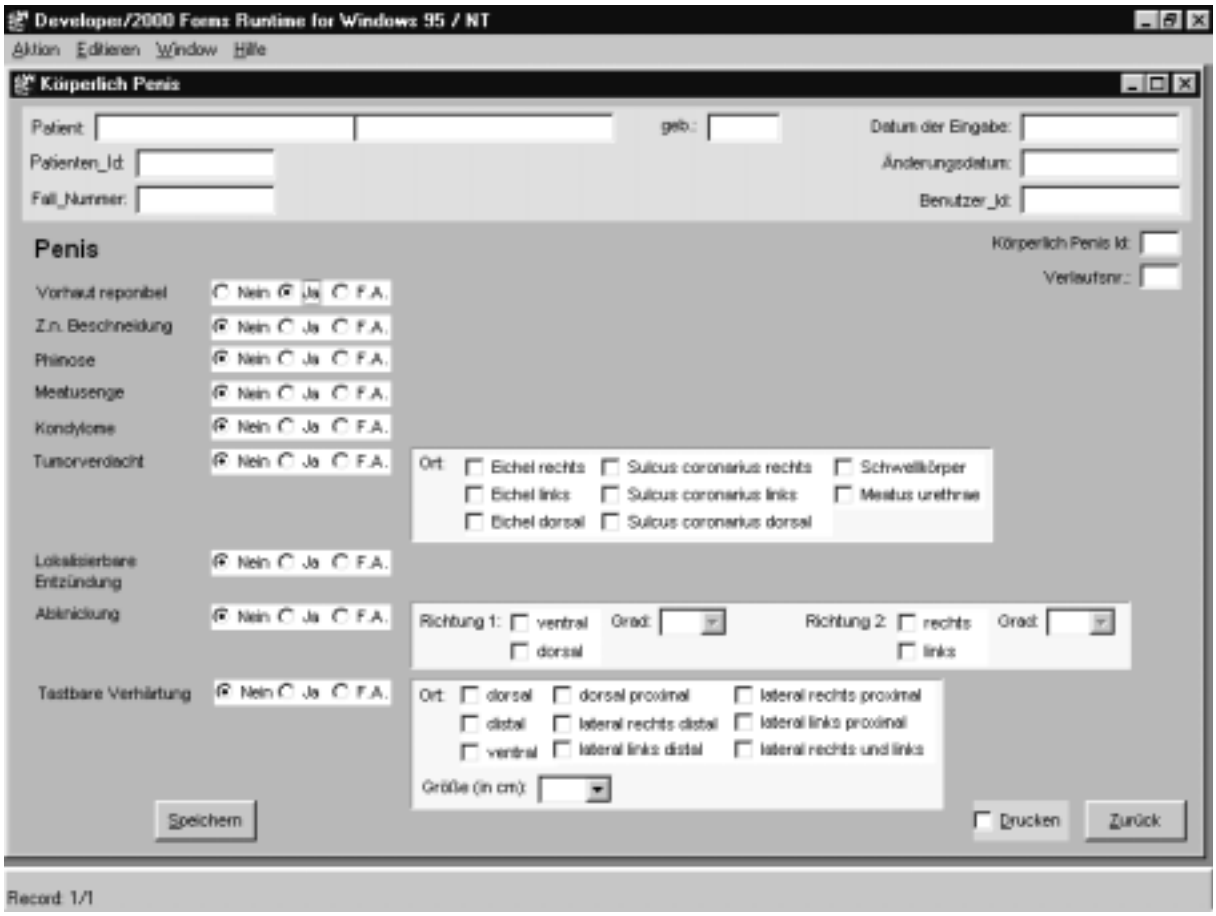

Abbildung 10: Maske *Körperlich Penis*

Hier sind 9 Merkmale zu dokumentieren: *Vorhaut reponibel*, *Z.n. Beschneidung*, *Phimose*, *Meatusenge*, *Kondylome*, *Tumorverdacht*, *Lokalisierbare Entzündung*, *Abknickung* und *Tastbare Verhärtung*. *Tumorverdacht* verlangt beim Anklicken von *Ja* nach einer detaillierteren Lokalisationsangabe. Das Merkmal *Abknickung* kann noch nach *Richtung 1* (*ventral*, *dorsal*) bzw. *Richtung 2* (*rechts*, *links*) und dem dazugehörigen Grad (*10 - 130°*) spezifiziert werden. Orts- (insgesamt 9 Auswahlmöglichkeiten) und Größenangabe (*1,0 - 5,0*) sind bei Bejahung des Merkmals *Tastbare Verhärtung* zu erfassen.

Maske Körperlich Skrotum

Die Maske *Körperlich Skrotum* ist für die Dokumentation der körperlichen Untersuchung des Skrotums vorgesehen.

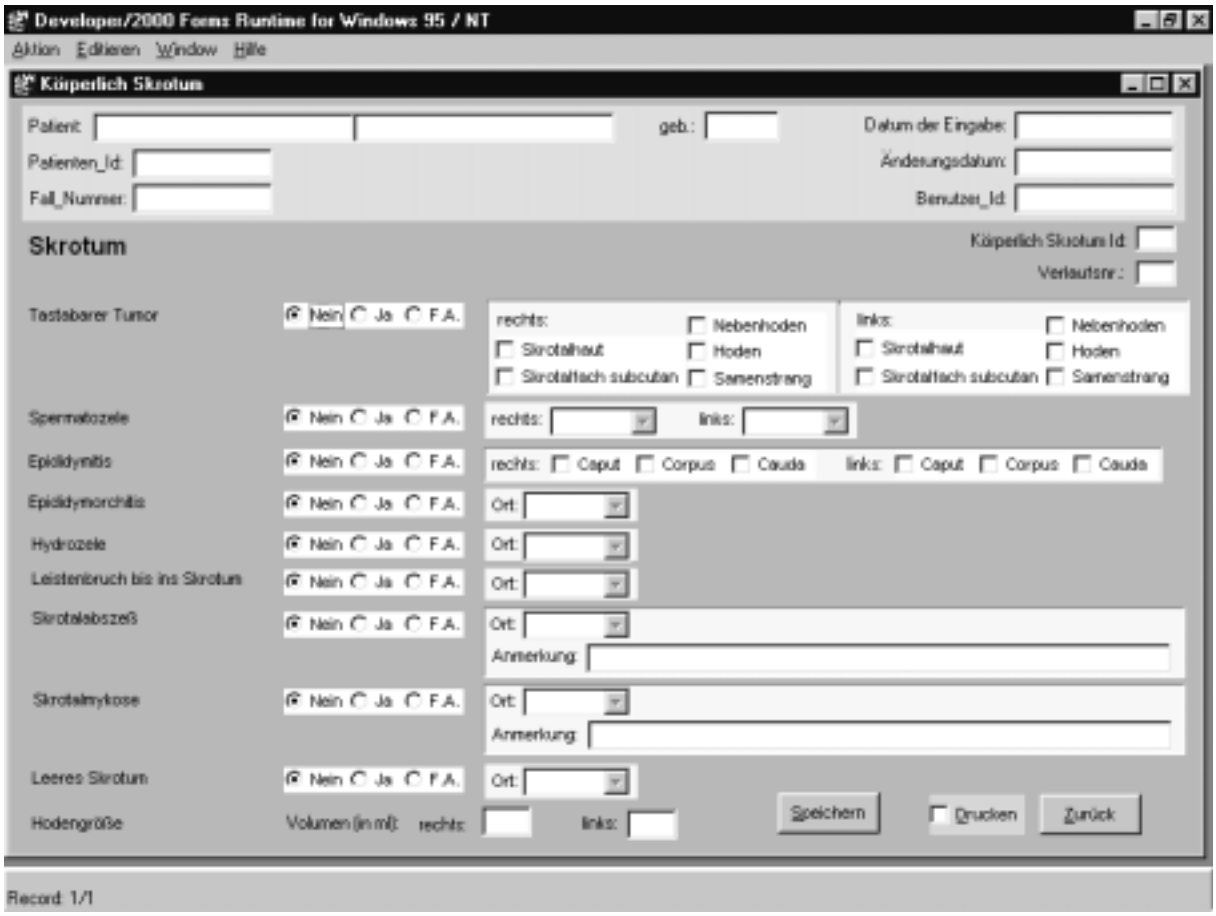

Abbildung 11: Maske *Körperlich Skrotum*

Die 10 zu erfassenden Merkmale sind: *Tastbarer Tumor*, *Spermatozele*, *Epididymitis*, *Epididymorchitis*, *Hydrozele*, *Leistenbruch bis ins Skrotum*, *Skrotalabszeß*, *Skrotalmykose*, *Leeres Skrotum* und *Hodengröße*. Bis auf das letzte Merkmal ist bei Anklicken von *Ja* eine

genauere Lokalisationsangabe erforderlich. Bei den beiden Merkmalen *Skrotalabszeß* und *Skrotalmykose* kann die jeweilige Ortsangabe noch durch eine Anmerkung ergänzt werden.

Maske Körperlich Prostata/Rektum

Diese Maske dient zur Dokumentation der körperlichen Untersuchungen der Prostata und des Rektums. Es sind jeweils 8 bzw. 6 Merkmale zu erfassen.

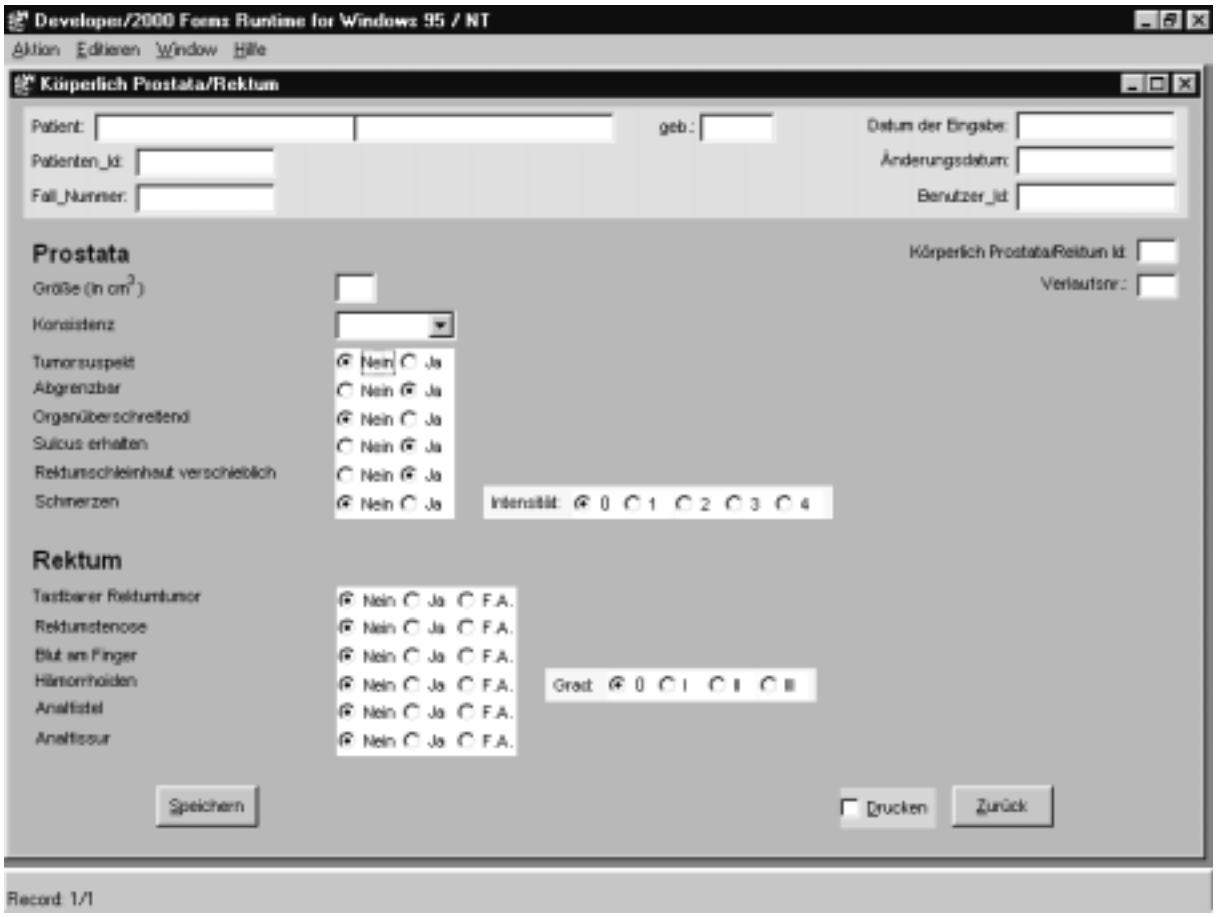

Abbildung 12: Maske *Körperlich Prostata/Rektum*

Bei der Untersuchung der Prostata ist zunächst die *Größe* in *cm* anzugeben. Die *Konsistenz* kann durch Auswahl eines Elementes aus einer Liste (*weich, prall, elastisch, knotig, verhärtet, diffus*) entnommen werden. Die nächsten 6 Merkmale (*Tumorsuspekt, Abgrenzbar, Organüberschreitend, Sulcus erhalten, Rektumschleimhaut verschieblich* und *Schmerzen*) lassen nur eine Beantwortung mit *Nein* oder *Ja* zu, eine Ausweichmöglichkeit (*fehlende Angaben*) gibt es hier nicht. Liegen *Schmerzen* vor, müssen diese in einer Skala von *1* bis *4* genauer spezifiziert werden.

Bei der Untersuchung der Rektums sind die Merkmale *Tastbarer Rektumtumor*, *Rektumstenose*, *Blut am Finger*, *Hämorrhoiden*, *Analfistel* und *Analfissur* zu dokumentieren. Sind *Hämorrhoiden* vorhanden muß der Grad (*I,II* oder *III*) mit angegeben werden.

Den Körperlichen Untersuchungen schließt sich die Sonographie (Ultraschalldiagnostik) an. Sie steht als sichere, schmerzlose, wiederholbare, schnelle und nichtinvasive Untersuchungsmethode an erster Stelle in der Reihenfolge der bildgebenden Verfahren in der Urologie (Urologie, Altwein, Rübben). Sie umfaßt Niere, Hoden, Nebenhoden/Penis, Retroperitoneum/Leber sowie Milz/Prostata.

Maske Sonographie Niere

Diese Maske erfaßt die Ergebnisse der Sonographie der Niere.

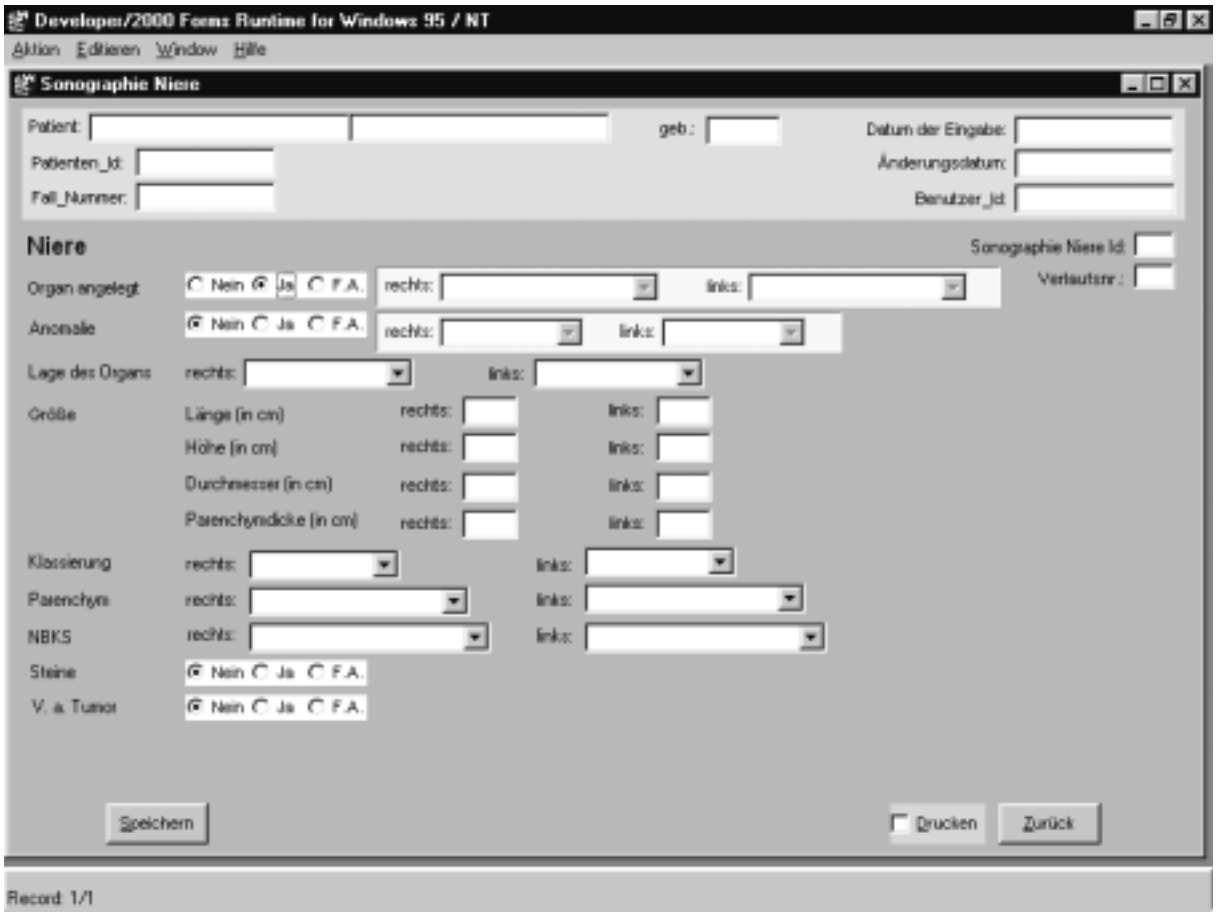

Abbildung 13: Maske *Sonographie Niere*

Da die Niere paarig angelegt ist, müssen die Angaben jeweils für beide Seiten erfaßt werden. Die Voreinstellung für das erste Merkmal *Organ angelegt* ist *Ja*. Erst bei Anklicken von *Nein*

kann zur Detailangabe navigiert werden und es steht eine Auswahlliste von 3 Elementen (*Agenesie*, *maximale Schrumpfniere*, *Zustand nach Nephrektomie*) zur Verfügung. Beim Vorliegen einer *Anomalie* kann diese noch jeweils in *Hufeisenniere* und *Doppelniere* unterschieden werden. Die *Lage des Organs* läßt eine Spezifizierung in *orthotop*, *deszendiert*, *im kleinen Becken* und *medialisiert* zu. Das Merkmal *Größe* verlangt jeweils *in cm* Angaben zu *Länge*, *Höhe*, *Durchmesser* und *Parenchymdicke*. Eine *Klassierung* kann in *Hypotrophie*, *normal*, *Hypertrophie* und *Agenesie* vorgenommen werden. Das *Parenchym* kann mit eines der Elemente *normal*, *verdickt*, *verschmälert/gemindert*, *vernarbt*, *mit Verkalkungen*, *tumorsuspekt*, *hyporeflexiv* oder *hyperreflexi*v beschrieben werden.

Das *Nierenbeckenkelchsystem* ist in *normal*, *Ektasie I° isolierte Kelchektasie*, *Ektasie II° Nierenkelche und Nierenbecken* oder *Ektasie III° mit Harnleiterdarstellung* einzuteilen. Muß bei den Merkmalen *Steine* oder *V.a. Tumor* mit *Ja* geantwortet werden, so öffnet sich jeweils ein Fenster und es kann der genauere *Ort* sowie *Anzahl* und *Größe* näher spezifiziert werden.

Maske Sonographie Hoden

Mit dieser Maske kann die sonographische Untersuchung des Hodens erfaßt werden.

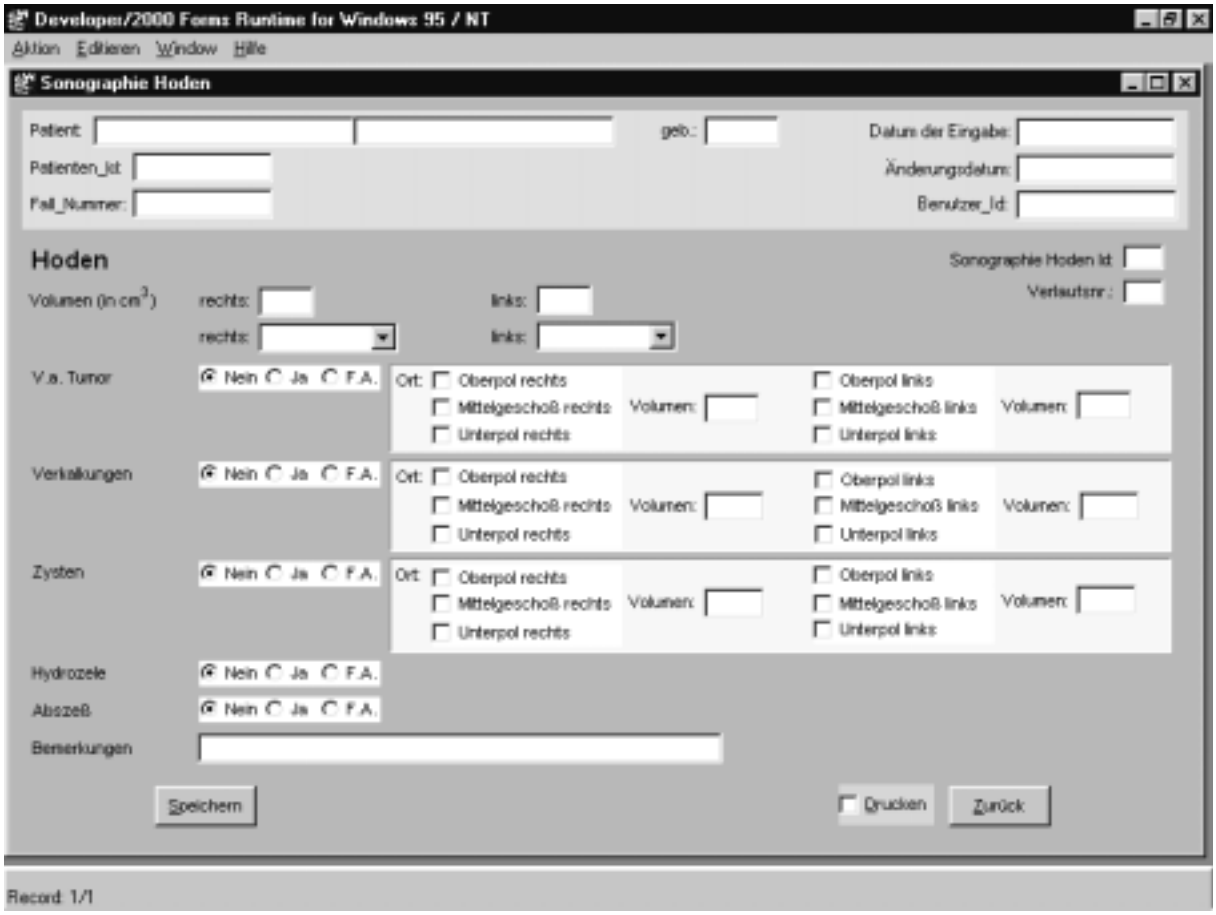

Abbildung 14: Maske *Sonographie Hoden*

Die 6 zu erfassenden Merkmale *Volumen*, *V.a. Tumor*, *Verkalkungen*, *Zysten*, *Hydrozele*, *Abszeß* können durch freitextliche *Bemerkungen* ergänzt werden.

Beim *Volumen* wird neben der Angabe in Kubikzentimeter noch eine genauere Angabe (*normal*, *hyporeflexiv*, *hyperreflexiv)* verlangt. Bei den Merkmalen *V.a. Tumor*, *Verkalkungen* und *Zysten* kann je Seite zwischen drei verschiedenen Ortsangaben gewählt und diese durch eine Volumenangabe ergänzt werden. Bei *Hydrozele* und *Abszeß* muß nur mit *Nein* oder *Ja* geantwortet werden.

Maske Sonographie Nebenhoden/Penis

Diese Maske dient der Erfassung der sonographischen Untersuchung der Nebenhoden und des Penis.

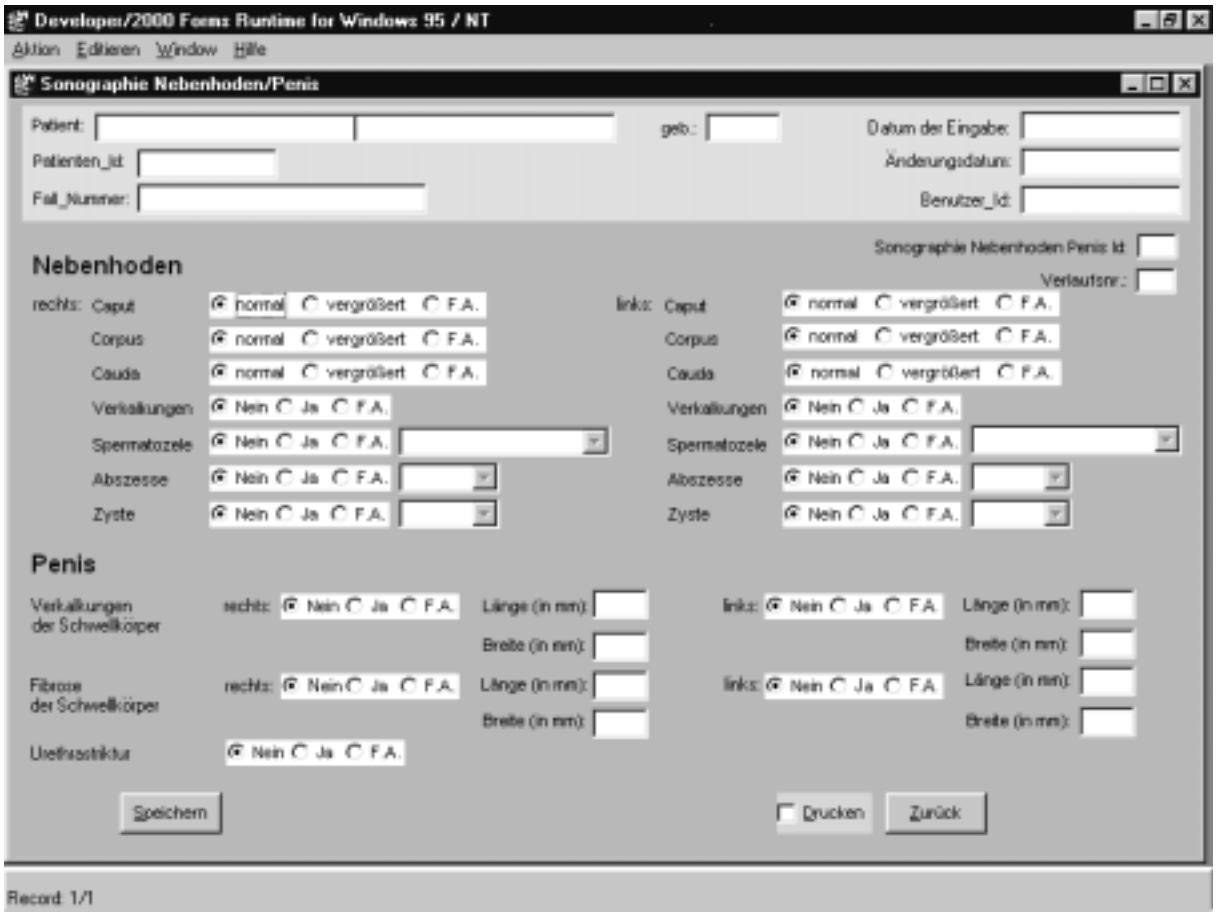

Abbildung 15: Maske *Sonographie Nebenhoden/Penis*

Bezüglich der Nebenhoden sind 7 Merkmale, sowohl für die rechte als auch für die linke Seite, zu dokumentieren. Bei *Caput*, *Corpus*, *Cauda* ist zu bestimmen, ob sie *normal* oder *vergrößert* sind. Die Merkmale *Verkalkungen*, *Spermatozele*, *Abszesse* und *Zysten* sind wiederum mit *Nein* oder *Ja* zu beantworten. Bei der Beantwortung mit *Ja* kann beim Merkmal

Spermatozele noch eine detailliertere Beschreibung in *Samenstrang*, *Nebenhoden Kopf*, *Nebenhoden Körper*, *Nebenhoden Schwanz* ausgewählt werden. *Abszesse* und *Zyste* lassen noch eine Detailangabe in *Caput*, *Corpus*, *Cauda* zu.

3 Merkmale sind bei der sonographischen Untersuchung des Penis zu dokumentieren. Die Merkmale *Verkalkungen der Schwellkörper* und *Fibrose der Schwellkörper* benötigen bei Bejahung eine Längen- und Breitenangabe in *mm* für die rechte und linke Seite. *Urethrastriktur* ist dagegen nur mit *Nein* oder *Ja* zu beantworten.

Sonographie Retroperitoneum/Leber

Die Maske Sonographie Retroperitoneum/Leber erfaßt die sonographischen Untersuchungen des Retroperitoneums und der Leber.

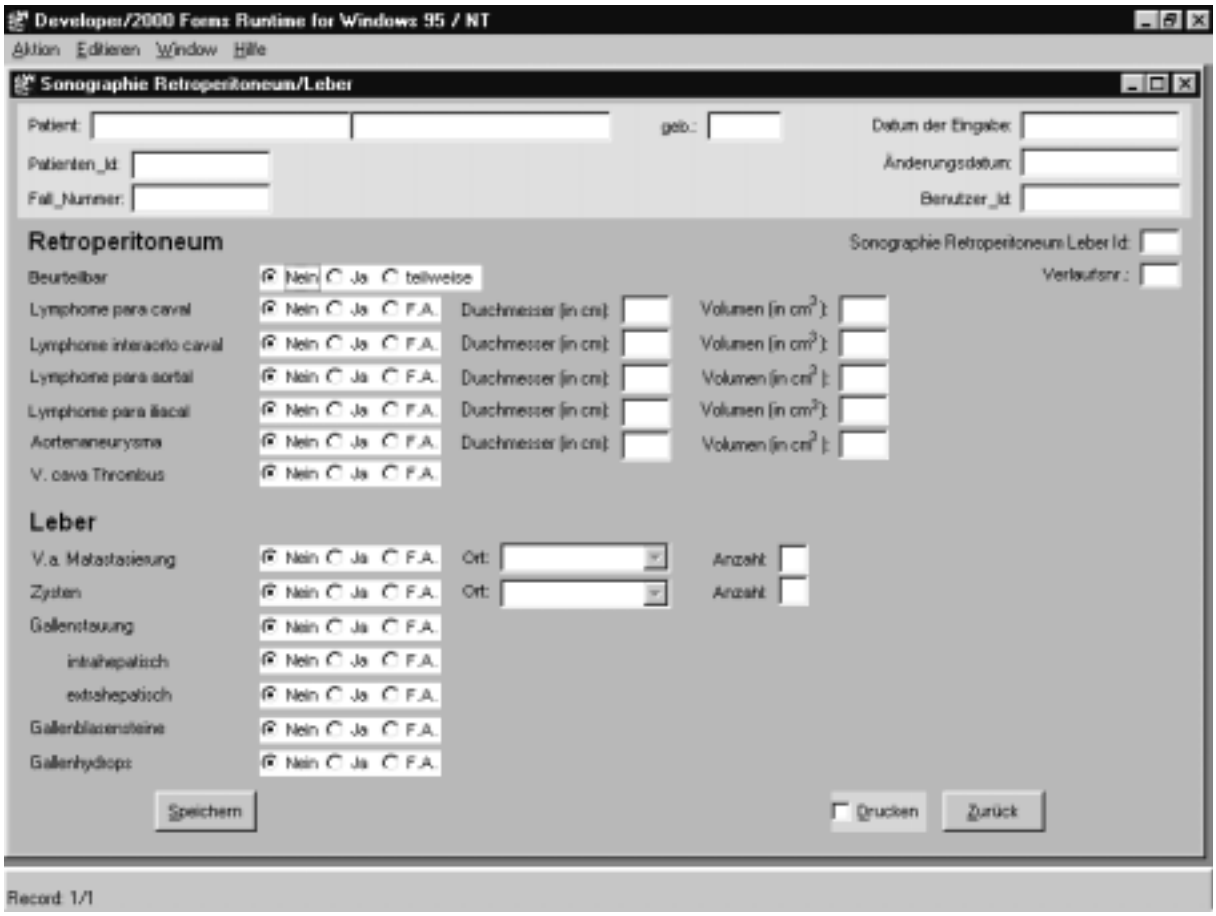

Abbildung 16: Maske *Sonographie Retroperitoneum/Leber*

Beim *Retroperitoneum* sind insgesamt 7 Merkmale zu dokumentieren. Das erste Merkmal *Beurteilbar* kann nur in *Nein*, *Ja* oder *teilweise* unterschieden werden.

Die nächsten 6 Merkmale *Lymphome para caval*, *Lymphome interaorto caval*, *Lymphome para aortal*, *Lymphome para iliacal* und *Aortenaneurysma* müssen bei positivem Befund noch jeweils um die Angabe des *Durchmessers (in cm)* und des *Volumens (in cm³)* ergänzt werden. Beim Merkmal *V. cava Thrombus* ist nur eine Beantwortung mit *Nein* oder *Ja* erforderlich.

Auch bei der Leber sind 7 Merkmale (*V.a Matastasierung*, *Zysten*, *Gallenstauung*, *intrahepatisch*, *extrahepatisch*, *Gallenblasensteine*, *Gallenhydrops*) zu erfassen, die jeweils die Angabe von *Nein* oder *Ja* erfordern. Wird bei den ersten beiden Merkmale mit *Ja* geantwortet, so kann noch eine genauere Spezifikation des *Ort*es und der *Anzahl* erfolgen.

Sonographie Milz/Prostata

Mit dieser Maske werden die sonographischen Untersuchungen von Milz und Prostata erfaßt.

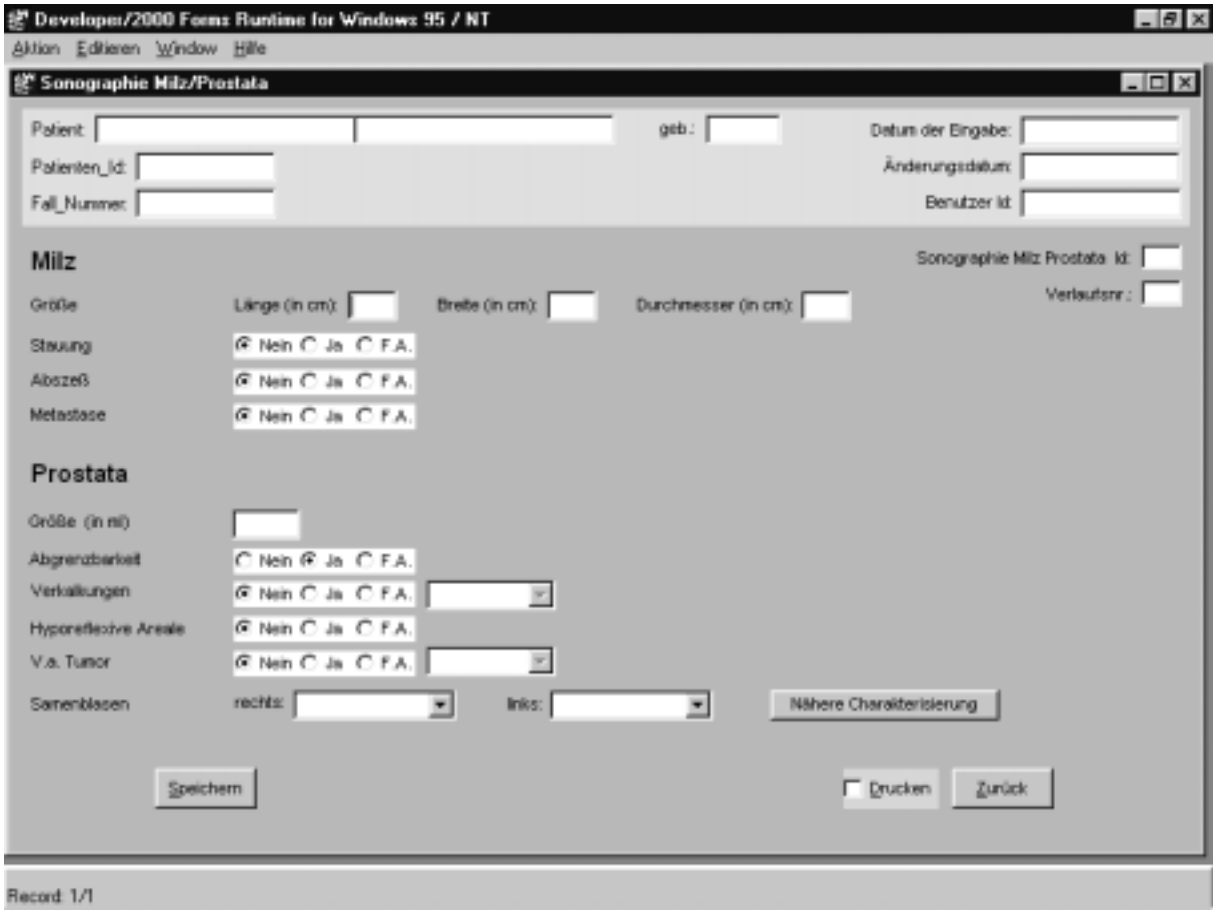

Abbildung 17: Maske *Sonographie Milz/Prostata*

Bei der Milz ist zunächst eine genaue Größenangabe in *Länge*, *Breite* und *Durchmesser* erforderlich. Die Merkmale *Stauung*, *Abszeß* und *Metastase* müssen nur verneint oder bejaht werden.

Auch bei der Prostata ist anfangs eine Größenangabe notwendig. Während *Abgrenzbarkeit* und *Hyperreflexive Areale* nur eine Beantwortung mit *Nein* oder *Ja* verlangen, ist bei *Verkalkungen* und *V.a. Tumor* noch zusätzlich (bei Beantwortung mit *Ja*) eine nähere Angabe (*solitär*, *multilokulär*) gefordert. *Samenblasen* müssen *rechts* und *links* genauer (*normal*, *fehlend*, *symmetrisch*, *asymmetrisch*, *verkleinert*, *vergrößert*, *mit Zyste*) spezifiziert werden. Ein zusätzliches Freitextfeld erlaubt eine *Nähere Charakterisierung*.

Nach der Ultraschalldiagnostik erfolgt die urologische Röntgenuntersuchung. Sie erlaubt eine genaue morphologische Darstellung und kann daher als Basisdiagnostik der Urologie bezeichnet werden. Sie umfaßt Thorax, i.V. Urogramm, CT-Thorax/CT-Abdomen, CT-Schädel/NMR-Schädel, NMR-Thorax/NMR-Abdomen und Knochenszintigraphie.

Röntgen Thorax

Diese Maske dient der Dokumentation der Röntgenuntersuchung des Thoraxes.

Sie umfaßt 8 Merkmale: *Erguß*, *Atelektase*, *Rundherde*, *Fremdkörper*, *Pneumothorax*, *Pneumonie*, *Herzgröße* und *Tumorstatus.* Die ersten 6 Merkmale sind mit *Nein* oder *Ja* zu beantworten. Bei Vorliegen eines positiven Befundes erfordern sie jeweils eine Ortsangabe. Bei *Erguß* steht eine Liste mit Elementen (*rechts*, *links*, *beidseitig*) zur Verfügung, *Atelektase*, *Rundherde, Fremdkörper, Pneumothorax und Pneumonie* können durch Anklicken einer oder mehrerer Lokalisationen spezifiziert werden. Bei *Rundherde* ist zusätzlich noch eine Angabe über die *Anzahl* und den *größten Durchmesser* verlangt.

Für die Beschreibung der *Herzgröße* und des *Tumorstatuses* steht je eine Liste (*normal*, *nicht vergrößert*, *mittel vergrößert*, *stark vergrößert)* bzw. (*no evidence of disease*, *complete remission*, *partial remission*, *partial progression*, *progressive disease*, *stable disease*) zur Verfügung.

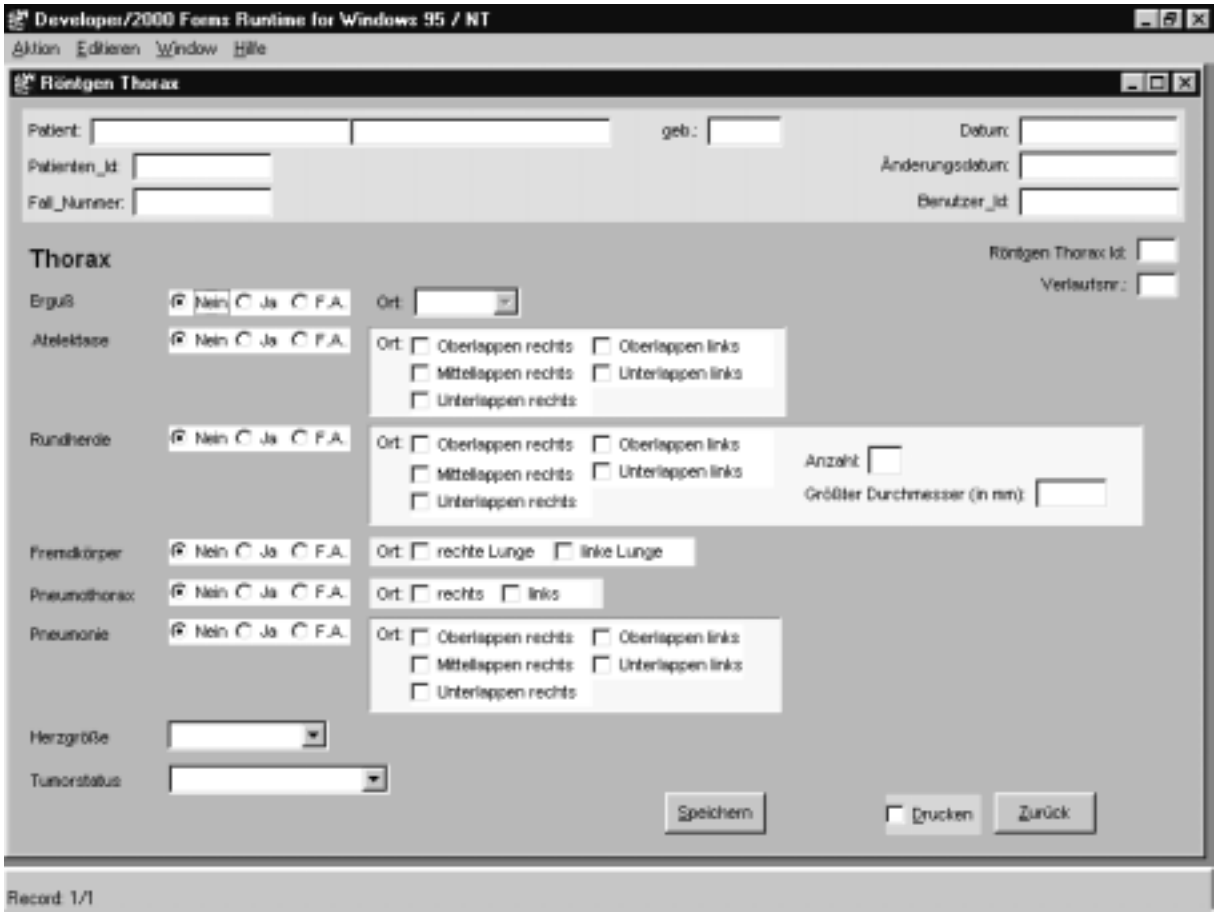

Abbildung 18: Maske *Röntgen Thorax*

Röntgen i.v. Urogramm

Diese Maske dient zur Dokumentation des i.v. Urogramms. Insgesamt sind 8 Merkmale zu erfassen: *V.a. Steine*, *Blasenstein*, *Blasendivertikel*, *V.a. Nierenzyste*, *V.a. Nierenbeckentumor*, *V.a. Nierentumor*, *Abflußbehinderung*, *Harnleiterstein*. Muß bei den Merkmalen *V.a. Steine*, *V.a. Nierenbeckentumor*, *V.a. Nierentumor* und *Harnleiterstein* mit *Ja* geantwortet werden, so kann der genauere Ort durch Anklicken einer oder mehrerer Checkboxen bestimmt werden. Bei den Merkmalen *Blasendivertikel*, *V.a. Nierenzyste* und *Abflußbehinderung* wird zwischen *rechts* und *links* unterschieden, wobei bei der *Abflußbehinderung* noch jeweils der *Grad* (*I – III*) bei positivem Befund mit angegeben werden soll.

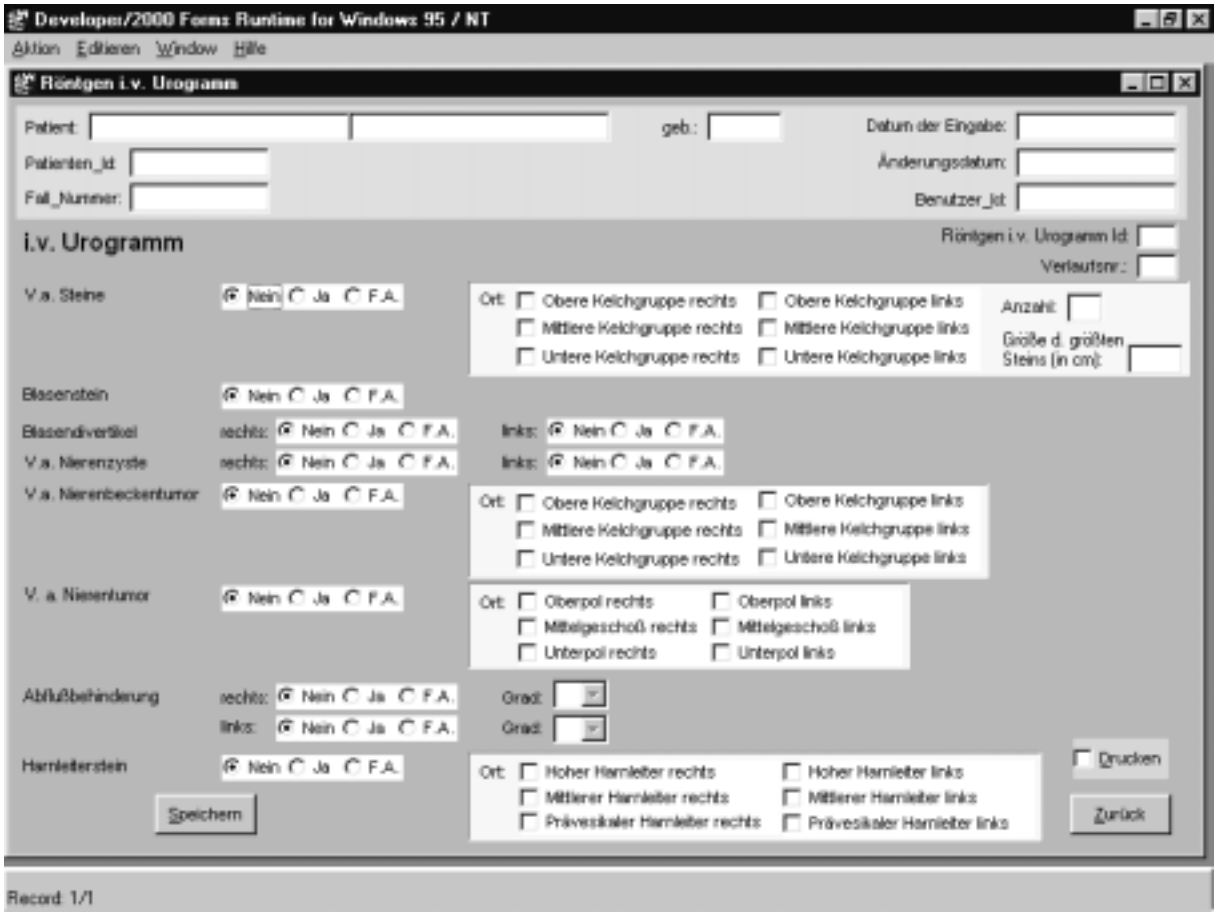

Abbildung 19: Maske *Röntgen i.v. Urogramm*

CT-Thorax/CT-Abdomen

Mit dieser Maske werden die CT-Untersuchungen des Thoraxes und des Abdomens dokumentiert. Beim Thorax sind insgesamt 5 Merkmale zu erfassen, von denen die ersten vier (*Lungenfiliae*, *Mediastinale Lymphome*, *Pleura Metastasen*, *Infiltrierende Tumore*) mit *Nein* oder *Ja* zu beantworten sind. Bei der Beantwortung mit *Ja* muß jeweils die *Anzahl* mit angegeben bzw. bei Vorliegen von *Infiltrierenden Tumoren* der *Ort* noch zusätzlich spezifiziert werden.

Auch beim Abdomen sind alle bis auf das letzte Merkmal zu verneinen oder zu bejahen.

Beim Vorhandensein von *Lokalrezidiven* und *Lymphomen* steht für rechts und links noch eine Liste mit mehreren Elementen (*Nierenlager*, *iliacal*, *Beckenboden*, *para caval*, *para aortal*) zur Verfügung.

Für die Beschreibung des *Tumorstatus* steht für beide Untersuchungen jeweils eine Liste mit sämtlichen Auswahlmöglichkeiten (*no evidence of disease, complete remission, partial remission, partial progression, progressive disease, stable disease*) bereit.

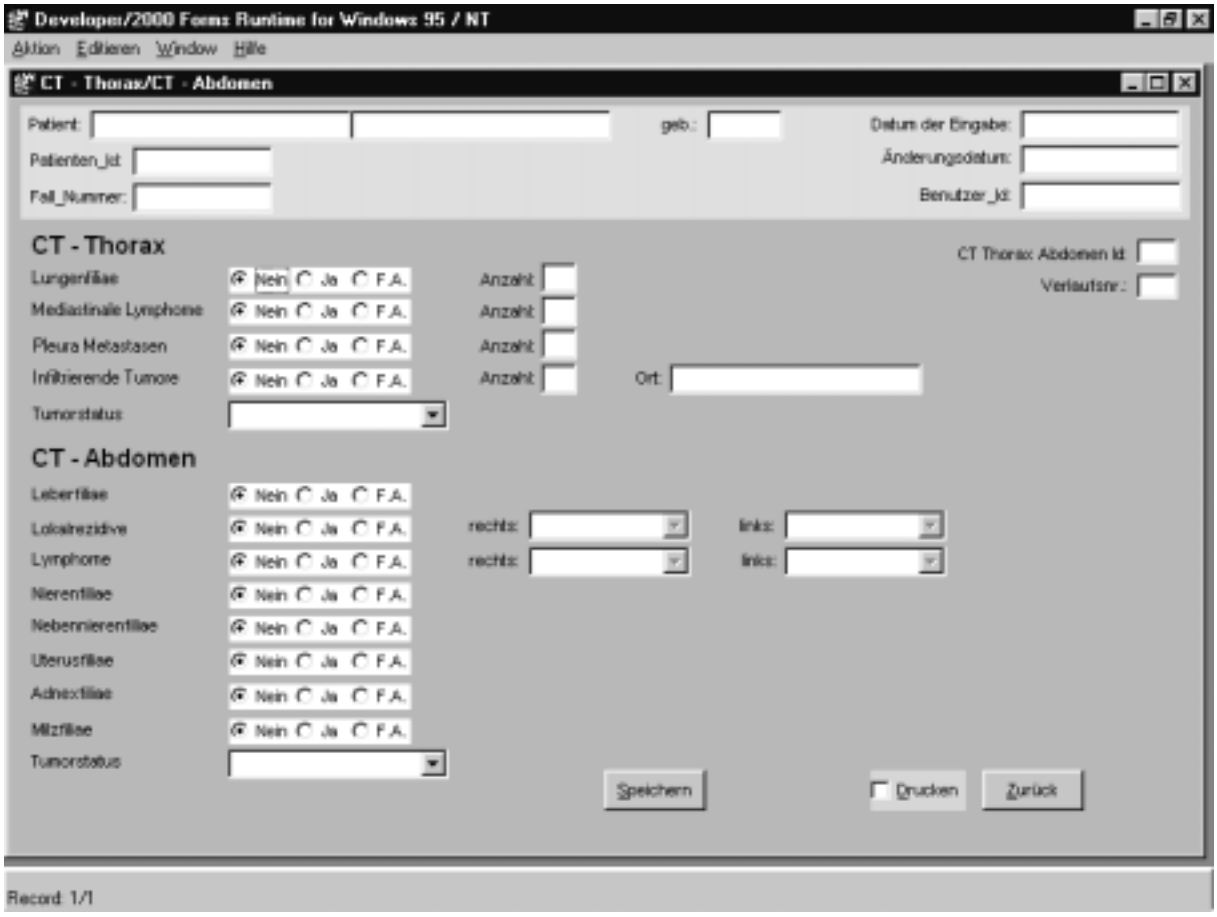

Abbildung 20: Maske *CT-Thorax/CT-Abdomen*

CT-Schädel/NMR-Schädel

Diese Maske erfaßt sowohl die CT- als auch die NMR-Untersuchung des Schädels. Für jede dieser beiden Untersuchungen sind die gleichen 4 Merkmale zu dokumentieren: *Hirnfiliae, Hirndruck, Liquorstau* und *Tumorstatus*. Die erste drei Merkmale sind mit *Nein* oder *Ja* zu beantworten, wobei *Hirnfiliae* bei positivem Befund noch eine detailliertere Angabe der *Anzahl* und des *Ortes* verlangt. Für die Beschreibung des *Tumorstatus* steht wie in der vorherigen Maske eine Liste mit sämtlichen Auswahlmöglichkeiten (*no evidence of disease, complete remission, partial remission, partial progression, progressive disease, stable disease*) zur Verfügung.

Abbildung 21: Maske *CT-Schädel/NMR-Schädel*

NMR-Thorax/NMR-Abdomen

Diese Maske dient der Dokumentation der NMR-Untersuchungen des Thoraxes und des Abdomens. Sie ist analog der Maske CT-Thorax/CT-Abdomen aufgebaut.

Knochenszintigrapie

Mit dieser Maske werden die Ergebnisse der Knochenszintigrapie erfaßt. Bei den ersten beiden Merkmalen (*Degenerative Veränderungen* und *V.a. Knochenfiliae*) ist bei Anklicken von *Ja* eine genauere freitextliche Lokalisationsangabe erforderlich. *Röntgenkontrolle angezeigt* ist nur mit *Nein* oder *Ja* zu beantworten. Für die Spezifizierung des *Tumorstatus* steht, wie auch schon in anderen Masken, eine Auswahlliste (*no evidence of disease, complete remission, partial remission, partial progression, progressive disease, stable disease*) zur Verfügung.

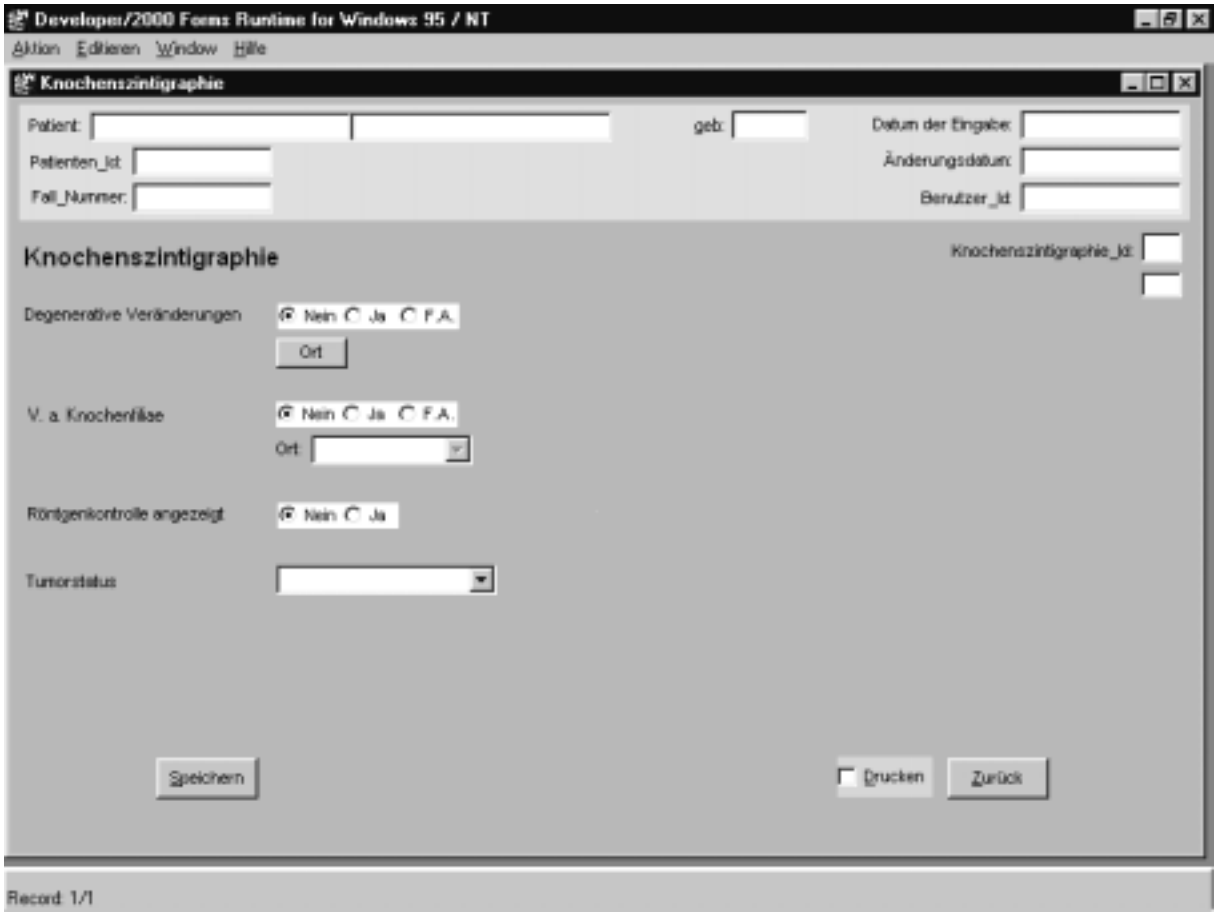

Abbildung 22: Maske *Knochenszintigraphie*

Labordiagnostik

Die Urinuntersuchung stellt den wichtigsten und ersten Schritt der urologischen Diagnostik dar.

Masken Labor: Urin und Labor: Blut

Die Laboruntersuchungen von Urin und Blut werden in den Masken *Labor: Urin* und *Labor: Blut* erfaßt.

In der Urinmaske wird standardmäßig *Urinstatus*, *Urikult* und *Urinzytologie* dokumentiert.

Für *Urinstatus* sind 9 Items mit dem dazugehörigen Probendatum zu erfassen. Bei *Urikult* kann aus einer Liste (*kein Kaliumnachweis, nicht signifikant, 10.000 - 50.000, 50.000 - 100.000, > 100.000*) ein Element ausgewählt werden. *Urinzytologie* läßt sich in eines von fünf Stadien (*I, II, III, IV, V*) einteilen.

Für die Forschung werden fünf Items (*NMP22*, *TM2PK*, *TPA*, *PHI* und *Telomerase*) zur Verfügung gestellt. Das Probendatum ist jeweils für jedes Item einzugeben.

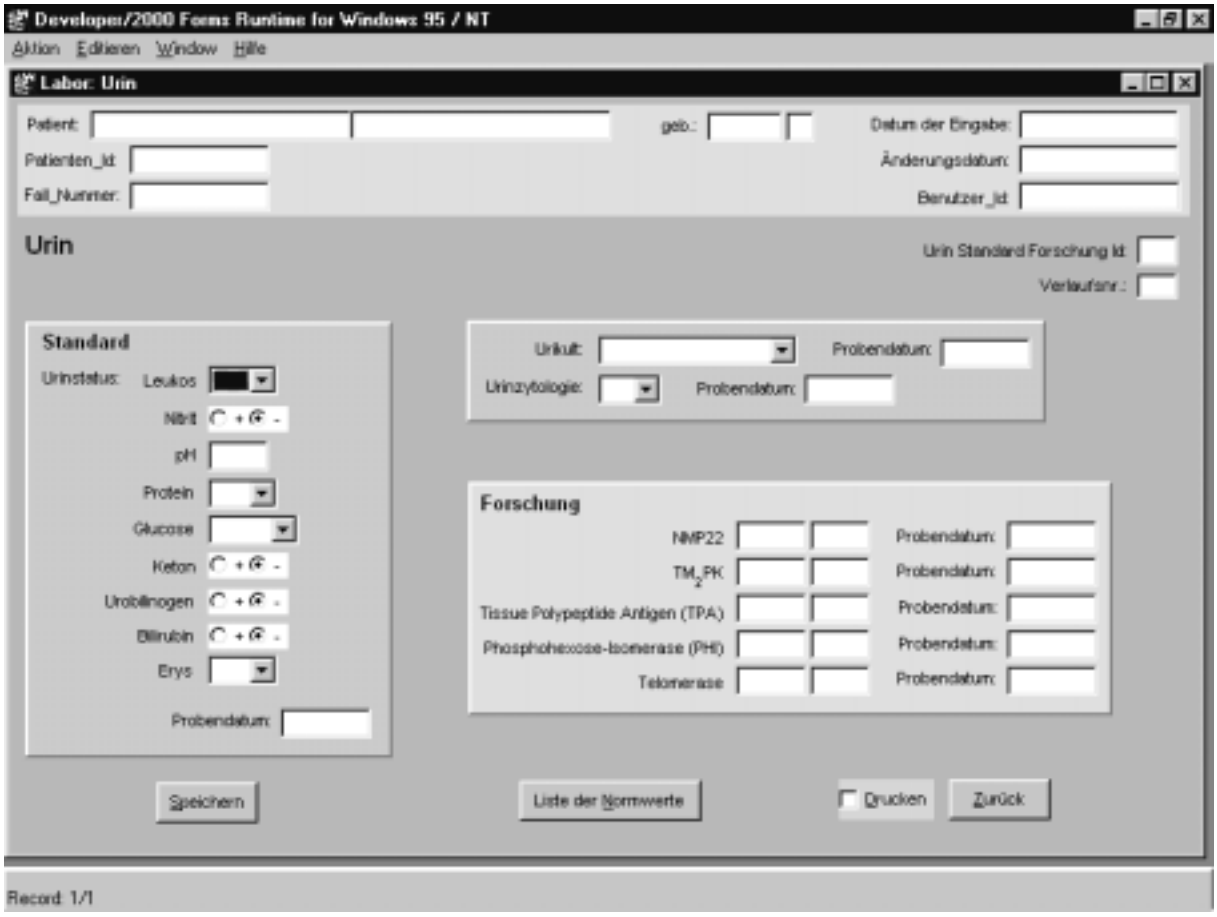

Abbildung 23: Maske *Labor: Urin*

In der Blutmaske werden die Items in sechs unterschiedlichen Gruppen zusammengefaßt: *Standard*, *Tumormarker*, *Blutgase*, *Hämostaseologie*, *Forschung* und *fakultativ*. Für jede Gruppe ist das Probendatum getrennt zu erfassen bzw. muß bei drei Items der Gruppe *Tumormarker* individuell eingetragen werden.

Einige Werte (z.B. Kreatinin und Harnstoff) können in verschiedenen Einheiten angegeben werden. Bei Eingabe eines Wertes in einer Einheit wird automatisch in die andere Einheit umgerechnet. Beide Werte werden mit der dazugehörigen Einheit in der Datanbank abgespeichert.

Liegt in einer der Labormasken ein eingegebener Wert außerhalb der Norm so wird dieser automatisch grau unterlegt. Dadurch werden kritische Werte hervorgehoben und können nicht so leicht übersehen werden. Ein *Liste der Normwerte* kann über den entsprechenden Knopf eingesehen werden.

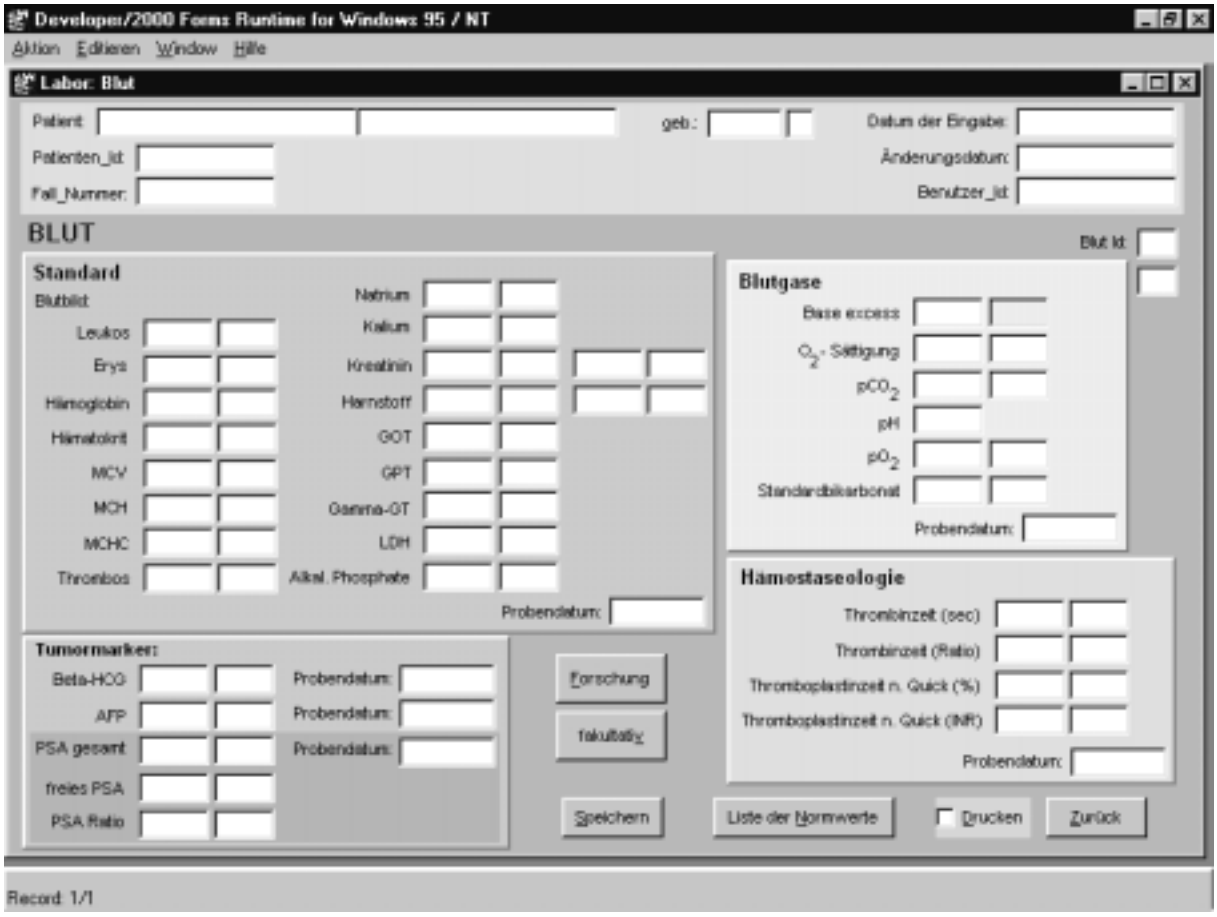

Abbildung 24: Maske *Labor: Blut*

4.6 Technische Merkmale

Die Erweiterung des GTDS wurde, wie das GTDS selbst, mit dem relationalen Datenbanksystem ORACLE(TM) und seinen Tools erstellt. Neben dem Datenbank-Kern (Oracle RDBMS, V.8.05) wurden die 4GL-Werkzeuge von Oracle verwendet. Zur Masken- und Menüerstellung diente der Developer/2000 V.1.6 mit SQL*Forms V.4.5. und SQL*ReportWriter V.2.5.

Das Programm läuft zur Zeit noch im Client/Server-Betrieb, d.h. der Datenbank-Kern auf einer Sun Sparc, die Toolprogramme peripher auf der PC-Arbeitsstation. Unterstützt werden die Betriebssysteme Windows NT und Windows 95.

Bei geeigneter Rechnerausstattung ist geplant, das System als Web-DB Anwendung laufen zu lassen.

Oracle ist der größte nicht an eine bestimmte Hardware gebundene Datenbankhersteller und stellt ein offenes System zur Verfügung, d.h. es erlaubt Werkzeuge von verschiedenen Herstellern zu verwenden.

Auf Standards basierende Interaktionen zwischen Oracle Developer und anderen Applikationen und Werkzeugen wird durch OCX/ActiveX-Controls, OLE (Object Linking und Embedding) und DDE (Dynamic Data Exchange) ermöglicht. Mit Unterstützung für vielfältige Multimediaformate, ergänzt durch ein offenes API, gibt der Oracle Developer die Flexibilität, Applikationen zu erweitern und andere Komponenten zu integrieren.

Der Oracle Developer bietet auch transparenten Zugriff auf andere Datenbanken, einschließlich Oracle Rdb, Microsoft SQL Server, Informix, Sybase und DB2. Der Datenbankzugriff wird durch Datenbanktreiber, das Oracle Developer API oder Oracle Gateways realisiert.

Das bedeutet, daß das Dokumentationssystem keine feste Bindung an einen Rechner oder ein Betriebssystem eingeht. Somit ist die Gefahr, daß das System an eine bestimmte System- oder Toolsoftware gebunden und schon bald überholt ist und nicht mehr weiterentwickelt werden kann, sehr gering.

4.6.1 Datenmodell

Die Daten der GTDS-Erweiterung werden in einem relationalen Datenbanksystem in einer der dritten Normalform genügenden Datenstruktur gespeichert [siehe Diagramm 1].

Insgesamt umfaßt die Erweiterung des GTDS 18 zusätzliche Masken (eine Übersichtsmaske und 17 Masken mit jeweils ein oder zwei speziellen urologischen Untersuchungen). Zu jeder

urologischen Untersuchung gibt es eine (Haupt-) Tabelle, zu der es noch jeweils bis zu 8 (Unter-) Tabellen (für Detailangaben) geben kann. Im ganzen gibt es 46 Tabellen mit ungefähr 1000 Spalten.
Das System ist flexibel gehalten und kann wechselnden Anforderungen angepaßt werden. Sollte eine neue urologische Untersuchung hinzukommen, wird eine Haupttabelle mit den dazugehörigen Untertabellen angelegt und die darüberliegende Maske erstellt. Auch können die bestehenden Masken durch neue Felder und Detailangaben ergänzt werden.

Im Diagramm 1 stellen die Entitätsmengen die Datenbanktabellen Externer_Patient sowie jeweils die Haupttabellen und Untertabellen dar. Die Verbindung zwischen den Tabellen zeigt eine identifizierende Beziehung zwischen einer und vielen Entitäten (optional). Die im oberen Feld der Entitätsmenge aufgeführten Attribute veranschaulichen die Primärschlüsselfelder der Datenbanktabellen.

Der Primärschlüssel einer Haupttabelle besteht jeweils aus zwei Feldern:
- Patienten_Id
- Haupttabellenname _Id

Der Primärschlüssel einer Untertabelle besteht jeweils aus drei Feldern:
- Patienten_Id
- Haupttabellenname_Id
- Untertabellenname_Id

Die Attribute im unteren Feld stellen die übrigen Felder dar. (FK) neben einem Feld zeigt einen Fremdschlüssel an.

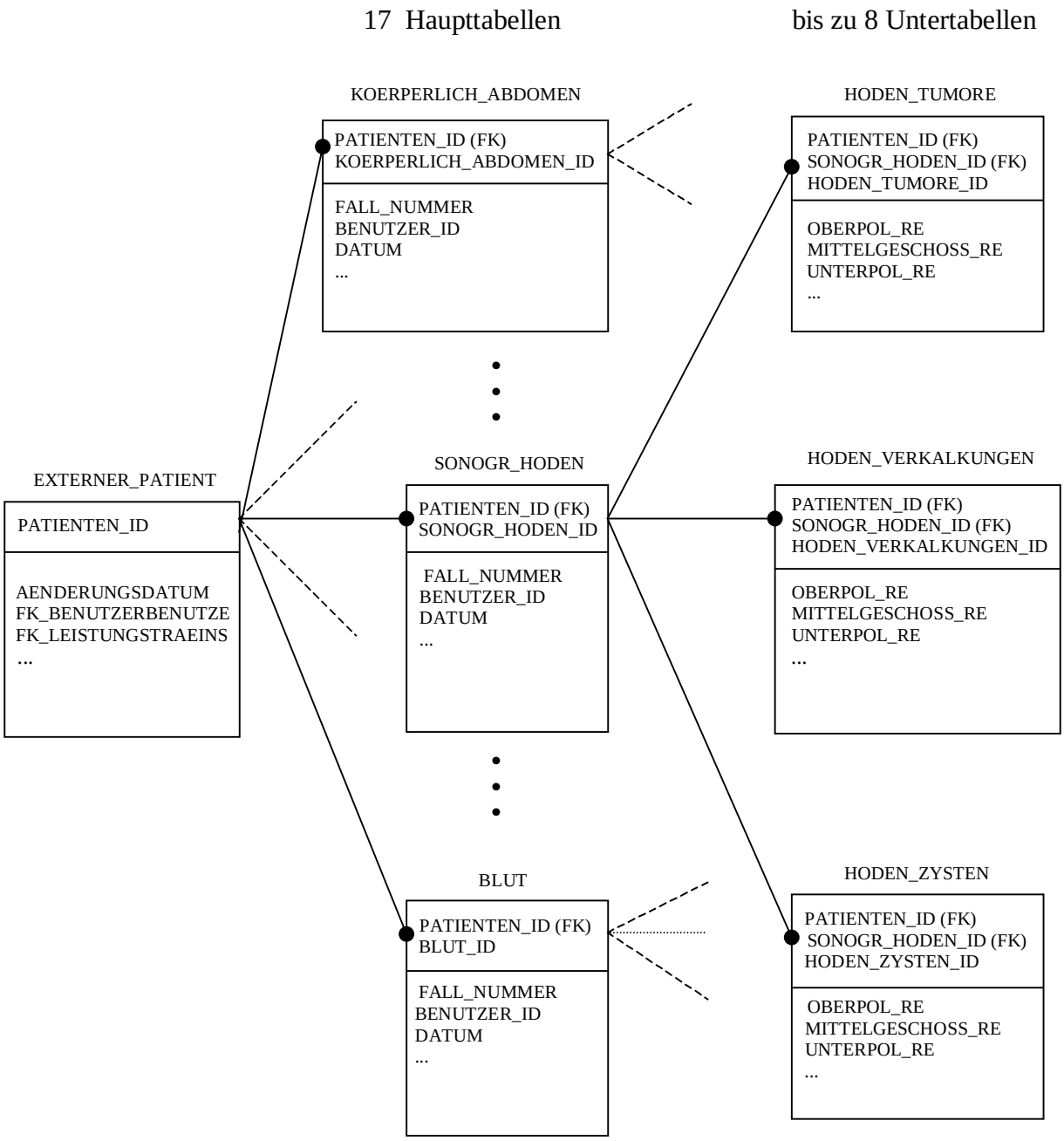

Diagramm 1: Datenmodell der GTDS-Erweiterung

4.6.2 Datenintegrität

Das erweiterte GTDS wurde als integraler Bestandteil des Klinikinformationssystems WING realisiert. Über eine HL7-Schnittstelle werden Stammdaten aus der Verwaltung und OP-Daten (Diagnose- und Prozedurdaten) aus dem MedAccess System übertragen und im GTDS zur Verfügung gestellt (siehe Kapitel 5).

4.6.3 Datensicherheit

Das Gießener Klinikumsnetz ist physikalisch vom Internet getrennt, d.h. es gibt keine Verbindung zwischen beiden Netzen. Um die Arbeit mit dem System zu beginnen, benötigt jeder Anwender eine Benutzerkennung und ein Paßwort. Gewöhnliche Anwender gelangen i.d.R. nach dem Einloggen direkt ins Dokumentationssystem (kein Zugriff auf Betriebssystem-Ebene).

5. Bereitstellung aktueller Daten über eine HL7-Schnittstelle

5.1 Problematik

Ende der siebziger und in den achtziger Jahren wurden in den Kliniken unterschiedliche EDV-Anwendungen eingeführt, um den Informationsbedarf an klinischen Daten zu decken und organisatorische Abläufe zu automatisieren. Dabei waren die Anwendungen jedoch weitgehend auf den Einsatz im administrativen Bereich beschränkt. Erst nach und nach entstanden im klinischen und pflegerischen Bereich einzelne Abteilungssysteme, zuerst im Labor, dann in der Radiologie, schließlich in den Ambulanzen und Stationen.

Daneben sind in zunehmendem Maße medizinische Geräte zur Diagnose im Einsatz, die entweder selbst hochspezialisierte Rechner sind oder die für die Gerätesteuerung, Datenauswertung und -speicherung und anderes mehr entsprechende Computerunterstützung benötigen.

Meistens handelt es sich dabei um Einzelanwendungen, die nur Teile der EDV-technischen Belange in den einzelnen Abteilungen abdecken und die keine bzw. nur mit erheblichem Aufwand Kommunikation untereinander zulassen [Jostes 1993]. Es gibt beispielsweise Softwaresysteme für die Patientendatenverwaltung und die Abrechnung mit den Kostenträgern, die Finanzbuchhaltung, die Datenverarbeitung im Labor und in der Radiologie, für die Leistungserfassung sowie Dokumentationssysteme etc.. Die verteilten Systeme zeichnen sich durch einen hohen Grad an Heterogenität aus: sie stammen von verschiedenen Herstellern, arbeiten auf unterschiedlichen Betriebssystemen und wurden fast immer als autarke Anwendungen entworfen [Steimke 1995].

Durch die Autonomie entsteht eine funktionale Überlappung der verschiedenen operationalen Systeme, was zur Folge hat, daß Daten, für eine vollständige Dokumentationsdatenbasis, bisher auch mehrfach in verschiedene Systeme und von verschiedenen Institutionen eingegeben werden müssen. Dadurch entsteht nicht nur ein erheblicher Mehraufwand, sondern die Daten sind untereinander auch häufig inkonsistent.

Ärzte verwenden andererseits zur Dokumentation größtenteils noch gedruckte Formulare. Sie müssen dabei jeweils die Stammdaten des Patienten erfassen, Diagnose-, Prozedur- und alle weiteren Dokumentationsschlüssel in den entsprechenden Büchern nachschlagen und doku-

mentieren, und dabei das Gesamtbild der Krankheit in Relation zu anderen Fällen beibehalten. Diese Methode ist sehr fehleranfällig und zeitintensiv, insbesondere auch durch das nachträgliche Übernehmen der Daten vom Formular in das EDV-System mit Hilfe von Dokumentaren. Außerdem erhöht dieser Schritt die Fehlerquote durch Probleme beim Entziffern und natürlich auftretende Flüchtigkeitsfehler.

Klinische Daten sollten deshalb möglichst am Enstehungsort EDV-gerecht vom ärztlichen Personal dokumentiert werden. Eine wichtige, zusätzliche Prüfung auf Richtigkeit und Konsistenz der Daten kommt dadurch zustande, daß die Eingabe direkt durch Ärzte vorgenommen wird. Letzten Endes kann nur das Fachpersonal, basierend auf allen verfügbaren Daten, für die Korrektheit der Eingaben bürgen. Weitere Hilfen wie z.B. Arztbriefschreibung und Nachsorgeunterstützung, steigern die Motivation zusätzlich.

Für eine weitergehende Akzeptanz des GTDS bei der Anwendung im klinischen Bereich ist es daher notwendig, einen automatisierten Übertrag und Abgleich aller Daten zu gewährleisten. Mit Hilfe einer Online-Übersicht aller verfügbaren Daten und durch die Verwendung von Eingabehilfen kann die Qualität der Daten entscheidend verbessert werden.

5.2 WING und MedAccess

Grundlage jeglicher Datenübertragung am Klinikum der Universität Gießen ist ein seit dem Jahre 1989 bestehendes Klinikinformationssystem mit dem Namen WING, welches auch im Laufe der Zeit immer weiterentwickelt wurde. WING steht hierbei für Wissensbasiertes Informationsnetzwerk-Universitätsklinikum-Gießen. Es handelt sich hierbei um ein Rechnerverbundsystem, bestehend aus einem zentralen Datenbankrechner (ein Tandemsystem mit non-stop Verfügbarkeit), auf dem alle Patientendaten gespeichert werden, und mehreren dezentralen Systemen, wie z.B. UNIX-Rechner bzw. Personal-Computernetzwerke. Über ein umfassendes LAN (Local Area Network) sind alle Abteilungen des Klinikums miteinander verbunden.
Das Netzwerk basiert auf Ethernet mit einem Backbone, das sich in den nächsten Jahren Richtung ATM bewegt. Es unterstützt TCP/IP, Novell's IPX, NETBIOS und DEC-LAT-Protokoll. Die gewählte Softwarearchitektur ist prinzipiell offen für die Integration von Message Presentation Standards wie HL7.

Die Konzeption gründet sich auf Erfahrungen, die mit der Entwicklung und dem Betrieb diverser Teilkomponenten eines Klinikkommunikationssystems (z.B. Laborsystem, Patientenverwaltungssystem, Rechnerkopplung) in den zurückliegenden Jahren gesammelt werden konnten.
Eine Schlüsselrolle nehmen dabei die in den Jahren 1984 bis 1987 gesammelten Erfahrungen mit einer Teilinstallation des amerikanischen Klinikinformationssystems HELP auf dem Rechner des Instituts für Medizinische Informatik ein. Von diesem wurde das Konzept der in den Datenfluß eines Klinikkommunikationssystems integrierten entscheidungsunterstützenden Funktionen übernommen [Michel 1990].

Zu den für das GTDS relevanten Daten gehören die Patientenstammdaten. Diese werden in den meisten Kliniken wie auch in der Urologischen Klinik bereits auf Rechnern der Verwaltung erfaßt. Hierzu wird das in das Klinikinformationssystem integrierte und schon in Abschnitt 4.3 erwähnte klinische ADT-Subsystem PILS für Patientenaufnahme, Verlegung und Entlassung verwendet.

Seit 1995 wird MedAccess in der Urologische Klinik der Universitätsklinik Gießen eingesetzt. MedAccess ist ein Planungs- und Dokumentationssystem, das vor allem für drei Aufgaben konzipiert wurde [MedAccess]:

1. Planung und Organisation klinischer Abläufe:
   Einbestellplanung, OP-Planung, Verlaufsdokumentation, OP- Personalplanung, Dienstplanung, klinische Textverarbeitung, Kommunikation (e-Mail).

2. Erleichterte Dokumentation im Sinne des Gesundheitsstrukturgesetzes:
   Diagnoseverschlüsselung (ICD-9) und Therapieverschlüsselung (ICPM) zur Ermittlung von Fallpauschalen und Sonderentgelten, OP-Materialverbrauch.

3. Dokumentation und Auswertung nach wissenschaftlichen Gesichtspunkten:
   Klinisch - wissenschaftliche Verschlüsselungen (AO-Frakturklassifikation, TNM-Staging, graphische Verschlüsselungstools) in Kombination mit Klartexteingaben, Auswertungstools und Datenexporten.

Dieses OP-System ist auf einem Macintosh Computer installiert und greift auf die Daten der Tandem zu. Mit dem System werden die Patientenstammdaten durch Diagnosen und Prozeduren während der Behandlung im OP ergänzt. MedAccess enthält den kompletten 4-stelligen ICD-9 Schlüssel mit seinen Originaltexten, welche jedoch oftmals mißverständlich und zum Suchen ungeeignet sind. Deshalb wurden die ICD-9 Diagnosen durch ein Synonymwörterbuch mit durchschnittlich 5-10 Synonymen pro ICD-Diagnosen ergänzt. Das Synonymwörterbuch ist erweiterbar und kann vom Benutzer beliebig erweitert werden. Mit Hilfe dieses Thesaurus werden durch den Anwender Klartexteingaben automatisch nach ICD-9 codiert. Die Praxis hat gezeigt, daß die computergestützte Verschlüsselung nicht nur schneller, sondern auch exakter ist als das bekannte Wälzen von ICD-Buchbänden. Einmal kodierte Diagnosen können auch als Aufnahmediagnose, Hauptdiagnose, Nebendiagnose, Entlassung oder Komplikation klassifiziert werden.
Auch Operationen (Prozeduren) können im MedAccess analog zur ICD-Codierung mit Hilfe des umfangreichen Therapiethesaurus verschlüsselt werden. Hierzu wird der gesetzlich vorgeschriebene ICPM – Schlüssel verwendet [MedAccess]. Die Erfassung der Diagnose- und Prozedurdaten mittels MedAccess wird nicht unter wissenschaftlichen Gesichtspunkten betrieben, sondern dient rein merkantilen Zwecken.

Es ist nun wichtig, daß alle von der Verwaltung und vom MedAccess-System erfaßten und auf dem Zentralrechner gespeicherten Datensätze über das Kommunikationsprotokoll HL7 auf den GTDS Datenbankserver übertragen, in das für die GTDS-Tabellen notwendige Format gebracht und eingefügt werden. Da es für diesen spezifischen Ablauf keine allgemein verfügbare Software gab, mußten extra Programme entwickelt bzw. angepaßt werden. Der Ablauf der Datenerfassung- und übertragung ist in Abbildung 25 schematisch dargestellt.

Im nächsten Abschnitt soll auf das HL7-Kommunikationsprotokoll sowie auf die für das GTDS relevanten und übertragenen Daten näher eingegangen werden.

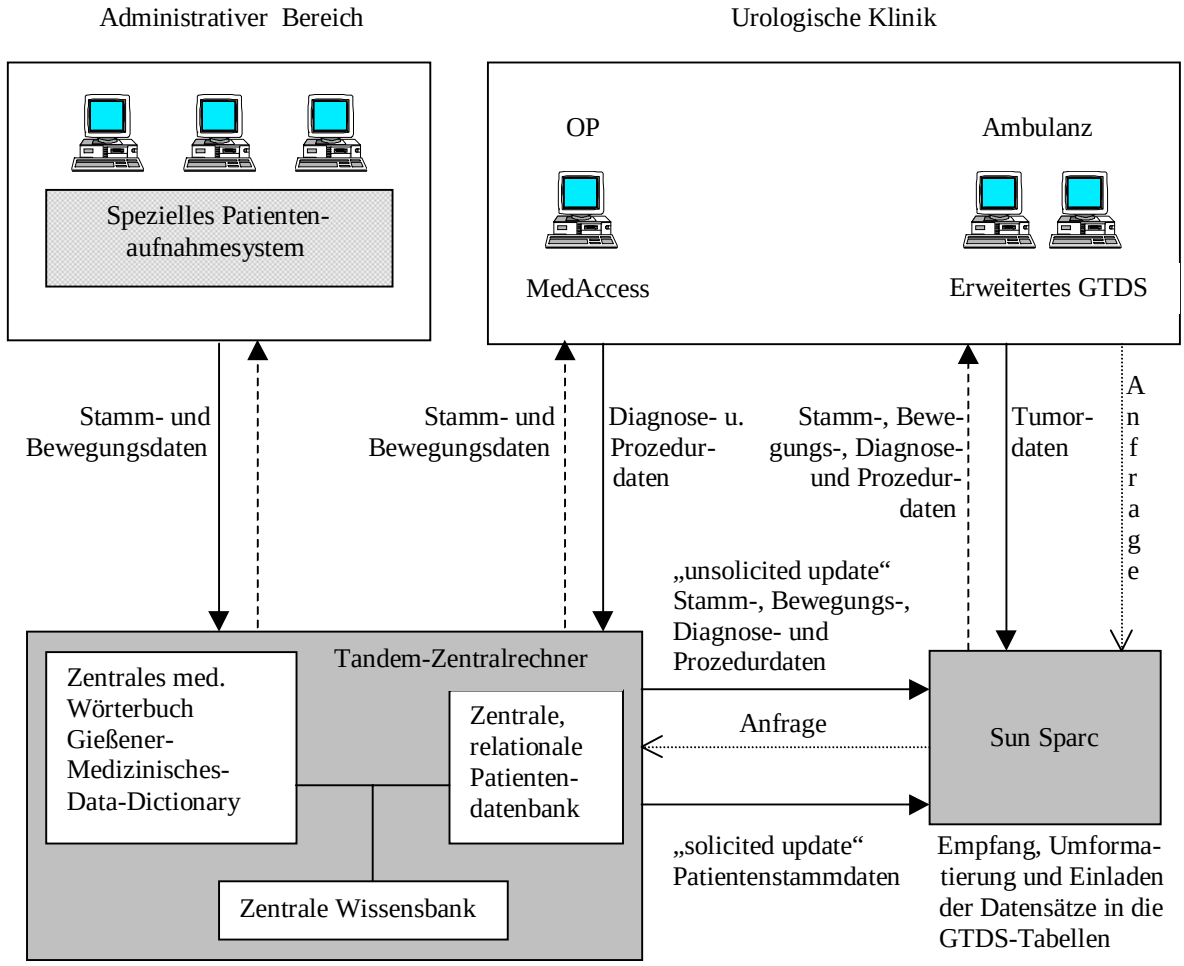

Abbildung 25: Datenerfassung und -übertragung zwischen administrativen Bereich, Med-Access und erweitertem GTDS

Das Rechnerverbundsystem, besteht aus einem zentralen Datenbankrechner (Tandem), auf dem alle Patientendaten gespeichert werden, und mehreren dezentralen Systemen. Über ein umfassendes LAN (Local Area Network) sind alle Abteilungen des Klinikums miteinander verbunden. Zu den für das GTDS relevanten Daten gehören die Patientenstammdaten. Diese werden in den meisten Kliniken wie auch in der Urologischen Klinik bereits auf Rechnern der Verwaltung (Patientenaufnahmesystem) erfaßt. Das OP-System MedAccess ist auf einem Macintosh Computer installiert und greift auf die Daten der Tandem zu. Mit diesem System werden die Patientenstammdaten durch Diagnosen und Prozeduren während der Behandlung im OP ergänzt. Es werden nun die von der Verwaltung und vom MedAccess-System erfaßten und auf dem Zentralrechner gespeicherten Datensätze über das Kommunikationsprotokoll HL7 auf den GTDS Datenbankserver (Sun Sparc) übertragen (entweder über einen automatisierten Abgleich oder über ein Anfrage) und in das für die GTDS-Tabellen notwendige Format gebracht und eingefügt und stehen dann dem Dokumentationssystem (erweitertes GTDS) zur Verfügung.

5.3 Health Level Seven (HL7)

Health Level Seven ist ein herstellerunabhängiges Übertragungsprotokoll, das in den USA als Quasi-Standard für den Austausch elektronischer Daten in Gesundheitsinformationssystemen große Verbreitung gefunden hat. Es wird dort seit März 1987 von einer HL7 Working Group, bestehend aus Herstellern, Anwendern, Beratungsunternehmen und verschiedenen

Institutionen fortentwickelt. Darüberhinaus haben sich in einigen europäischen Ländern nationale HL7-Arbeitsgruppen etabliert [Jostes 1993].
Level 7 bedeutet dabei wiederum, daß sich dieser Standard auf der Anwendungsschicht, also der Schicht 7 des OSI-Schichtenprotokolls befindet. Ende des Jahres 1994 wurde die Version 2.2 herausgegeben. Sie liefert Nachrichten für [Health Level Seven 1992]:

- Aufnahme, Entlassung, Verlegung (ADT)
- Anforderungseingänge für Hilfsdienste, Arzneimittel, Material und Diät
- Mitteilung über den Status /das Ergebnis der Untersuchung
- Finanzwesen
- Master File Indices
  (Master File Messages sollen die Modifikation von Stammdaten-Verzeichnissen aus verschiedenen Anwendungssystemen ermöglichen)
- Anfragen (Queries)

Damit wird auf internationaler Ebene der gesamte Bereich der Krankenhausorganisation abgedeckt, im Gegensatz zu vielen anderen Protokollen, die nur auf einzelne Anwendungsbereiche beschränkt sind [Jostes 1993]. Version 2.3 mit zusätzlichen Nachrichten für Pflege, Ressourcenplanung und Patientenakten wurde 1996 veröffentlicht. Auch im Klinikum der Justus-Liebig-Universität wird die Kommunikation zwischen verschiedenen Rechnersystemen zunehmend im HL7 Standard durchgeführt. Zur Zeit wird noch Version 2.2 verwendet.

Neben dem unmittelbaren Austausch einzelner Informationseinheiten erlaubt HL7 die Zusammenfassung mehrerer Transfervorgänge zu einem Batch. Auch die Nachrichtenübertragung mittels File Transfer ist möglich. Das Protokoll stellt keine spezifischen Anforderungen an die verwendete Netzwerkarchitektur. Vielmehr unterstützt es Konfigurationen von der einfachen RS-232-Terminalverbindung bis zum voll ausgebildeten Rechnernetz [Jostes 1993].

Eine HL7-Nachricht besteht aus zwingenden und optionalen Segmenten, die im Standard definiert sind. Ein Segment besteht wiederum aus Datenfeldern, die die zu übertragenden originären Informationen sowie für den Transfer notwendige, zusätzliche Metainformationen enthalten. Innerhalb eines Segments ist für die Datenfelder festgelegt, ob es sich um „Muß-“ oder „Kannfelder“ handelt. Nur zwingend notwendige Datenfelder sind als Mußfelder gekennzeichnet, der überwiegende Teil ist optional [Steimke 1995]. Dabei kann ein Datenfeld aus Komponenten und Subkomponenten zusammengesetzt sein. Für jedes Datenfeld eines

Segments ist im Protokoll neben dem jeweiligen Inhalt und Datentyp auch die maximale Länge und die Anzahl zulässiger Wiederholungen festgelegt [Jostes 1993].

Die Kodierregeln definieren auch, daß das empfangende System Felder oder Segmente ignorieren soll, wenn es nicht in der Lage ist, diese zu verarbeiten. Dies sollte nicht als Fehler behandelt werden. Durch diese Vereinbarungen wird die Flexibilität und Anpassungsfähigkeit des Standards an lokale Erfordernisse erhöht.

Nachrichten werden durch spezielle Triggerereignisse initiiert und sind im Standard definiert. Sie stellen eine Art Nachrichteninhaltsanzeiger dar und bestehen aus einem 3-stelligen Kürzel des auslösenden Ereignisses. Triggerereignisse können unaufgefordert übertragen werden, wenn ein vordefiniertes Triggerereignis auftritt. Sie stützen sich auf die Annahme, daß jedes Ereignis in der Gesundheitsversorgung ebenfalls ein Startsignal zum Versenden von neuen bzw. aktualisierten Daten initiiert.
So ist beispielsweise das Ereignis, daß ein Patient neu aufgenommen (Ereigniscode A01), verlegt (Ereigniscode A02) oder entlassen (Ereigniscode A03) wird, ein Vorgang, der anderen Systemen mitgeteilt werden muß. Sofern diese Informationen in weiteren DV-Systemen benötigt werden und zwischen den Applikationen ausgetauscht werden müssen, ist dies die notwendige Information, wie die beteiligten Kommunikationspartner die eintreffende Nachricht zu behandeln haben. Dies kann beispielsweise ein Daten-Update in der lokalen Datenbank zur Folge haben.

Die empfangende Anwendung antwortet dann mit einer weiteren Nachricht. Das kann je nach Situation eine Empfangsbestätigung (Acknowledgement) oder eine Fehlermeldung sein.
Seit der Version 2.2 werden zwei unterschiedliche Typen von Bestätigungsnachrichten unterschieden, die Akzeptiert- (accept) und die Anwendungsbestätigung (application acknowledgement). Die Akzeptiertbestätigung bestätigt, daß die Nachricht erfolgreich übertragen und beim empfangenden System gespeichert wurde. Beim Erhalt der Bestätigungs-nachricht muß sich der Sender nicht mehr länger um die Nachricht kümmern.
Die Anwendungsbestätigung bestätigt die erfolgreiche Verarbeitung der syntaktisch und semantisch richtigen Nachricht im empfangenden System. Da sich die Qualität des Netzwerkes verbessert hat und eine erfolgreiche Übertragung schon durch die Transport- und Kommunikationsschicht des Netzwerkes geliefert wird, ist speziell eine Akzeptiert-bestätigung nicht immer erforderlich. Aus diesem Grund kann der Sender nun in seinem

Nachrichtenkopf definieren, ob er eine Akzeptiert- und/oder eine Anwendungsbestätigung erwartet und unter welchen Bedingungen diese übertragen werden soll (immer, niemals, im Fall von Fehlern, nach erfolgreicher Übertragung).

Die von der Tandem verschickten und auf dem Datenbankrechner des GTDS von einem speziellen Serverprogramm angenommenen Nachrichten sind vom Typ ADT und Finanzwesen. Der ADT Transaktionssatz sorgt für die Übermittlung neuer oder aktualisierter demographischer und Besuchsinformationen von Patienten. Die empfangenen Nachrichten haben die Ereigniscodes ADT^A01, ADT^A03, ADT^A08, ADT^A18 und ADT^A31. Der Transaktionssatz Finanzwesen beschreibt dagegen Patientenabrechnungstransaktionen. Hierbei hat die empfangene Nachricht den Ereigniscode BAR^P01.
Die Nachrichten werden zunächst unverändert in eine Datei geschrieben. Durch eine Empfangsbestätigung wird dem Sender jeweils signalisiert, daß die Nachricht empfangen wurde. Danach werden die Nachrichten durch ein für diese Situation entwickeltes Umformatierungsprogramm in das richtige Format gebracht. Das sich anschließende spezielle Ladeprogramm schreibt die angelieferten Datensätze in die richtigen Felder der dafür vorgesehenen Tabellen.
Die Abbildung 26 zeigt den Typ der erhaltenen Nachrichten sowie deren Segmente an. Die Datensätze der mit -> gekennzeichneten Segmente werden in die dahinter genannten Tabellen übernommen. Dabei werden die Inhalte der ADT-Nachrichten in die Tabellen EXTERNER_PATIENT und ABTEILUNG_PATIENT_BEZIEHUNG, die Inhalte der BAR^P01 Nachricht in die Tabellen EXTERNE_DIAGNOSE und EXTERNE_ PROZEDUR geladen.

Diese Daten können in einer bestimmten Maske des Dokumentationssystems (Maske *Stationsliste)* angesehen werden. Es handelt sich um alle aktuell in der Abteilung stationär oder ambulant befindlichen Patienten.
Da hierbei jedoch auch Nicht-Tumorpatienten zur Ansicht gelangen, müssen die Tumorpatienten beim ersten Mal durch Auswahl von ->*GTDS* ins GTDS übernommen werden.

Grundsätzlich können auch Anfragen (Queries) einen Datentransfer auslösen. In der Maske *Übersicht über alle Patienten der Klinik* (siehe Abschnitt 4.3) wurde ein solcher „solicited update“ realisiert . Der behandelnde Arzt gibt eine Patienten_Id in ein dafür vorgesehenes Feld ein. Über *Suchen* wird eine Anfrage über eine HL7 Query Nachricht (A19) an die

Patientenverwaltung geschickt und da der Patient dem Verwaltungssystem bereits bekannt ist, werden die Patientenstammdaten an den GTDS Datenbankserver gesendet.

Aufnahme eines Patienten (Admit a patient) Ereigniscode A01

ADT	ADT Nachricht
MSH	Message Header
EVN	Event Type
PID	Patient Identification -> Tabelle EXTERNER_PATIENT
NK1	Next of Kin
PV1	Patient Visit -> Tabelle ABTEILUNG_PATIENT_BEZIEHUNG
IN1	Insurance Information -> Tabelle EXTERNER_PATIENT
IN2	Insurance Information - Addit. Info

Entlassung eines Patienten (Discharge a patient) Ereigniscode A03

ADT	ADT Nachricht
MSH	Message Header
EVN	Event Type
PID	Patient Identification -> Tabelle EXTERNER_PATIENT
PV1	Patient Visit -> Tabelle ABTEILUNG_PATIENT_BEZIEHUNG
PV2	Patient Visit - Addit. Info

Die Nachrichten mit den Ereigniscodes ADT^A08 und ADT^A31 Aktualisierung der Patientendaten (Update patient information) haben die gleichen Segmente wie die Nachricht mit dem Ereigniscode ADT^A01.

Abgleich der Patientendaten (Merge patient information) Ereigniscode A18

ADT	ADT Nachricht
MSH	Message Header
EVN	Event Type
PID	Patient Identification -> Tabelle EXTERNER_PATIENT
MRG	Merge Information

Hinzufügen und Ändern von Abrechnungsinformationen
(Add and Update patient accounts) Ereigniscode P01

BAR	Add/Change Billing Account
MSH	Message Header
EVN	Event Type
PID	Patient Identification -> Tabelle EXTERNER_PATIENT
DG1	Diagnosis Information -> Tabelle EXTERNE_DIAGNOSE
PR1	Procedures -> Tabelle EXTERNE_PROZEDUR

Abbildung 26: Typ der empfangenen Nachrichten sowie deren Segmente und die für die Datensätze vorgesehenen Tabellen

Eine HL7 Nachricht besteht aus einer Gruppe von Segmenten in einer definierten Sequenz. Jede Nachricht hat einen Nachrichtentyp, einen dreistelliger Buchstabencode, der ihren Zweck bestimmt. Die empfangenen Nachrichten sind vom Typ ADT (Aufnahme, Entlassung, Verlegung) und Finanzwesen. Sie haben die Ereigniscodes ADT^A01, ADT^A03, ADT^A08, ADT^A18, ADT^A31 und BAR^P01. Die Inhalte der ADT-Nachrichten werden in die Tabellen EXTERNER_PATIENT und ABTEILUNG_PATIENT_ BEZIEHUNG, die Inhalte der BAR^P01 Nachricht in die Tabellen EXTERNE_DIAGNOSE und EXTERNE_PROZEDUR geladen.

Entsprechend der Nachrichtenstruktur ist die HL7-Schichtenarchitektur in drei Protokollebenen unterteilt.
Die oberste Schicht (Abstract Messages) spezifiziert, bezogen auf die definierten Triggerereignisse, den jeweiligen Nachrichtenaufbau. Jede Nachricht ist in einer speziellen Notation, die die Segment IDs in der Reihenfolge auflistet, in der sie in der Nachricht erscheinen würden, definiert.
Weitere Sachverhalte innerhalb der Anwendungsschicht beinhalten den Zeitpunkt des Austausches, die gültigen Antwortnachrichten und die Behandlung von bestimmten Fehlern der Anwendung (application level errors) oder das Versagen des darunterliegenden Kommunikationssystems. Die siebte Ebene unterstützt Funktionen wie Sicherheitschecks, Identifikation der Teilnehmer, Verfügbarkeitchecks und Verhandlungsaustauschmechanismen.
Die darunterliegende Ebene beschreibt die genaue Darstellung einer abstrakten Nachricht anhand der HL7 Verschlüssselungsregeln (Encoding Rules). Diese Ebene korrespondiert am nächsten mit den Schichten 5 und 6. Sie setzt auf einer Kommunikationsumgebung ohne Presentation Layer (Darstellungsschicht, Schicht 6), aber mit robustem Transport Layer (TCP/IP, DECNET, SNA) auf. Die Verschlüsselungsregeln unterstützen eine etablierte Verbindung für jede Nachricht und ihre Antwort.
Fehlt die Transportschicht (beispielsweise bei einer RS-232-Terminalverbindung), so deckt ein HL7 Lower Layer Protocol die Ebene zwei bis vier des Referenzmodells ab [Jostes 1993].

Es wurde versucht, einen „maximalen Standardisierungsgrad“ zu erreichen. Dennoch wird es immer Situationen geben, in denen lokale Gegebenheiten nicht berücksichtigt sind. In solchen Fällen läßt HL7 sowohl auf Nachrichtenebene- als auch auf Segmentebene benutzerdefinierte Erweiterungen zu und ist somit auch offen für Transaktionstypen, die im Protokoll nicht berücksichtigt sind. Jede Institution kann nämlich für benötigte Datengruppen individuelle Z-Segmente zusammenstellen, die an dem eigenen Bedarf, d.h. für den erforderlichen Umfang der Inhalte für die Datenübertragung angepaßt sind. Diese Segmente sind an bestehende Nachrichten anzuhängen, bzw. es können unter Zuhilfenahme von neuen Triggerereignissen komplett neue Nachrichten zusammengestellt werden [Haeberlin 1999]. Sie müssen durch ein „Z“ als den ersten Buchstaben des Segmentidentifikationstags oder der Nachricht markiert werden.
Auf der anderen Seite sollte diese Möglichkeit vorsichtig verwendet werden, ansonsten werden die Vorteile eines Kommunikationsstandards verloren gehen, wenn jede Anwendung ihre eigenen Z-Segmente einführt. Die deutsche HL7 Benutzergruppe führt ein Z-Segment

und Z-Nachricht Register, um notwendige Erweiterungen des Standards zu liefern und um ein unkontrolliertes Wachsen von zusätzlichen Segmenten und Nachrichtendefinitionen zu vermeiden.

Es kann also eine für spezielle Anwendungen erforderliche Übertragung von Daten, die in der aktuellen Version noch nicht spezifiziert sind, dennoch erfolgen. HL7 ist ausdrücklich als ein sich über die Zeit verändernder Standard gekennzeichnet und wird entsprechend dokumentiert [Steimke 1995].

6. XML

6.1 Problematik

Ein generelles Problem bei webbasierten Anwendungen sind die wechselnden Anforderungen an die Eingabemasken und damit auch an die Berichterstellung.

Ein weiteres allgemeines Problem besteht durch die vielen unterschiedlichen Datenformate, die den Austausch von Informationen zwischen verschiedenen Computerprogrammen- und plattformen erschweren. Hinzu kommt, daß auch Schnittstellendefinitionen wie HL7 einige Defizite ausweisen. Denn HL7 beruht auf einem gewachsenen, teilweise inkonsistenten Ereignismodell. Problematisch sind die oftmals unscharfen Ereignis- und Felddefinitionen selbst, die dem Betreiber für die Berücksichtigung individueller Bedürfnisse einen teilweise erheblichen Interpretationsspielraum lassen. Aus diesem Grund besteht eine Vielzahl ähnlicher, aber mehr oder weniger subtil unterschiedlicher Schnittstellendefinitionen [Köhler 1998]. Somit ist HL7 weit davon entfernt, ein „Plug and Play“ Standard zu sein. Jedes Krankenhaus muß genau feststellen, auf welche der festgelegten Ereignisse die vorhandenen Anwendungen überhaupt reagieren können, welches System die Kommunikation auslösen soll, und wie mit optionalen Inhalten von Nachrichten (Kannfelder, optionale Segmente) zu verfahren ist, ob sie manchmal, immer oder nie übertragen werden [Steimke 1995]. Darüberhinaus ist HL7 nicht dazu entwickelt worden, Informationen mit Webbrowsern zu präsentieren. Eine typische HL7-Nachricht kann also nicht mit einem einfachen Webbrowser gelesen werden. Dafür ist die Nutzung einer speziellen HL7-Anwendung erforderlich.

6.2 Einführung XML

In den letzten zwei bis drei Jahren hat sich die erweiterbare Markup-Sprache XML (Extensible Markup Language) entwickelt. Das Format von XML ist textbasiert, was Einfachheit und Austauschbarkeit mit den Daten maximiert, da nur Text hard- und softwareunabhängig von jeder Computerplattform dargestellt werden kann [Hunter 2000]. Die Vision von XML für die Zukunft ist damit auch, die Sprache für jede Art von Daten zu etablieren, nicht nur im Bereich der Strukturierung von Dokumenten, sondern auch bei Austauschformaten für Standardnachrichten. Die heute anfallende Doppelarbeit bei Erfassung und Konvertierung fällt damit weg [Pott 1999].

XML ist eine Teilmenge von SGML, dem bereits 1985 definierten ISO8879 Standard für die Strukturierung von Dokumenten. Das Konzept wurde auf der SGML EUROPE 96 in München entwickelt. Seit März 1998 liegt die Version 1.0 vor [Dudeck 2000].

Ziel der Entwicklung von XML war es einerseits, das durch viele Ausnahmeregelungen sehr flexible SGML sowohl einfacher zu gestalten als auch "webtauglich" zu machen, andererseits aber mit SGML kompatibel zu bleiben, um die Vielfalt der verfügbaren SGML Tools auch für XML nutzen zu können. Im Gegensatz zur sehr schwerfälligen und umfangreichen SGML-Sprache mit einer Standardbeschreibung von über 500 Seiten, ist XML kompakt und übersichtlich (knapp 33 Seiten).

SGML und XML sind Metasprachen. Mit ihrer Hilfe lassen sich eigene neue Sprachen zur Dokumentbeschreibung erstellen. Man kann eigene Markup-Befehle und Attribute nach Bedarf definieren. Dokumentstrukturen können in ihrer Komplexität an die erforderlichen Informationen angepaßt werden. Beispielsweise läßt sich mit XML die Sprache HTML definieren, die klassische Sprache zur Formatierung von Internetdokumenten. Diese umfaßt einen festen Stamm von definierten Befehlen ("Tags") und ist nicht erweiterbar [Pott 1999].

Ein XML Dokument besteht aus [Hranitzky 2000]:

- einem Prolog mit Startanweisung
  Der Prolog sagt, welche XML-Version vorliegt und optional welcher Zeichensatz verwendet wird: <?xml version="1.0" encoding="ISO-8859-1" ?>
- einem Verweis auf die externe sog. DTD- und/oder einer internen DTD-Beschreibung (optional)
- einem Wurzelelement
- hierarchisch strukturierten Elementen (mit optionalen Attributen)
- sog. Entitäten, die ersetzbare Einheiten darstellen
  Mit ihrer Unterstützung werden z.B. Sonderzeichen mit Hilfe von Abkürzungen definiert. Zusätzlich lassen sich Kürzel zur Verfügung stellen, die automatisch durch längere oder häufig eingesetzte Zeichenfolgen ersetzt werden.
- Verarbeitungsanweisungen (optional)
- sog. CDATA(Character Data)-Abschnitten (optional), weisen den Parser an, diesen Text ohne weitere Analysen an die Anwendung weiterzugeben
- Kommentaren (optional) <!-- Hier könnte ein längerer Kommentar eingefügt werden. -->

In der Terminologie der Auszeichnungssprachen wird also eine in XML formulierte Beschreibung als XML-Dokument bezeichnet, auch wenn der Inhalt nichts mit Textverarbeitung zu tun hat.

XML gruppiert Information in Hierarchien, einer sog. Baumstruktur, d.h. die Elemente stehen in einer Eltern/Kind- und Geschwister/Geschwister-Beziehung zueinander. Alle Elemente außer dem Wurzelelement haben einen "Vater". Ein Element enthält eine Anfangsmarke, eine Endmarke und einen zwischen den Marken geklammerter Inhalt. Der Inhalt kann aus weiteren Elementen und aus reinem Text bestehen. Die Elemente können auch Attribute haben, die in der Anfangsmarke definiert werden [Hranitzky 2000]. Da diese Elemente einen Inhalt (engl. "content") haben nennt man sie auch "Container".

Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE adresse SYSTEM "adresse.dtd"> <!-- extern -->
<adresse>
  <nachname>Bergmann</nachname>
  <vorname>Hannes</vorname>
  <strasse>Birkenweg 108</strasse>
  <plz>36243</plz>
  <ort>Gießen</ort>
</adresse>
```

XML setzt konsequent auf die Benutzung der sogenannten semantischen Tags. In HTML ist es üblich, physische Auszeichnungen wie <B> für „bold“ (= „fett“) oder <TITLE> (für den Titel) einzusetzen. Diese Tags machen durch ihren Namen deutlich, daß sie Formatierungshinweise oder Strukturelemente kennzeichnen. Semantische Tags wie <adresse> enthalten Bezeichnungen, die etwas über den Inhalt des Dokuments aussagen. Sie dienen weniger der äußerlichen Darstellung als vielmehr der inhaltlichen Dokumentstruktur, also der späteren Auswertung der Daten nach semantischen Gesichtspunkten [Pott 1999].

Einige weitere wesentliche Unterschiede zu HTML sind [Hranitzky 2000]:

a) Elemente dürfen nicht überlappen:

```
<p>Das ist <b>falsch</p><b>
```

b) Jedes Element muß mit einer Endmarke abgeschlossen sein.

c) Leere Elemente können in Kurzform benutzt werden:

```
<IMG SRC="picture.gif"></IMG> oder <IMG SRC="picture.gif"/>
```

d) Die Groß- und Kleinschreibung der Element- und Attributnamen muß beachtet werden.
e) Attributwerte müssen in Anführungszeichen gesetzt werden:
<adresse privat="ja"> oder <adresse privat='ja'>

Die sog. Document Type Definition (DTD) beschreibt eine bestimmte Klasse von Dokumenten (Elemente und Attribute und die Regeln, in welcher Reihenfolge und wie oft die Elemente im Dokument vorkommen dürfen). Die DTD-Beschreibung kann vollständig innerhalb oder außerhalb des Dokuments oder teils im Dokument und teils extern stehen [Hranitzky 2000].

Die DTD für das Adressbeispiel:

```
<!DOCTYPE adresse [
<!ELEMENT adresse (nachname, vorname+, strasse, plz, ort)>
    <!ELEMENT nachname (#PCDATA)>
    <!ELEMENT vorname (#PCDATA)>
    <!ELEMENT strasse (#PCDATA)>
    <!ELEMENT plz (#PCDATA)>
    <!ELEMENT ort (#PCDATA)> ] >
```

Mit Hilfe des Schlüsselwortes #PCDATA (Parsed Character Data), also vom Parser analysierte und weiterzuverarbeitende Daten, wird angegeben, daß der Inhalt des Elementes aus beliebigen Zeichenfolgen bestehen kann. Ein Pluszeichen, Sternchen oder Fragezeichen hinter einer Elementbezeichnung kennzeichnet die Kardinalität, d.h. ob beliebig viele Elemente größer Null, beliebig viele Elemente, also auch Null, oder optionale Elemente eingefügt werden dürfen [Pott 1999].

Einen Parser, der die DTD-Beschreibung auswertet, um die Korrektheit eines XML-Dokumentes zu prüfen, nennt man *validierenden* Parser und das geprüfte Dokument bezeichnet man als *gültig*. Wenn der Parser das Dokument nur auf Einhaltung der allgemeinen XML-Syntax prüft, ohne die Grammatik aus der DTD zu berücksichtigen (*nichtvalidierender* Parser), ist das Dokument nur *wohlgeformt* [Hranitzky 2000].

Das Konzept der DTD stammt aus der dokumentorientierten Welt SGML. XML wird aber nicht nur als Auszeichnungssprache für Dokumente benutzt, sondern allgemein für Datenaustausch. DTD arbeitet nur mit einem Datentyp - dem Textstring. Es können zwar gewisse Beschränkungen auf einige Strings angewendet werden (z.B. ID, NMTOKEN und

related Attributtypen), aber diese sind sehr schwache und begrenzte Kontrollen. Es werden keine Vorkehrungen für numerische Datentypen wie Datum, Zeit oder URLs getroffen.
Hier soll die neue Sprache XSchema helfen. XSchema kann alles was die DTD kann, kennt darüberhinaus die gängigen Datentypen (string, real, boolean,...) und ermöglicht die Definition eigener Typen. Ein weiterer Vorteil der XSchema-Sprache ist, daß sie keine eigene Syntax wie die DTD benutzt, sondern selbst in XML formuliert wird, so daß kein spezieller Parser wie für DTD erforderlich ist [Hranitzky 2000]. DTDs verwenden keine wohlgeformte XML-Syntax. Die meisten DTDs werden in der Extended Backus Naur Form (EBNF), einer formalen Sprache für Syntaxbeschreibungen, dargestellt. Weitere Gründe XSchema der DTD vorzuziehen sind u.a. die Tatsache, daß DTDs nicht gut mit Namensräumen arbeiten und keine DOM (Document Object Model) Unterstützung zur Verfügung stellen (siehe Abbildung 27). Die Standardisierung der XSchema-Sprache ist noch nicht abgeschlossen.

XML ist aber mehr als nur eine Metasprache: XML ist inzwischen ein Gattungsbegriff für eine ganze Reihe von Technologien [Hranitzky 2000].

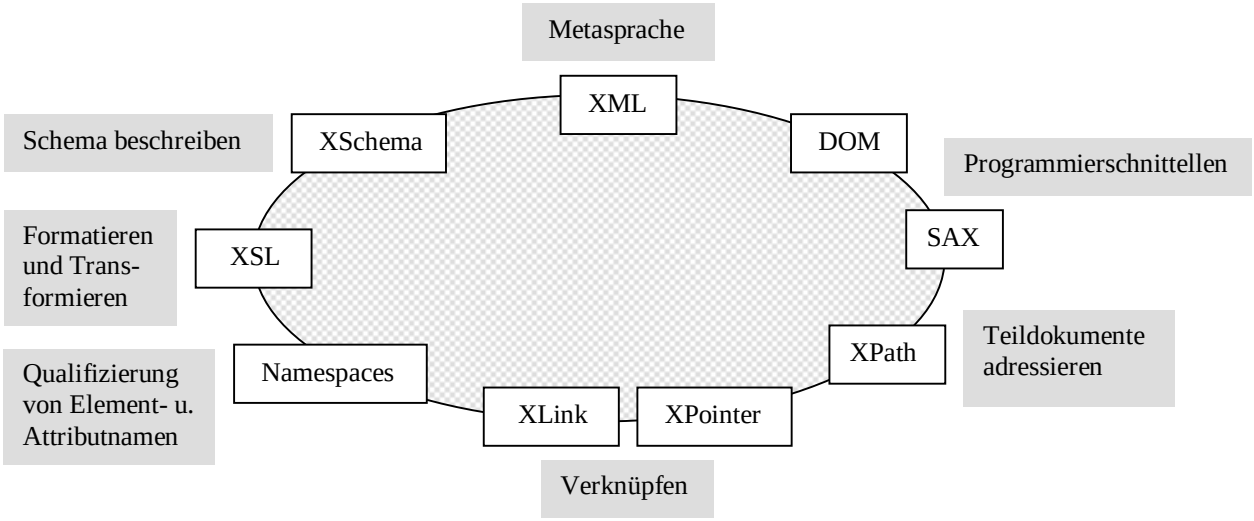

Abbildung 27: Die mit XML verwandten Technologien
XML ist eine Metasprache für die Definition eigener Auszeichnungssprachen. Zu den Mitgliedern der XML-Familie gehören u.a. XSchema, das zur Beschreibung der Struktur und zum Beschränken des Inhalts von XML-Dokumenten sowie dem Zuordnen von Datentypen zu XML-Elementtypen und -attributen dient, die Stylesheet-sprache XSL (Extensible Stylesheet Language), mit der XML-Dokumente formatiert und konvertiert werden können, die Programmierschnittstellen DOM (Document Object Model) und SAX (Simple API for XML), die Verknüpfungen XLink und XPointer, XPath, eine Sprache zum Adressieren von Teilen eines XML-Dokuments, sowie Namespaces. Namespaces sollen Dokumente in die Lage versetzen, in fremden DTDs angegebene Namen zu verwenden. Eine Namespace-Deklaration innerhalb eines XML-Dokuments verweist über einen URI auf einen Namespace "ns". Daher sind die Namen in diesem Namespace in der Form "ns:name" in einem bestimmten Teil des Dokumentenbaums verfügbar.

6.3 Ausgabe

Mit Hilfe der Stylesheetsprache Extensible Stylesheet Language (XSL) für XML, die aus zwei Sprachen, einer Transformationssprache (XSLT) und einer Formatierungssprache "XSL-Formatting Objects" (FO) besteht, können dann die XML-Dokumente formatiert und konvertiert werden. Hiermit können zum Beispiel auch HTML-Formulare zur Eingabe von Werten konstruiert werden.
Grundsätzlich stellt die Formatierungssprache Möglichkeiten zur Veränderung von Elementen der folgenden Art zur Verfügung [Pott 1999]:

- Schriftarten / Schriftgröße
- Farben
- Vertikale und horizontale Ausrichtungen
- Gliederung (Überschriften)
- Tabellen
- Behandlung von Leerräumen (Leerzeichen- und zeilen)

Eine Konvertierung mit der Transformationssprache erfolgt heute in erster Linie in Richtung HTML, da HTML-Browser die weiteste Verbreitung haben. In der Entwicklung befinden sich auch Möglichkeiten zur Umwandlung von XML-Dokumenten in RTF- oder PDF-Formate.

Die konsequente Durchsetzung der Grundintention, die bei SGML und XML verfolgt wurde, bedeutet eine klare Trennung von semantischer und physischer Auszeichnung in zwei Dokumenten. XML als klare Auszeichnung der semantischen Struktur und XSL als Umsetzung dieser Struktur in eine visuelle Ausgabe.

Für die Präsentation in einem Browser kann auch die aus der HTML-Umgebung bekannte CSS-Sprache (Cascading Stylesheet) verwendet werden. CSS bietet aber nicht die Möglichkeit, die Struktur des Dokumentes zu verändern, wie es XSLT kann.

Mit Hilfe eines einzigen XML-Dokuments, auf das verschiedene XSL-Stylesheets angewendet wurden, können völlig verschiedene HTML-Ausgabedokumente erzeugt werden. So können z.B. Daten, die mit einer Eingabemaske in Formularstruktur erfaßt wurden, in Form eines Berichts mit zusätzlichem Text erscheinen. Das heißt, die gleichen Daten können ohne großen Aufwand und ohne Änderung an der semantischen Struktur verschiedenartig dargestellt werden.

Dies veranschaulicht eine der großen Stärken von XML in Verbindung mit XSL. Für verschiedene Sichtweisen auf die gleichen Daten ist keine Änderung der Daten selbst mehr notwendig. Man wendet einfach ein anderes Stylesheet auf das Dokument an. Die Trennung von Semantik und Visualisierung wird hier sehr deutlich.

XSL-Anweisungen werden in einem (externen) Dokument (mit der Endung "xsl") angelegt. Diese Datei enthält die Übersetzungsregeln zu den in dem XML-Dokument benutzten Strukturen. Die einzelne Stilanweisung oder auch Konstruktionsregel gliedert sich in zwei zusammenhängende Teile:

- Muster (pattern)

  Durch das Muster wird festgelegt, auf welchen (selbstdefinierten) XML-Befehl sich die dann folgende Stilanweisung (Action) bezieht. Das Muster ist das Auswahlkriterium, wann die definierte Ausgabeform auf den Inhalt eines Markups umzusetzen ist.

- Aktion (action)

  Wurde das angegebene Muster im XML-Dokument erkannt, folgt die Umwandlung des betreffenden Elements in die angegebene Ausgabeform. Eine „action“-Anweisung kann neben passiven Elementen zur reinen Stilgestaltung auch dynamische Anweisungen, beispielsweise den Aufruf eines Java-Scripts oder anderer Scriptsprachen, enthalten.

Der Inhalt der XSL-Datei wird mit dem Markup <xsl> umschlossen. Der Inhalt besteht dann aus den folgenden Regeln, die dann mit <rule> markiert werden. So entsteht das Grundgerüst einer XSL-Datei:

```
<xsl>
<rule>
      <!--Konstruktionsregel 1-->
      <Muster>
      <Action>
</rule>
 <rule>
      <!--Konstruktionsregel 2-->
      <Muster>
      <Action>
</rule>
<rule>
      ....
</rule>
</xsl>
```

Eine vollständige XSL-Datei, die das vorher definierte Markup <nachname> mit der Schriftfarbe "Blau" und dem Font-Style "Fett" versieht, zeigt das folgende Beispiel:

```
<xsl>
 <rule>
      <!--Konstruktionsregel -->
      <target-element type="nachname"/>
      <H1 COLOR="blue" font-style="bold">
            <children/>
      </H1>
 </rule>
```

Wichtig im Beispiel ist, daß die Stilinformationen mit einem HTML-Element transportiert werden müssen (im dargestellten Fall ist es das Markup </H1>). An dieses Element werden die visuellen Informationen weitergegeben. Bei der anschließenden Konvertierung des Dokumentes in HTML gehen die semantischen Informationen verloren und werden durch physische Auszeichnungen ersetzt [Pott 1999].

6.4 Einsatz von XML für die Weiterentwicklung von GTDS

XML bietet gegenüber der mit dem Developer/2000 durchgeführten Vorgehensweise einige Vorteile.

- Zum einen benötigt man keine ORACLE spezifischen Werkzeuge, wie Forms- und Reports Designer für die Entwicklung der Eingabemasken und Berichte, sondern man kann freiverfügbare Tools verwenden.

  Im Grunde reicht für die Entwicklung von gültigen XML-Dateien, wie bei HTML und jeder anderen textorientierten Markup-Sprache, bereits ein einfacher Texteditor aus, mit dem man Dateien der Extension .xml erstellt. Aber gerade bei der hohen Komplexität von XML gibt es für die Programmierung weitere Hilfsmittel, spezielle SGML- und XML-Editoren, die die Erstellung und Verwaltung von XML-Dokumenten oder den zugehörigen DTDs bzw. Schematas erleichtern. Aufwendige Einarbeitungszeiten in zwei verschiedene Designer (die oben erwähnten Forms- und Reports Designer) entfallen damit. Für die Entwicklung von elektronischen Formularen und Berichten muß man sich nur mit einer, der XML-Technologie, auseinandersetzen.
- Ein weiterer Vorteil von XML ist, daß es schon webfähig ist, es müssen also für die Web-Formularerstellung kein Forms Client (Java Applet) und Forms Server bzw. für die Web-Berichterstellung kein Reportsserver in Verbindung mit der Reports Web Cartridge oder dem Web CGI installiert werden. Da kein Java Applet mehr benötigt wird, entfallen somit

auch die damit verbundenen Ladezeiten für das Herunterladen vom Anwendungsserver zum Webbrowser des Endbenutzers.

Allerdings war die Entwicklung von XML zum Zeitpunkt der Entscheidungsfindung über die zu verwendenden Tools bei die Realisierung der GTDS-Erweiterung bei weitem noch nicht abgeschlossen. So fehlte z.B. die Schemadefinition, auch waren geeignete Tools noch nicht so verfügbar.

Auch viele der bisher existierenden Browser unterstützen XML noch sehr rudimentär. Im Webbrowser ist daher ein XSL-Prozessor, der die XML-Dokumente in HTML umwandelt, erforderlich. Die neue Browsergeneration 5.0 intergriert XML im wesentlich stärkeren Maße.

Auch das XML Format kann durch Verschlüsselung und Authentisierung ergänzt werden, so daß es so sicher wie HTML ist. Zunächst kann XML auf dem Server vor der Übertragung zum Client verschlüsselt und dann auf dem Client entschlüsselt werden. Zusätzlich kann XML durch digitale Signaturen authentisiert werden. Ein solches Element wird aus dem Inhalt der XML-Datei berechnet und beinhaltet innerhalb des Dokumentes die digitalen Unterschrifteninformationen. Auf diese Art kann der Urheber eines XML-Dokumentes - oder eines Teiles davon - die Echtheit und Vollständigkeit dieses Datensatzes garantieren. Der Empfänger kann wiederum die Integrität eines Datensatzes feststellen.

6.5 XML als Austauschformat für HL7-Nachrichten

Im Gegensatz zu HL7 ist es also mit XML möglich, strukturierte Information mit Hilfe der bestehenden Internet- bzw. Intranet-Technologie zu präsentieren und mit Hilfe von XML-Anwendungen weiter zu bearbeiten und zu speichern.
Die einzigen anwendbaren Alternativen, HTML und SGML, sind nicht praktisch genug für diesen Zweck. HTML kommt, wie bereits erwähnt, mit einer ganzen Reihe von festen Befehlen und deren Syntax, die keine willkürliche Struktur zulassen. SGML läßt eine willkürliche Struktur zu, aber die Implementierung in einem Webbrowser ist damit sehr schwierig [Messaritakis 1999].

An mehreren Stellen [Biron 1999, Dolin 1999, Messaritakis 1999] gelang es, die Einschränkungen und Anforderungen des HL7 Standards mit Hilfe von XML auszudrücken.

XML kann als Nachrichtenspezifikation für HL7 Nachrichten der Versionen 2.3 und 3.0 dienen. Die Version 3.0 ist ein restriktiverer Standard als V2.3 und mit diesem auch nicht direkt kompatibel. Der Algorithmus für die Konvertierung einer HL7 Nachricht der Version 3.0 nach XML ist deshalb auch ein anderer als der für Version 2.3.
Allerdings gibt es viele Möglichkeiten, um HL7 in eine XML Syntax zu übertragen. Die optimale Methode wird teilweise durch technische Übertragungsanforderungen und teilweise durch praktische Implementierungsfragen bestimmt sein. Einige dieser Fragen sind mehr quantitativ ausgedrückt (z.B. Nachrichtenlänge), während andere mehr qualitativ sind und weitere Analysen erfordern (Geschwindigkeit der Nachrichtenerstellung) [Dolin 1999].

XML stellt zwar die Syntax zur Verfügung, den Inhalt von Dokumenten über zusätzliche Markups zu spezifizieren, doch die Interpretation dieser Markups kann nicht automatisch erfolgen, sondern nur im Rahmen von Programmen, die direkt dafür geschrieben wurden. XML ermöglicht die Spezifikation einer Art Schema für Hypertextdokumente, vergleichbar mit einem Datenbankschema. Damit eine solche Spezifikation aber für die Integration autonomer Systeme von Nutzen sein kann, muß selbstverständlich jede Komponente, die ein entsprechendes Dokument verarbeitet, dafür auch Vorkehrungen getroffen haben. Um das Problem der semantischen Heterogenität autonomer Komponenten zu lösen, ist es also erforderlich, entsprechende DTDs bzw. Schemata zu standardisieren [Lenz 1998].

Gerade eine ganz neue Technologie wie XML bringt es mit sich, daß unterstützende Software zunächst noch Mangelware ist. Trotzdem ist heute schon der Trend der breiten Unterstützung und das Bekenntnis zu XML, vor allem der Industrie zu spüren. Insbesondere hat sich XML in den letzten Monaten als „Standardsprache“ für elektronische Marktplätze im Bereich Business-to-Business (B2B), dem zur Zeit am schnellsten wachsenden Markt für Unternehmensanwendungen, etabliert, und wird von den Marktführern wie z.B. Ariba, CommerceOne, i2 Technologies, IBM, Oracle und SAP unterstützt, so daß mit einer sehr weiten Verbreitung von XML in den nächsten Jahren zu rechnen ist.

7. Ergebnisse und Diskussion

Im Verlauf dieser Arbeit wurde eine Erweiterung des GTDS um urologische Untersuchungsmasken entwickelt und in das vorhandene Klinikinformationssystem integriert.

Dafür wurden zunächst bestehende Internettechnologien für die Realisierung berücksichtigt, da das Internet, im Gegensatz zu nur lokal installierten Programmen, eine hohe Verfügbarkeit und leichte Zugangsmöglichkeit bietet. Es wurden die Konzepte und Techniken webbasierter Datenbank-Anbindungsarchitekturen, also Architekturen, die den Zugriff auf eine Datenbank über das Internet- oder Intranet ermöglichen, bezüglich ihrer jeweiligen Stärken und Schwächen untersucht sowie anhand eines Forderungskatalogs bewertet. Dabei waren vor allem

- eine hohe Produktivität bei der Anpassung gefordert, damit effiziente Adaptationen der Applikationen und damit eine rasche Anpassung an neue Erkenntnisse aus der Forschung und Erfahrungen aus der Praxis möglich sind, sowie
- eine hohe Portabilität verlangt, es also keine oder nur eine geringe Abhängigkeit vom Browser und vom Betriebssystem bestehen sollte. Weiterhin sollte
- die volle Funktionalität einer Programmiersprache zur Verfügung stehen und
- das System eine gute Performance aufweisen, damit keine langen Wartezeiten beim Laden der Anwendung entstehen bzw. das Navigieren innerhalb der Applikation ohne zeitliche Verzögerung durchgeführt werden kann. Das System sollte außerdem
- wegen der sensiblen Daten, die in einem Klinikum anfallen, hohen Sicherheitskriterien genügen.

Aus der Gegenüberstellung der Evaluationsergebnisse server- und clientseitiger Anwendungen, wurde ersichtlich, daß JavaApplets für die Realisierung einer GTDS-Erweiterung vor allem wegen ihrer Plattformunabhängigkeit und Sicherheit, zumindest zum Zeitpunkt der Entscheidungsfindung, die beste Wahl darstellte. Der größte Nachteil einer Entwicklung mit Java war jedoch der zu erwartende beträchtliche Aufwand für die Programmerstellung, der eine flexible Gestaltung der Applikation zunächst schwierig erscheinen ließ. Eine geeignete Lösung stellte daher der Developer/2000 von Oracle dar, der Werkzeuge zur Erstellung von Masken, Berichten, Grafiken, Abfragen, Datenbankobjekten und Prozeduren beinhaltet. Er läuft auf allen gängigen Plattformen (Windows, Macintosh, UNIX) und bot sich für diese Aufgabe vor allem durch seine Produktivität bei der Client/Server- und Webentwicklung durch RAD (Rapid Application Development) -

Techniken, Objektorientierung und eine vereinheitlichte Client-, Applikationsserver- und Datenbankserverarchitektur an. Denn einmal erstellte Developer-Anwendungen können sowohl im Client/Server-Betrieb, als auch im Web eingesetzt werden – ohne Änderungen am Programmcode. Das Java System im Browser des Anwenders lädt dabei das Developer/2000 JavaApplet. Dieses verbindet sich mit dem Web Anwendungsserver, auf dem die Developer/2000 Anwendung läuft. Die Formulare und Berichte können dann in einem separaten Appletfenster betrachtet werden. Der einzige Unterschied zu einer lokalen Installation ist, daß kein ORACLE, Developer/2000 oder Anwendungscode auf dem PC, auf dem der Webbrowser läuft, benötigt wird. Die Anwendung läuft über den zentralen Anwendungsserver, der wiederum auf den Datenbankserver zugreift.

Für die GTDS-Erweiterung wurden insgesamt 18 Masken, welche Datensätze zur Dokumentation von körperlichen Untersuchungen, Sonographie, Röntgen und Labor beinhalten, mit zahlreichen Untermenüs und Popup-Fenstern erstellt. Die Daten werden in einem relationalen Datenbanksystem in einer der dritten Normalform genügenden Datenstruktur gespeichert. Im ganzen gibt es zusätzlich fast 50 Tabellen mit ungefähr 1000 Spalten. Da bundesweit kein anerkannter Standard für die Dokumentation von urologischen Untersuchungen besteht, konnten nicht einfach schon bereits existierende, bedruckte Dokumentationsformulare als Vorlage für die Entwicklung der elektronischen Formulare verwendet werden. Die Realisierung spezieller abteilungsspezifischer Wünsche und klinischer Anforderungen wurde daher in enger Zusammenarbeit mit den späteren Anwendern in einer Art Trial and Error Methode erarbeitet. In einer iterativen Verfahrensweise wurde die Struktur der Masken, der inhaltliche Aufbau und die notwendige Anzahl der Items festgelegt. Dabei waren mehrere Durchläufe der Maskenerstellung erforderlich.
Die erfaßten Daten können sowohl als tabellarischer Bericht zur Übersicht und Kontrolle auf dem Bildschirm angezeigt, als auch auf Papier ausgedruckt werden. Dabei besteht die Möglichkeit an mehreren Stellen freitextliche Anmerkungen zu ergänzen. Die Berichte können zur Kommunikation mit anderen Ärzten (z.B. Arztbriefschreibung) bzw. mit den Patienten (z.B. Einbestellung) verwendet werden. Bei der Erstellung der Berichte kann zwischen den durchgeführten Untersuchungen, bzw. zwischen den unterschiedlichen Untersuchungsterminen ausgewählt werden.

Praktische Erfahrungen konnten zunächst mit der ersten webfähigen Probeversion (Trialversion V 1.4) des jetzige Developer/2000 (V 1.6) gemacht werden. In dieser Umgebung zeigten sich jedoch einige aus der Oracledokumentation nicht zu ersehende Probleme. Zum einen lief der Serverprozeß in dieser Version nicht stabil. Als schwerwiegendstes Hindernis für eine praktische Einsetzbarkeit erwiesen sich jedoch die langen Wartezeiten für das Herunterladen des Applets vom Anwendungsserver über das Kliniknetz zum Webclient, die ohne weiteres über 10 Minuten betrugen. Auch die Navigation von einer Maske zur anderen lief nur mit erheblicher zeitlicher Verzögerung, so daß nur ein sehr geduldiger und nicht unter Zeitdruck stehender Anwender damit gearbeitet hätte. Eine Nutzung des System unter diesen Umständen wäre zu umständlich gewesen.
Allerdings sind inzwischen mit der neuen Developer/2000 Version von Oracle (zur Zeit liegt die Version 2.0 vor) die Probleme der langen Ladezeit und des verzögerten Navigierens durch die Masken gelöst. Während das Wechseln von einer Maske zur anderen ohne zeitliche Verzögerung abläuft, so als ob das GTDS lokal installiert sei, ist beim Starten des Systems über das Web mit eine kurzen Wartezeit von etwa 1 bis 2 Minuten zu rechnen, die jedoch keinerlei Einschränkung des Betriebes bedeutet.

Um zu vermeiden, daß Daten, für eine vollständige Dokumentationsdatenbasis mehrfach in verschiedene Systeme von verschiedenen Instanzen eingegeben werden müssen und um den damit verbundenen Mehraufwand zu umgehen, wurde ein automatisierter Übertrag und Abgleich von schon erfaßten Patientenstammdaten aus der Verwaltung sowie von bereits dokumentierten Diagnose- und Prozedurdaten aus einem OP-System (MedAccess) über eine HL7-Schnittstelle realisiert. Dadurch wird die Konsistenz der Daten gewährleistet. Grundlage dieser Datenübertragung ist am Klinikum der Universität Gießen dabei das seit dem Jahre 1989 bestehende Klinikinformationssystem WING.

Seit November 1999 wurde die Anwendung erstmals in der Urologischen Ambulanz eingesetzt und konnte im laufenden Betrieb nach den Erfordernissen und Wünschen der Kliniker weiterentwickelt werden.

Die Akzeptanz des Systems konnte durch abteilungsspezifische Erweiterungen, durchgehende Verfügbarkeit von Patientendaten, einheitliches Aussehen sowie einfacher Benutzerführung gesteigert werden. Eine ausgedehnte Schulung war nicht notwendig, auch sind Kenntnisse der

Arbeit mit der Datenbank nur bedingt erforderlich. Es wurde lediglich eine kurze Einführung in die Verwendung des GTDS durchgeführt.

Mit dieser Implementierung wurde eine direkte Dateneingabe durch das ärztliche Personal und die damit verbundenen Vorteile erreicht:

- Die Datenerfassung ist stärker in den Behandlungsablauf integriert.
- Unterstützung der täglichen klinischen Tätigkeit durch einfachen Zugriff auf die Daten, leichtes Überblicken der Informationen und diverse Möglichkeiten der Berichterstellung.
- Patientendaten und Krankheitsprofile können in einheitlicher und vergleichbarer Form erfaßt und somit für Forschungszwecke und Statistiken verwendet werden.
- Dateneingaben durch Dokumentare und damit zeitaufwendige Rückfragen und Fehlinterpretationen der von Ärzten handschriftlich erfaßten Datensätze entfallen.
- Insgesamt ergibt sich dadurch eine gewisse Zeitersparnis und eine verbesserte Datenqualität.

Darüber hinaus wurde während der Entwicklung des Systems deutlich, daß nicht die Technologie der das System limitierende Faktor ist. Im GTDS waren von Anfang an Funktionen und Dienste integriert, um spezifischen klinischen Anforderungen gerecht zu werden. Auch ermöglichen die eingesetzten Tools eine effiziente und korrekte Implementierung spezieller Wünsche sowie eine kontinuierliche Anpassung an neue Anforderungen und Entwicklungen.
Wesentlich für die zügige Umsetzung von klinischen Erweiterungen von Tumordokumentationssystemen ist, im voraus eine genaue Aufwandsabschätzung für die Entwicklung einzuplanen, die sowohl das klinische Personal als auch die Entwicklungsressourcen mit einschließt:

1. Seitens des ärztlichen Personals wird der zu investierende Zeitaufwand für die Datendefinition erheblich unterschätzt. Abteilungsspezifische Erweiterungen sind nicht wie z.B. die Organspezifische Dokumentation schon als Formulare oder Standards erhältlich, sondern nur als Wissen und Erfahrungen bei den zukünftigen Anwendern verfügbar. Diese zu strukturieren ist sehr zeitintensiv und bedarf mehrerer Ergänzungen und Revisionen.

2. Auch der entstehende Entwicklungsaufwand ist nicht zu unterschätzen. Zwar wurden im Verlaufe dieser Arbeit wesentliche Technologien und Verfahren entwickelt, mit deren Hilfe weitere Implementierungen effizient realisiert werden können. Dennoch muß aufgrund von Abteilungsspezifika jede weitere Implementierung maßgeschneidert entwickelt und dann auch gewartet werden. Hierzu müssen die notwendigen Ressourcen und Mittel eingeplant werden. Die Verfügbarkeit allgemein akzeptierter Standards für die Dokumentation von Untersuchungen in klinischen Abteilungen könnte die Realisierung ähnlicher Lösungen zukünftig wesentlich vereinfachen und beschleunigen.

Im wesentlichen gilt hier also, was aus anderen Bereichen der Softwareentwicklung bekannt ist: immerhin enden 84% aller Softwareprojekte nicht innerhalb des geplanten Zeitraums, Budgets, und mit der geplanten Funktionalität. Darüber hinaus werden 30 % aller Softwareprojekte abgebrochen. Der wesentliche Grund ist ein großes Maß an Ungewißheit: unklare, variierende Kundenwünsche, sich ständig ändernde Technologien und Programmentwürfe, die nicht vollständig vorhersehbar sind [Hoch 2000].

Das sich in den letzten 3-4 Jahre entwickelnde XML ist sehr gut zur Datenstrukturierung geeignet. Mit Hilfe von XML-Techniken wie XSL können sowohl Masken als auch Berichte generiert werden. Zukünftige Applikationen könnten sich dabei darauf beschränken, einen Rahmen zu bilden, innerhalb dessen z.B. abteilungsspezifische Anpassungen mit Hilfe von XML-Dokumenten erfaßt und dargestellt werden. Für die Programmierung solcher Anpassungen genügen XML-Werkzeuge, wodurch diese von der Anwendung unabhängig und leichter pflegbar sind.

XML als universelles Austauschformat wird leichter zu implementieren und insgesamt flexibler sein als die bisherigen Schnittstellendefinitionen. Mit XML ist man potentiell in der Lage, die semantischen Inhalte der zu übertragenden Daten eindeutig zu übermitteln, sowie die Beziehungen zwischen Datenelementen eindeutig darzustellen.

Ausblick

Nach über einem Jahr der praktischen Nutzung des erweiterten GTDS in der Urologischen Ambulanz hat sich herausgestellt, daß auch die automatische Übertragung und Bereitstellung der im Labor erfaßten Daten und Werte von Vorteil wäre und eine weitere Arbeitserleichterung darstellen würde. Eine Neustrukturierung der Ambulanz hat dazu geführt, daß es

statt nur einem drei Behandlungszimmer gibt. Es muß nun zunächst die geeignete Hardware zur Verfügung gestellt werden, um auch hier den Zugriff auf das erweiterte GTDS zu ermöglichen. Weitere Mitarbeiter sollen in die Tumordokumentation eingeführt werden, damit auch diese Daten mit dem System erfassen können.
Auch ist in Zukunft eine Onkologische Tagesklinik mit der Einführung von GTDS als interdisziplinäres Tumordokumentationssystem geplant.

8. Zusammenfassung

Aufgabe dieser Arbeit war es, mit Hilfe der Internettechnologie das Gießener Tumordokumentationssystem (GTDS) in die Urologische Klinik der Universitätsklinik Gießen als Teil des Gießener Klinikinformationssystems WING zu integrieren. Hierzu wurde das GTDS um Untersuchungsmasken erweitert, die auf die individuellen, benutzergruppenspezifischen Bedürfnisse einer urologischen Abteilung ausgerichtet sind. Das System ist durch folgende Punkte gekennzeichnet:

- die GTDS-Erweiterung umfaßt insgesamt 18 zusätzliche Masken, welche Datensätze zur Dokumentation von Körperlichen Untersuchungen, Sonographie, Röntgen und Labor beinhalten. Die Daten werden in einem relationalen Datenbanksystem in einer der dritten Normalform genügenden Datenstruktur gespeichert.
- es besteht ein nahtloser Übergang von den urologischen Untersuchungsmasken zum GTDS und in umgekehrter Richtung vom GTDS in den urologischen Teil der Dokumentation.
- die eingegebenen Daten können als tabellarischer Gesamtbericht ausgedruckt werden. Dabei besteht die Möglichkeit den Bericht an mehreren Stellen durch freitextliche Anmerkungen zu ergänzen.
- das erweiterte GTDS ist keine Stand-Alone-Lösung, sondern als integraler Bestandteil des Klinikinformationssystems WING realisiert worden. Jeweils über eine HL7-Schnittstelle werden dem System Patientenstammdaten aus der Verwaltung, sowie Diagnose- und Prozedurdaten aus einem OP-System zur Verfügung gestellt.
- die entwickelte Lösung setzt auf plattformunabhängige Tools, um eine leichte Distribution und Installation des Systems zu erreichen. Um wechselnden Anforderungen gerecht zu werden, können neue Masken hinzugefügt bzw. bestehende Masken durch neue Felder und Detailangaben ergänzt werden.
- das System weist eine gute Performance auf und genügt hohen Sicherheitsanforderungen.

Einschränkungen durch Ort, Benutzerschnittstellen, Hardwareplattformen und Anwendungsentwicklung wurden durch diesen Ansatz überwunden. Er nutzt die Vorteile von bereits existierenden Ressourcen. Einige dieser Ressourcen, wie das Internet, das WWW und verschiedene Online Dienste sind öffentlich verfügbar. Eine Ressource, der Tandem Zentralrechner, ist für das GISNET spezifisch, hat aber Spezifikationen ähnlich den Zentralrechnern, die an anderen Stellen in Verwendung oder Entwicklung sind.

Mit der Implementation einer klinischen Erweiterung wurden die Voraussetzungen geschaffen, die Integration der Tumordokumentation zu verbessern und die Akzeptanz durch die Ärzte zu erhöhen. Dabei zeigte sich, daß die technischen Schwierigkeiten nicht im Vordergrund standen. Bei zukünftigen Implementationen im klinischen Bereich ist darauf zu achten, daß der Aufwand sowohl beim ärztlichen Personal als auch bei der Entwicklung sorgfältig abgeschätzt und eingeplant wird. Die Verfügbarkeit allgemein akzeptierter Standards für die Dokumentation von Untersuchungen in klinischen Abteilungen könnte die Realisierung ähnlicher Lösungen zukünftig wesentlich vereinfachen und beschleunigen.

Zukünftig wird XML in vielen Bereichen eine wichtige Rolle spielen, da es erlaubt Datenstandards zu beschreiben und zu implementieren, somit Standards und Daten für unterschiedliche Verwendungszwecke aufzubereiten.

Anhang A: Literaturverzeichnis

[Altmann 1996 a]
Altmann, U., Katz, F.R., Tafazzoli, A., Dudeck, J.
GTDS - a Tool for Registries to Support Shared Patient Care
AMIA Annual Fall Symposium:512-516 (1996)

[Altmann 1996 b]
Altmann, U., Katz, F.R., Haeberlin, V., Dudeck, J.
Aspects of Integrating a Disease Specific System into a Hospital Information System
MIE 1996

[Altmann 1999]
Altmann, U., Katz, F.R.
Gießener Tumodokumentationssystem ein Werkzeug in der Qualitätssicherung
1999

[Benn 1998]
Benn, W., Gringer, I.
Zugriff auf Datenbanken über das World Wide Web
Informatik Spektrum 21:1-8, Springer Verlag 1998

[Berard 1993]
Berard, E.V.
Essays on Object-oriented Software Engineering
Software Engineering, Volume I, Prentice Hall, 1993

[Biron 1998]
Biron, P.
XML: Basics
Permanente Clinical Systems Development
Kaiser Permanente, Southern California

[Developer/2000 1997]
Developer/2000 Release 1.6, Deploying Applications on the Web
Oracle Corporation 1997

[Dolin 1998]
Dolin, R., Rishel, W., Biron, P.
SGML and XML as Interchange Formats for HL7 Messages
AMIA, 1998

[Dudeck 1994]
Dudeck, J., Wagner, G., Grundmann, E., Hermanek, P.
Basisdokumentation für Tumorkranke, 4. Aufl., Springer Verlag Berlin Heidelberg New York etc. 1994

[Dudeck 1999]
Dudeck, J., Wagner, G., Grundmann, E., Hermanek, P.
Basisdokumentation für Tumorkranke, 5. Aufl., Zuckschwerdt Verlag 1999

[Dudeck 2000]
Dudeck, J.
Die Entwicklung von XML
Institut für Medizinische Informatik, Gießen, 2000
Internetdokument

[Haeberlin 1999]
Haeberlin.V.
Schnittstellenkonzepte in Tumordokumentationssystemen
Inaugural-Dissertation, Justus-Liebig-Universität Gießen, 1999

[Health Level Seven 1992]
Health Level Seven Inc.: Health Level Seven
An Application Protocol for Electronic Data Exchange in Health Care Enviroments
Version 2.2, Ann Arbor, MI, 1992, 1993

[Henderson-Sellers 1990]
Henderson-Sellers, B., Edwards. J.M.
The Object-oriented Systems Life Cycle
Communications of the ACM 33:9, September 1990, S. 142-159

[Hoch 2000]
Hoch, Detlef et al.
The Secrets Of Software Success, Management Insights From 100 Software Firms Around the World
Harvard Business School Press, Boston, Mass., 2000

[Hunter 2000]
Hunter, D., Cagle, C., Gibbons, D., Ozu, N., Pinnok, J., Spencer, P.
Beginning XML
Wrox Press Ltd, Birmingham, UK, 2000

[Hranitzky 2000]
Hranitzky, Norbert
Write Once, Read Anywhere, XML -- Der Stein der Weisen?
Internetdokument

[Jostes 1993]
Jostes, C., Paczkowski, J., Schröder, J.
HL7 – Schnittstelle für Klinikanwendungen, Ganzheitliche Medizin
iX 7/1993 S.92-96

[Köhler 1998]
Köhler, S., Neumann, S.
Probleme, Defizite und Strategien bei der Integration heterogener Systeme mit HL7
43. Jahrestagung der GMDS, Bonn
MMV Verlag Medizin Verlag München, 1998 S.254-257

[Loeser 1998]
Loeser, H.
Techniken für Web-basierte Datenbankanwendungen: Anforderungen, Ansätze, Architekturen
Informatik Forschung und Entwicklung (1998) 13: 196-216

[Lulushi 2000]
Lulushi, A.
Oracle Forms Developer‘s Handbook
Prentice Hall 2000

[MedAccess]
MedAccess Handbuch
ACM Consult GmbH, Eschborn

[Messaritakis 1999]
Die HL7 Markup Language
Internetdokument

[Michel 1990]
Entwicklung eines orthopädischen Diagnosecodiersystems innerhalb des Klinikinformationssystems WING
Inaugural-Dissertation, 1990

[Muller 1997]
Muller, R. J.
Oracle Developer/2000 Handbook
Osborne/McGraw-Hill 1997

[Pott 1999]
Pott, O., Wielage, G.
xml, praxis und referenz
Markt&Technik, Buch- und Software-Verlag, München 1999

[Rahm 1999]
Rahm, E.
Datenbankeinsatz im Internet, Anbindungstechniken zwischen WWW und DBS
Problemseminar SS99, Universität Leipzig, Institut für Informatik

[Steimke 1995]
Steimke, F.
HL7: Standard für den Datenaustausch in Krankenhäusern? Pragmatischer Ansatz
iX 10/1995 S.92-96

[Turau 1999]
Turau, V.
Techniken zur Realisierung Web-basierter Anwendungen
Informatik Spektrum 22:3-12 (1999), Springer Verlag 1999

[Wagner 1995]
Wagner, G., Dudeck, J., Grundmann, E.
Organspezifische Tumordokumentation, Prinzipien und Verschlüsselungsanweisungen für Klinik und Praxis
Springer Verlag Berlin Heidelberg New York 1995

Lebenslauf

Name	Iris Stolte
Schulausbildung	1978-1984 Helene-Lange-Gymnasium, Ffm-Höchst
	1984-1987 Friedrich-Dessauer-Oberstufengymnasium, Ffm-Höchst
Studium	1987-1988 Western Oregon State College, Monmouth, Oregon, USA
	1988-1989 Universität Frankfurt/Main Studiengang Informatik, Nebenfach Medizin
	1989-4/1995 Universität Hildesheim/Niedersachsen Studiengang Informatik, Anwendungsfach Medizin Sommersemester 1992: Vordiplom
Wissenschaftliche Tätigkeit	6/1995-3/1996 Diplomarbeit am Forschungszentrum Karlsruhe GmbH Technik und Umwelt, Karlsruhe
	8/1996 Ende des Studiums
	4/1996-6/2001 Wissenschaftliche Mitarbeiterin am Institut für Medizinische Informatik der Justus-Liebig-Universität, Gießen

Danksagung

Mein Dank gilt vor allem Herrn Professor Dr. J. Dudeck für die Überlassung des Themas und für seine Anregungen bei der Durchsicht der Arbeit.

Weiterhin möchte ich mich bei meinen ehemaligen Arbeitskollegen Herrn Dr. U. Altmann und Herrn F. Katz für die Hilfestellung in vielen Bereichen bedanken.