Entwicklung von Metriken für XML-Dokumentkollektionen

Diplomarbeit

Universität Rostock, Fachbereich Informatik

vorgelegt von: Lars Schneider
geboren am: 21. April 1969 in Rostock

Betreuer: Prof. Dr. Andreas Heuer
Prof. Dr. Clemens Cap
Dr.-Ing. Meike Klettke

Abgabedatum: 28. Dezember 2001

*"Not everything that counts can be counted,
and not everything that can be counted counts."*

Albert Einstein

Zusammenfassung

XML ist eine Auszeichnungssprache und hat seit der Verabschiedung als Empfehlung 1998 in vielen Anwendungsbereichen eine zunehmende Bedeutung erlangt. Die einfache und einheitliche Beschreibung von Dokumenten und die Erweiterbarkeit sind dabei die wichtigsten Gründe für die Attraktivität von XML. Die Struktur von XML-Dokumentkollektionen wird durch die Dokumenttyp-Definition (DTD) beschrieben. Um gültige Dokumente zu erstellen oder Änderungen an der DTD durchzuführen, ist es notwendig, die DTD zu kennen. Dabei ist die Syntax der DTD relativ einfach und im allgemeinen gut lesbar. Dennoch gibt es DTDs, die schwerer lesbar und verständlich sind als andere.

In dieser Arbeit werden XML-Dokumentkollektionen auf der Grundlage ihrer allgemeinen Eigenschaften bewertet. Ausgangspunkt dieser Bewertung ist die DTD und im Mittelpunkt der Betrachtung stehen dabei die qualitativen Kriterien Änderbarkeit und Benutzbarkeit. Die beiden Kriterien sind wesentlich geprägt durch die Größe und Strukturiertheit der DTD und lassen sich durch die entwickelten Metriken quantitativ bewerten. Die Größe der DTD läßt sich direkt und die Strukturiertheit der DTD indirekt über einen speziellen DTD-Graphen bestimmen. Die entwickelten Metriken wurden evaluiert und auf Beispiel-DTDs angewendet.

Abstract

XML is a markup language and has gained an incremental importance since the publication as a recommendation in 1998 in many fields of application. The simple and uniform description of documents and the extendability are the most important reasons for the attractiveness of XML. The structure of XML document collections is described by the document type definition (DTD). To create valid documents or make modifications at the DTD, it is necessary to know the DTD. The syntax of the DTD is relatively simply and generally readable in a good way. Nevertheless there are DTDs, which are more heavily readable and understandable than others.

In this master thesis XML document collections are evaluated on the basis of their general characteristics. Starting point of this assessment is the DTD and the qualitative criteria maintainability and usability are in the focal point of the contemplation. The two criteria are substantially shaped by the size and structuredness of the DTD and are made through the developed metrics evaluate quantitatively. The size of the DTD can be determined directly and the structure of the DTD indirectly by a special DTD graph. The developed metrics were evaluated and applied on example DTDs.

CR-Klassifikation:

D.2.8	Metrics (Product metrics, Complexity measures)
G.2.2	Graph Theory
I.7.2	Document Preparation (Markup languages)
I.7.5	Document Capture (Document analysis)

Key Words:

XML, DTD, Softwaretechnik, Softwaremessung, Softwarequalität, Metrik, Komplexität

Inhaltsverzeichnis

Kapitel 1

Einleitung

XML ist ein vielversprechendes Datenformat für vielfältige Anwendungen und derzeit in aller Munde. Es vergeht kaum ein Monat, in dem es keine neuen Empfehlungen gibt. Da versteht es sich von selbst, daß die IT-Welt neuerdings nicht mehr nur kompatibel sondern nun auch X-patibel sein will. XML werden viele positive Softwareeigenschaften nachgesagt, wie Erweiterbarkeit, Flexibilität, Modularität, Lesbarkeit, Wiederverwendbarkeit etc.. Derzeit sucht man aber vergebens nach Möglichkeiten, diese Kriterien durch Metriken bewerten zu können, um so beispielsweise die Qualität der XML-Dokumente überprüfen zu können. Auf dieses Thema wird im folgenden Abschnitt eingegangen. Im Anschluß daran wird der Aufbau der Arbeit vorgestellt.

1.1 Motivation der Arbeit

XML wird zur Beschreibung, Speicherung und zum Austausch von Daten verwendet. Aufgrund seiner vielfältigen Verwendungsmöglichkeiten wird es immer populärer und häufig als zukünftiger Standard für das Internet betrachtet. Diese Tatsache ist Grund genug, weiterhin mit einer stark wachsenden Anzahl von XML-Dokumenten und Schemabeschreibungen – z.B. DTD, XML-Schema – zu rechnen. Im Gegensatz zu anderen Gebieten, wie Datenbanken, Softwaretechnik, gibt es jedoch bislang noch keine Kriterien, die die Qualität der Dokumente und des Entwurfs bewerten.

Die XML-Dokumentkollektionen basieren jeweils auf denselben Schemabeschreibungen, wie beispielsweise die HTML-Dokumente auf der HTML-DTD (SGML-DTD). Im Gegensatz zu HTML, die als Auszeichnungssprache konzipiert wurde, ist XML als Metasprache zu betrachten. Mit XML lassen sich eigene Dokumentgrammatiken bzw. Schemabeschreibungen definieren. Betrachtet man die Schemabeschreibungen näher, so lassen sich gewisse Parallelen zu den Lebenszyklusmodellen der Software oder auch zum Phasenmodell von Datenbanken herstellen, da die Schemabeschreibungen ähnliche Phasen durchlaufen. Aus diesem Grund liegt es nahe, an sie ähnliche qualitative Anforderungen zu stellen, wie beispielsweise in der Softwaretechnik. Um die qualitativen Eigenschaften messen zu können, müssen diese geeignet quantifizierbar sein. Das Forschungsgebiet, welches sich mit diesem Aufgabengebiet befaßt, ist die Softwaremetrie, ein Teilgebiet der Softwaretechnik.

Bei einer Bewertung von XML-Dokumentkollektionen sind die Kriterien Wohlgeformtheit und Gültigkeit der XML-Empfehlung nicht ausreichend. Sie können als formale Voraussetzung für die Bewertung betrachtet werden. Durch die Schemadefinition – beispielsweise durch DTDs – wird die Struktur von XML-Dokumentkollektionen definiert. Sind die XML-Dokumente gültig bezüglich der angegebenen Schemabeschreibung, so stimmen sie einerseits mir der XML-Spezifikation (Wohlgeformtheit) und andererseits mit den strukturellen Eigenschaften des Schemas (Gültigkeit) überein. Die Übereinstimmung der strukturellen Eigenschaften ist der Grund dafür, in dieser Arbeit die Schemabeschreibung als Ausgangspunkt der Bewertung zu betrachteten. Als Schemabeschreibungssprache wurde die DTD ausgewählt. Ziel ist es nun, zunächst qualitative Kriterien für die Bewertung von DTDs zu bestimmen und diese dann durch Metriken geeignet quantitativ zu bewerten. In diesem Zusammenhang soll auch betrachtet werden, inwieweit sich die Metriken auf XML-Dokumente anwenden lassen.

1.2 Aufbau der Arbeit

Im weiteren Verlauf der Arbeit wird zunächst in Kapitel 2 ein Überblick über XML gegeben. Dabei wird zuerst die Sprachentwicklung und -einordnung von XML beschrieben, danach die XML-Sprachfamilie zur Abgrenzung für die Entwicklung von XML-Metriken vorgestellt und ein prinzipieller Überblick über die Einsatzmöglichkeiten von XML gegeben. In diesem Kapitel wird der Aufbau der XML-Dokumente und die DTD-Definition vorgestellt, die als Grundlagen von XML für die Entwicklung der Metriken zu sehen sind. Weiterhin wird die XML-DTD dem XML-Schema gegenübergestellt und es werden grundlegende Gemeinsamkeiten sowie Unterschiede diskutiert.

Die Softwaremetrie ist ein noch relativ junges Teilgebiet der Softwaretechnik und beschäftigt sich mit der Bewertung von Software. Die Bewertung von Software ist dabei ein weitläufiger Begriff und reicht von einer einfachen experimentellen Messung einzelner Softwarekomponenten bis hin zur rechnergestützten Messung kompletter Softwareprozesse. In Kapitel 3 werden die wichtigsten Aspekte der Softwaremetrie, wie die Durchführung von Softwaremessungen, qualitative und quantitative Softwaremerkmale, vorgestellt. Hauptaugenmerk wird dabei besonders der Untersuchung vorhandener Softwaremetriken gewidmet, um festzustellen, inwieweit diese für die Bewertung von XML verwendet werden können.

Im Anschluß an diese beiden eher vorbereitenden Kapitel wird in Kapitel 4 die XML-Bewertung auf der Grundlage der DTD-Bewertung beschrieben. Basierend auf den Beschreibungen der XML-Definition aus Kapitel 2 werden hier zunächst XML-Bestandteile bestimmt, die direkt meßbar sind und als Ausgangspunkt für Metriken und spezielle Bewertungen verwendet werden können. Danach werden die einzelnen DTD-Metriken vorgestellt, wobei der Zusammenhang zu anderen bekannten Metriken aus der Softwaremetrie hergestellt, die Metrik definiert und evaluiert wird. In der Zusammenfassung des Kapitels sind die wichtigsten Aspekte der XML-Bewertung zusammengefaßt und im Anhang B wird die Anwendung der Metriken anhand von Beispiel-DTDs vorgestellt.

Die wichtigsten Aspekte der Arbeit sind in der Schlußbetrachtung in Kapitel 5 noch einmal zusammengefaßt. Desweiteren wird im Ausblick die Verwendung der entwickelten Metriken beschrieben und auf zukünftige Änderungen und Erweiterungen eingegangen.

Da sich im Verlauf der Arbeit mehrfach auf die Regeln der XML-Spezifikation bezogen wird, sind im Anhang A die EBNF-Regeln als Ausschnitt aus der aktuellen XML-Empfehlung aufgeführt. Im Zusammenhang mit XML werden viele Abkürzungen verwendet. Die in der Arbeit verwendeten Abkürzungen sind im Anhang C im Verzeichnis der Abkürzungen enthalten.

Kapitel 2

XML-Einführung

In diesem Kapitel wird zunächst ein Überblick über XML gegeben. Einführend wird beschrieben, wie es zur Entwicklung von XML kam und wie XML einzuordnen ist. Weiterhin wird durch einen Ausschnitt der XML-Sprachfamilie der aktuelle Stand der XML-Entwicklung verdeutlicht, der hier auch als Abgrenzung der XML-Basis [W3C00b] von anderen XML-Empfehlungen für die Diplomarbeit zu sehen ist. XML wird vielfältig eingesetzt, weswegen auch auf die hauptsächlichen Einsatzmöglichkeiten eingegangen wird. Darauf aufbauend werden im Kapitel 3 bekannte Bewertungsmethoden aus den entsprechenden Gebieten auf ihre Anwendbarkeit hin überprüft.

Im zweiten Abschnitt wird die aktuelle XML-Empfehlung vorgestellt. Dabei werden die XML-Dokumente und ihre Grammatiken (DTDs) soweit betrachtet, wie dies für das Verständnis der später zu entwickelnden Metriken benötigt wird.

Im letzten Abschnitt wird auf die Gemeinsamkeiten und Unterschiede zwischen der XML-DTD und dem XML-Schema eingegangen. Da die XML-Schema-Empfehlung noch nicht in der aktuellen Version der XML 2^{nd} edition eingeschlossen ist, wird die XML-DTD-Beschreibung die Basis für die Diplomarbeit bilden. Dies ist nicht als Nachteil zu sehen, was auch in diesem Abschnitt erläutert wird.

2.1 XML-Überblick

2.1.1 Entwicklung und Einordnung

Bei der Beschäftigung mit XML erkennt man zwei unterschiedliche Wurzeln, welche zum Entstehen von XML geführt haben. XML ist zum einen eine Weiterentwicklung bestehender Auszeichnungssprachen und wäre zum anderen wohl nicht entstanden ohne den Erfolg des WWW.

Die Grundidee der Auszeichnungssprachen ist es, abstrahierend vom Aussehen der Dokumente deren Struktur und Inhalt zu beschreiben. Die ersten Anfänge gehen dabei zeitlich gesehen sehr weit zurück und wurden 1969 von C. Goldfarb, E. Mosher und R. Loire bei IBM in GML zusammengefaßt. 1986 wurde dann eine Version von GML als ISO-Standard 8879 [ISO86] übernommen, besser bekannt unter dem Begriff SGML. SGML ist eine Sprache, welche die Definition eigener Sprachen ermöglicht und wird deswegen auch als Metasprache bezeichnet. Durch SGML wird kein fester problembezogener Sprachumfang vorgegeben, sondern eine Menge von strukturellen Konstrukten zur Definition von Dokumentgrammatiken.

HTML [W3C99e] ist das Ergebnis einer vereinfachten Sprache zur Formulierung von Dokumenten, welche elektronisch ausgetauscht werden können. Dabei basiert HTML auf SGML, denn HTML ist

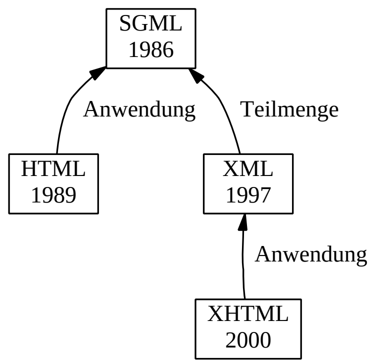


Abbildung 2.1: XML-Einordnung

ein spezieller Dokumenttyp von SGML. Gerade die Einfachheit von HTML gegenüber der Komplexität von SGML machte HTML schnell populär, führte aber auch zur teilweise unkontrollierten Anpassung an verschiedene Bedürfnisse. Als Folge dieser Entwicklung wurde eine große Menge von Elementen innerhalb von HTML definiert, die zu Kompatibilitätsproblemen der Dokumente auf unterschiedlichen Plattformen geführt hat.

Um die Nachteile von HTML zu überwinden, wurde vom W3C eine Arbeitsgruppe gebildet, deren Ziel es war, ein Web-SGML zu entwickeln. Dieser Sprachvorschlag ist als XML 1.0 [W3C98] bekannt und bildet eine Untermenge der Sprachmöglichkeiten von SGML, so daß jedes XML-Dokument auch ein gültiges SGML-Dokument ist. Hierbei wurden konsequenterweise nur benötigte Teile von SGML in XML übernommen. XML ist genauso wie SGML eine Metasprache zur Formulierung der Syntax von Dokumenten.

XHTML 1.0 [W3C00c] ist die Reformulierung von HTML 4.01 [W3C99e] als Anwendung von XML 1.0. Ziel war die Definition einer Sprache für Inhalte, die XML-konform sind und unter Beachtung einiger einfacher Richtlinien in HTML 4 konformen Benutzeragenten funktionieren. Der Vorteil beim Umstieg von HTML auf XHTML ist die Abwärts- wie auch die zukünftige Kompatibilität der Inhalte unter Nutzung der Vorteile von XML.

2.1.2 XML-Sprachfamilie

Aufgrund der Einfachheit von XML haben sich neben der eigentlichen XML-Sprache viele auf XML aufbauende Sprachen gebildet. In der Abbildung 2.2 wird der Zusammenhang von XML-Basistechniken und einigen XML-Sprachen - auch XML-Anwendungen genannt - dargestellt.
Inner- und außerhalb vom W3C entwickeln sehr viele Arbeitsgruppen aufbauend auf dem XML-Kern weitere Sprachen und optionale Module. Die Anzahl der XML-Anwendungen wächst rapide und es scheint auch noch kein Ende abzusehen zu sein. Eine aktuelle Zusammenstellung der technischen Berichte und Veröffentlichungen des W3Cs zu XML ist unter "http://www.w3.org/tr/" zu finden.

Die Grundlage der XML-Sprachen basiert zunächst auf der XML 1.0 Empfehlung vom Februar 1998 [W3C98]. Aktueller Stand dazu ist die XML 1.0 2nd edition [W3C00b], eine korrigierte und erweiterte Empfehlung zu XML 1.0, was bereits in der Abbildung 2.2 berücksichtigt wurde. Die überarbeitete Empfehlung schließt die zuvor getrennt erarbeitete Empfehlung zu XML-Namensräumen [W3C99a] mit ein. Die Document Type Definition (DTD) ist eine zusätzliche Sprache innerhalb der XML-Empfehlung zur Formulierung von XML-Grammatiken.

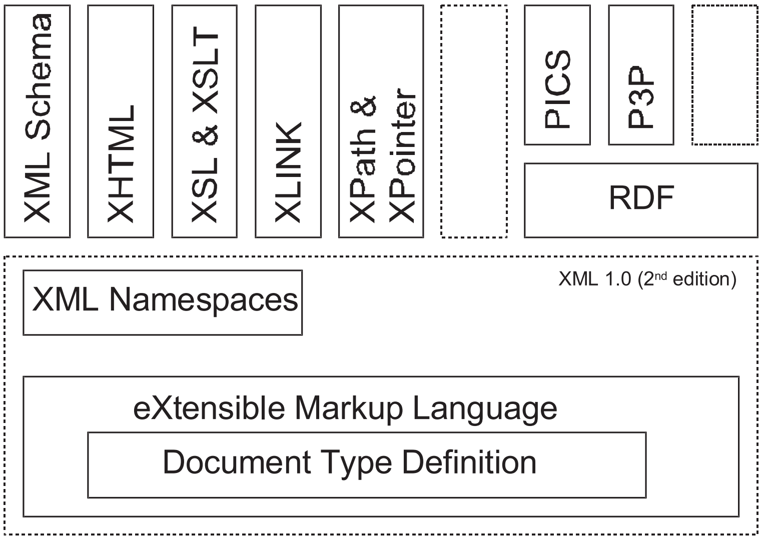


Abbildung 2.2: XML-Sprachen

XML-Schema ist eine in XML-Syntax formulierte kontextfreie reguläre Grammatik zur Definition von XML-Strukturen und im Sinne der vorherigen Beschreibung als eine XML-Anwendung zu betrachten. Die aktuelle Empfehlung wurde im Mai 2001 verabschiedet und besteht aus zwei Teilen, zum einen aus der Beschreibung der Struktur [W3C01a] und zum anderen aus der Beschreibung der Datentypen [W3C01b]. Da die Empfehlung zu XML-Schema sehr umfangreich ist, wurde zusätzlich in einer Einführung [W3C01f] ein umfassender Überblick mit Beispielen veröffentlicht. Die Bedeutung von DTDs wird in Zukunft wohl aufgrund der ausdrucksstärkeren Schemasprache abnehmen.

Einen wichtigen Bereich innerhalb von XML stellt die präsentationsorientierte Auszeichnungssprache XHTML 1.0 [W3C00c] durch die Reformulierung von HTML 4.01 [W3C99e] dar, welche durch das W3C nicht mehr weiterentwickelt wird. Wie bereits erwähnt, soll XML Nachteile von HTML überwinden. So ist es auch nicht verwunderlich, daß der Anwender mit XHTML zunächst einmal von HTML her bekannte Anwendungen, wie z.B. CSS Level 2, weiterhin nutzen kann. Dies wird in der Abbildung 2.2 nicht dargestellt, da CSS in HTML als Versuch anzusehen ist, Inhalte von Darstellungsinformationen zu trennen. Dies wurde jedoch nicht konsequent durchgesetzt. In SGML existiert in diesem Bereich eine ISO-Norm (DSSSL [ISO91b]), welche es ermöglicht, ein Dokument nach entsprechenden Transformationsregeln – z.B. in ein präsentations-optimiertes Format – zu überführen. Diese Aufgabe soll in XML XSL übernehmen, welche von den Funktionen her DSSSL entsprechen wird, nur bei weitem nicht so komplex ist. XSL besteht aus einem Formatierungsteil – XSL 1.0 [W3C00a] – für Formatierungsobjekte und einem Transformationsteil – XSLT 1.0 [W3C99c] – zur Transformation von XML-Dokumenten. Transformations- und Formatierungsteil von XSL arbeiten unabhängig voneinander.

Von der HTML-Sprache sind die Hyperlinks bekannt, welche eine nichtlineare Navigation inner- und außerhalb von HTML-Dokumenten ermöglichen. In XML steht ein umfangreicherer Verknüpfungsmechanismus zur Verfügung. XLink [W3C01e] definiert hierbei die äußeren Verknüpfungen eines Dokumentes mit anderen, XPointer [W3C01d] eine auf XPath [W3C99b] aufbauende Sprache, um Teile innerhalb eines Dokumentes zu adressieren.

RDF [W3C99d] ist eine XML-Anwendung zur formalen Beschreibung von Metadaten – Informationen über Informationen – und definiert die Codierung und Konventionen für den Austausch und die Wiederverwendung der Metadaten. Aufbauend auf diesem Standard werden verschiedene RDF-Vokabulare definiert. Diese RDF-Vokabulare können XML-Dokumente um semantische Informationen erweitern, sofern sich die RDF-Vokabulare auch als Standard-Vokabulare durchsetzen. Als Beispiele für RDF-Anwendungen seien hier PICS – Platform for Internet Content Selection – und P3P – Platform for Privacy Preferences – genannt. PICS spezifiziert eine Semantik zur Bewertung von Webseiten, z.B. zum Filtern der Seiten nach ihrem Inhalt. P3P ist ein Protokoll für die Übertragung von Anwenderdaten. Hierbei kommt es nur zur Übertragung von Daten, wenn der Anbieter einer Webseite angibt, welche Daten wozu übermittelt werden (Privacy Policy) und der Anwender dieser Datenweitergabe zustimmt. Aktuelle Informationen zu PICS sind unter "http://www.w3.org/PICS/" und zu P3P unter "http://www.w3.org/P3P/" zu finden.

Dieser Abschnitt ist als Ausschnitt über den aktuellen Stand der XML-Empfehlung und einiger wichtiger XML-Anwendungen zu betrachten. Im Zusammenhang mit der Entwicklung der Metriken für XML-Dokumentkollektionen werden sich die weiteren Betrachtungen zu XML bis auf eine Gegenüberstellung der DTD und XML-Schema in Abschnitt 2.3 auf den XML-Kern [W3C00b] ohne Namespaces beschränken.

2.1.3 XML-Einsatz

In den vorangegangenen Abschnitten wurden einige Gründe, welche zur Entwicklung von XML geführt haben und ein Ausschnitt aus dem Bereich der XML-Sprachfamilie vorgestellt. In diesem Abschnitt sollen einige Einsatzmöglichkeiten von XML beschrieben werden. Dies wird hier betrachtet, um im Kapitel 3 bekannte Metriken auf ähnlichen Gebieten vorzustellen.

Wie bereits erwähnt, basiert das Konzept von XML auf Trennung von Struktur, Inhalt und Präsentation. Die Teilung der logischen Komponenten eines Dokumentes ist im Gegensatz zu anderen Dokumentformaten – wie HTML, Word – die Voraussetzung für eine automatische Verarbeitung und erhöht die Flexibilität und Unabhängigkeit der Dokumente. Die Struktur wird durch die Regeln in der DTD festgelegt und gewährleistet dabei die Konsistenz gleichartiger Dokumente. Das eigentliche XML-Dokument enthält den Inhalt mit Auszeichnungen – Markups – zur Identifizierung der Datenelemente, also hier als Strukturinformation. Die Präsentationsinformation für die Darstellung auf einem Ausgabemedium – z.B. Bildschirm, Drucker – wird separat in einem Style Sheet festgelegt.

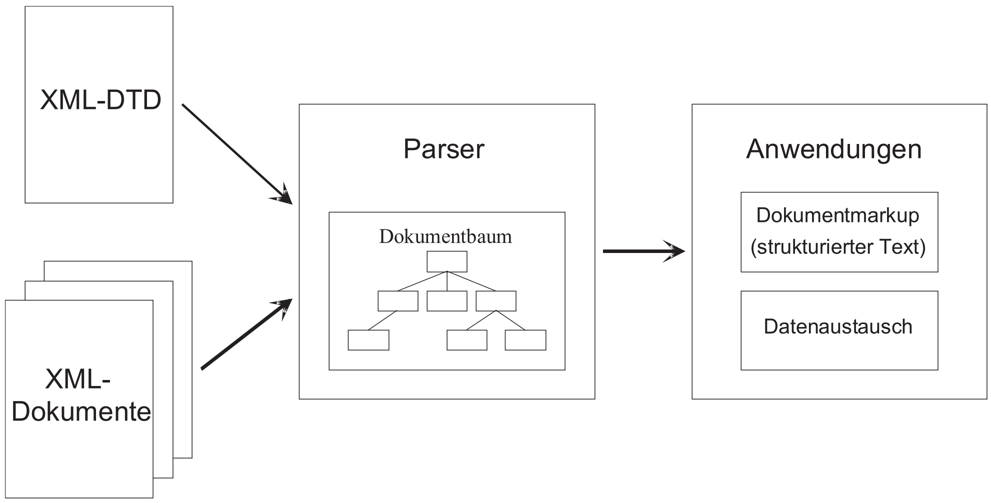


Abbildung 2.3: XML und Anwendungen

Wie in Abbildung 2.3 dargestellt, erfolgt die Verarbeitung der XML-Dokumente mit Hilfe eines XML-Parser. Es wird nach validierenden und nicht validierenden Parser unterschieden, je nachdem ob die Gültigkeit des XML-Dokumentes gegenüber der DTD geprüft wird oder nicht. Nicht validierende Parser prüfen lediglich die Wohlgeformtheit des XML-Dokumentes. Der Parser erzeugt aus einem XML-Dokument einen Dokument-Baum – im Sinne der objektorientierten Programmierung – damit Anwendungen auf die Inhalte des Dokuments zugreifen können.

Der Einsatz von XML ist vielfältig. Die Abbildung 2.4 entspricht einer ähnlichen Aufteilung wie in [Cho00]. Dieser Report beschreibt Kriterien für den XML-Entwurf. Aufgrund der DTD-Untersuchung wird in [Cho00] eine Einteilung der XML-Anwendungen vorgeschlagen. Einerseits wird XML als Auszeichnungssprache für Dokumente genutzt. Hier steht die Unabhängigkeit der XML-Dokumente gegenüber einer speziellen Präsentationsform im Vordergrund. Weiterhin ist die Wiederverwendbarkeit des so strukturierten Textes gegenüber proprietären Dateiformaten ein wichtiges Kriterium, da die Dokumente auch anderweitig verwendet werden können. Andererseits wird XML wegen seiner Einfachheit und Flexibilität als universelles Datenaustauschformat in vielfältigen Anwendungsbereichen eingesetzt. Mit XML lassen sich die unterschiedlichen Dokumentstrukturen modellieren, die beispielsweise beim Datenaustausch von Daten aus RDBS, ODBS oder diverser Anwendungsprogramme benötigt werden. Bei der Verwendung von DTDs gibt es einige Einschränkungen zu berücksichtigen, wie z.B. das unzureichende Datentypsystem. Diese Einschränkungen sollen mit der XML-Schemasprache aufgehoben werden.

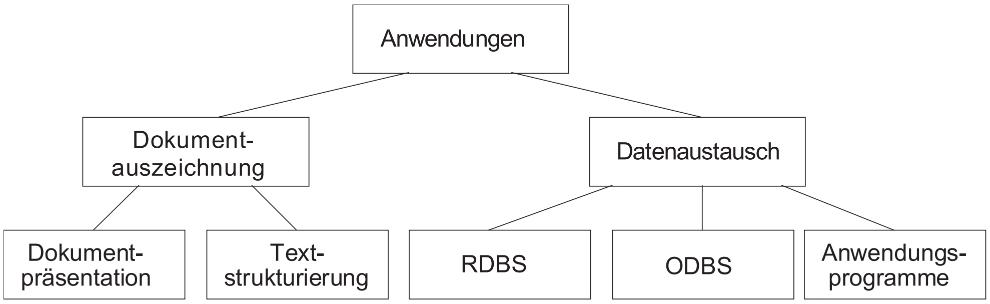


Abbildung 2.4: XML-Anwendungen

Bei der Einteilung der XML-Dokumente nach der Anwendung ist der Zusammenhang zwischen gewissen Eigenschaften der DTD – z.B. Flexibilität – und der speziellen Anwendung besonders interessant. Betrachtet man die DTDs entsprechend der genannten Anwendungen, so könnte man beispielsweise die DTDs in allgemeine und spezielle unterteilen. Bei den speziellen DTDs handelt es sich dabei um DTDs für Datenbank- oder auch Programmdaten, welche viele gleichartige Elemente enthalten. Deren Struktur wird wohl eher restriktiv sein, als die der allgemeinen DTDs z.B. für Bücher (DocBook-DTD) oder Webdokumente (HTML-DTD). Die Eigenschaft der DTD wirkt sich auf den Grad der Überdeckung zwischen DTD und den Dokumentinstanzen aus. Eine Kategorisierung der XML-Dokumente auf der Grundlage ihrer Verwendung ist aber schwierig, da es keine allgemeingültige Vorgehensweise bei der Datenmodellierung gibt. Das Problem "*Element versus Attribut*" wird im folgenden Abschnitt betrachtet. Die Betrachtung der grundlegenden bzw. allgemeinen DTD-Eigenschaften wird in Kapitel 4 im Zusammenhang mit der Metrikdefinition weitergeführt.

2.2 XML-Dokumente und XML-DTD

In diesem Abschnitt werden basierend auf der XML-Empfehlung [W3C98] bzw. [W3C00b] ohne Namespaces [W3C99a] die zum Verständnis der Arbeit benötigten Informationen vorgestellt. Der

Schwerpunkt liegt hierbei natürlich bei der DTD-Definition, da das Ziel der Diplomarbeit die Bewertung von DTDs ist. Verständliche Beschreibungen zur XML-Empfehlung [W3C98] mit guten Beispielen sind z.B. in [BM98] oder auch in [HE00a] nachzulesen. Vor der DTD-Beschreibung soll zunächst geklärt werden, wie XML-Dokumente aufgebaut sind.

2.2.1 XML-Dokumente

Wie bereits im Abschnitt 2.1.3 erwähnt, ist nicht zu jedem XML-Dokument eine DTD zwingend erforderlich. Dieser Fall soll im weiteren Verlauf der Arbeit nicht relevant sein. Wenn XML-Dokumente betrachtet werden, so soll ihnen eine DTD zugeordnet sein. Hierbei ist die Gültigkeit der Dokumente anhand der DTD-Regeln überprüfbar, wobei die Wohlgeformtheit des Dokumentes vorausgesetzt wird. Im folgenden sollen die EBNF-Notationen der XML-Empfehlung die Beschreibung des entsprechenden Sachverhaltes veranschaulichen, wobei hier nur die Regelnummer – z.B. `[1]` für die entsprechende XML-Dokument Regel – angegeben wird. Im Anhang A sind die EBNF-Regeln aus der aktuellen XML-Empfehlung [W3C00b] nachzulesen.

Ein XML-Dokument besitzt laut Spezifikation eine logische (siehe [W3C00b] Kapitel 3) und eine physische (siehe [W3C00b] Kapitel 4) Struktur. Physisch gesehen besteht ein XML-Dokument aus einer nichtleeren Reihenfolge-abhängigen Menge von Entities. Ein Entity kann andere Entities referenzieren, um diese in das Dokument einzubinden. Dies ermöglicht z.B. die Wiederverwendung von Daten. Außerdem können so auch Nicht-XML-Daten – z.B. Bilder, Video, Audio – in das Dokument eingefügt werden. Aus logischer Sicht besteht ein XML-Dokument `[1]` aus einem Prolog `[22]`, Elementen `[39]`, Kommentaren `[15]`, PIs `[16]` und Referenzen `[67]`. Die logische und physikalische Struktur müssen korrekt verschachtelt sein, wie in [W3C00b] Abschnitt 4.3.2 beschrieben.

Die Regel `[1]` legt fest, daß ein XML-Dokument aus einem optionalen Prolog `[22]`, mindestens einem Element `[39]` und weiteren verschiedenen Komponenten `[27]` besteht. Im Gegensatz zu HTML ist bei XML unbedingt auf Groß- und Kleinschreibung zu achten. Das gilt zum einen für die Schreibung der Schlüsselwörter, aber auch für die Namen von z.B. Elementen, Attributen, etc..

Der Prolog `[22]` leitet ein XML-Dokument ein und kann laut Spezifikation auch leer sein. Innerhalb der Prolog-Regel ist die Angabe einer XML-Deklaration `[23]` optional. Da bei der Überprüfung der Gültigkeit bzw. auch Wohlgeformtheit des Dokumentes die XML-Version wichtig werden wird, sollte auf diese Angabe nicht verzichtet werden. Zur Zeit ist die XML-Version auf die Konstante 1.0 festgelegt. Weiterhin können hier das gewählte Codierungsschema `[80]`, die Standalone-Deklaration `[32]`, die Dokumenttyp-Deklaration `[28]` und die unter `[27]` definierten Komponenten enthalten sein. Da hier nur gültige XML-Dokumente betrachtet werden sollen, ist im Prolog des XML-Dokumentes entweder eine interne `[28b]`, externe `[30]` oder eventuell auch gemischte Form `[28], [28b] und [30]` der Dokumenttyp-Deklaration vorhanden.

Dem Prolog folgt mindestens ein Element `[39]`, wobei genau ein Element das Wurzel- bzw. Dokumentelement ist und nicht im Inhalt eines anderen Elementes enthalten ist. Dies entspricht dem HTML-Tag von HTML. Alle Elemente sind strikt hierarchisch geschachtelt und werden durch einen Start- `[40]` und End-Tag `[42]` begrenzt. Es gibt zwei Arten von Elementen entsprechend der Regel `[39]`. Zum einen Elemente, die einen definierbaren Inhalt besitzen, wobei laut der Regel `[43]` der Inhalt auch leer sein kann. Zum anderen gibt es leere Elemente `[44]`. Die leeren Elemente müssen aber im Gegensatz zu leeren Elementen in HTML (z.B. IMG-Tag) korrekt geschlossen werden. Beide Elementarten können Attribute besitzen, welche im Start-Tag `[40]` bzw. Element-Tag `[44]` des Elementes angegeben werden. Sofern es sich um ein Element mit Inhalt handelt, wird in der Content-Regel `[43]` festgelegt, was als Elementinhalt angegeben werden darf. Die Elemente, der Elementinhalt und die Attribute werden im folgenden Abschnitt genauer beschrieben. Die gültigen Element- und Attributnamen müssen der Regel `[5]` entsprechen.

Laut der Regel [1] sind durch misc* (Regel [27]) verschiedene Komponenten im XML-Dokument erlaubt. Dazu zählen Kommentare [15], PIs [16] sowie Leerraum [3]. Kommentare in XML sind der Spezifikation nach mit HTML-Kommentaren identisch. Sie können laut Regel [1] und [22] überall stehen, wo Text stehen kann, außer in Element-Tags. PIs [16] sind Anweisungen für Anwendungsprogramme. Diese systemspezifischen Auszeichnungen werden an ein spezielles Anwendungsprogramm weitergeleitet, welches diese Informationen verarbeitet. Leerraum (Whitespace) [3] kann aus den Zeichen Leerzeichen, Tabulator, Zeilenvorschub und Wagenrücklauf bestehen und wird in XML im Gegensatz zu den anderen Zeichen gesondert betrachtet. Aus Datengesichtspunkten tragen diese Zeichen keine Information, sondern dienen meist der besseren visuellen Strukturierung des Dokument-Textes. Deshalb kann in der DTD angegeben werden, ob der Leerraum erhalten bleiben soll oder nicht.

2.2.2 XML-DTD

Eine XML-DTD beschreibt die Struktur und die logischen Elemente einer Klasse von XML-Dokumenten. Anders formuliert, dient die DTD als Muster oder Vorlage für eine Klasse von gleichartigen Dokumenten. Die Rolle der DTD entspricht dem Konzept der Klasse der objektorientierten Programmierung, die dort die logische Struktur und inhaltlichen Charakteristika beliebiger konkreter Ausprägungen festlegt. Analog kann man ein XML-Dokument auch als Objekt einer spezifischen Klasse (DTD) auffassen.

Die Verwendung von DTDs für XML-Dokumente mag einerseits als eine Einschränkung gesehen werden, denn in der DTD werden die zu verwendenden Elemente, deren Attribute sowie die logische Struktur der Elemente für XML-Dokumente festgelegt. Andererseits bringt gerade diese Einschränkung Vorteile. Wenn die XML-Dokumentkollektionen genau den Regeln der DTDs folgen, ist es möglich, die Informationen aus den Dokumenten auszulesen und automatisch zu verarbeiten. Ein großer Vorteil der DTD-Beschreibungen ist, daß ihre Syntax relativ einfach und im allgemeinen auch gut lesbar ist.

Wie bereits im vorherigen Abschnitt erwähnt, gibt es mehrere Möglichkeiten, einem XML-Dokument eine DTD [28] zugrunde zu legen. Die DTD kann in einer separaten Datei durch das externe Entity [75] und die externe Untermenge [30] definiert werden. In diesem Fall handelt es sich um eine externe DTD-Untermenge oder auch externe DTD. Wenn die DTD am Beginn, also im Prolog [22], von einem XML-Dokument definiert ist, handelt es sich um eine interne DTD-Untermenge [28b] oder auch interne DTD. Es ist aber auch möglich, die interne und externe DTD zu kombinieren. Hierbei können Namenskonflikte – z.B. der Elemente – auftreten, die durch die XML-Spezifikation gelöst werden, indem die zuerst gelesene DTD-Beschreibung den Vorzug erhält. Man kann das Problem auch durch konsequente Verwendung von Namespaces vermeiden. Prinzipiell macht es für die XML-Anwendung keinen Unterschied, ob die DTD intern, extern oder auch gemischt definiert ist. In der Praxis wird wohl weniger eine interne und eventuell auch eigene DTD für jedes XML-Dokument entworfen, sondern meist eine externe DTD für eine Klasse von XML-Dokumenten, sogenannten XML-Dokumentkollektionen, geschrieben. Aus dieser Sicht heraus ist die Verwendung der externen DTD-Beschreibung die flexiblere Variante. Durch die Verwendung einer internen DTD innerhalb einer Dokumentkollektion würde außerdem Redundanz entstehen. An dieser Stelle soll ein wesentlicher Unterschied der externen DTD nicht unerwähnt bleiben, denn im Gegensatz zur internen DTD [28b] lassen sich in der externen DTD [31] bedingte Abschnitte [61] definieren. In der externen DTD kann optional die XML-Version und Encoding-Deklaration – siehe Regel [30] und [77] – angegeben werden. Bei der internen DTD ist dies nicht der Fall, da diese optionale Definition bereits über den XML-Dokument Prolog [22] definiert wird.

Die DTD-Spezifikation ist, wie bereits in Abschnitt 2.1.2 beschrieben und in Abbildung 2.2 dargestellt, Bestandteil der XML-Empfehlung [W3C00b]. Im folgenden sollen alle in der DTD definierbaren Bestandteile vorgestellt werden. Dies ist notwendig, da sie die Basis zur Definition der Metriken im Kapitel 4 bilden, bzw. deren Grundeinheiten liefern. Kommentare und PIs können laut Regel `[29]` innerhalb von DTDs auftreten und wurden bereits im Zusammenhang mit den XML-Dokumenten im vorangegangenem Abschnitt beschrieben.

Elemente:

Bei der Verwendung der Elemente `[39]` in XML-Dokumenten wird aufgrund der unterschiedlichen Notation zwischen leeren Elementen `[44]` und Elementen mit Inhalt (`[40]`,`[42]``[43]`) unterschieden. In der DTD werden diese Elemente durch die Elementtyp-Deklaration `[45]` gleichermaßen definiert. Zunächst wird eine Elementtyp-Deklaration eingeleitet durch "`<!ELEMENT `", gefolgt vom Namen des Elementes entsprechend der Regel `[5]`. Danach wird das Inhaltsmodell entsprechend der Regel `[46]` angegeben und die Elementtyp-Deklaration mit ">" geschlossen.

In der Regel `[46]` werden die unterschiedlichen Inhaltsmodelle für Elemente festgelegt. Der Elementinhalt des Inhaltsmodells "`children`" besteht ausschließlich aus Elementen. Die Möglichkeiten der hierarchischen Verschachtelung der Elemente werden im Inhaltsmodell durch die logische Kombination der Elemente beschrieben, ähnlich der Notation für reguläre Ausdrücke. Dies ist durch die Regeln `[47]-[50]` genau festgelegt. Die einfachen Ausdrücke A und B sollen im folgenden den Elementinhalt symbolisieren. Die Klammer wird zur Gruppierung von Ausdrücken verwendet.

Wiederholungen

A? – A kann keinmal oder genau einmal angegeben werden

A+ – A muß mindestens einmal angegeben werden

A* – A kann keinmal oder mehrmals angegeben werden

Alternative

(A | B) – A oder B muß angegeben werden

Sequenz

(A , B) – A und B müssen in derselben Reihenfolge angegeben werden

Gruppierung

(...) – die Klammerung gruppiert zusammengehörende Ausdrücke

Neben dem genannten Element-Inhaltsmodell sind in der Regel `[46]` noch die folgenden angegeben:

EMPTY - Inhaltsmodell
Durch dieses Inhaltsmodell werden leere Elemente beschrieben, wie z.B. der `IMG`-Tag in HTML. Diese Elemente werden über ihre Attribute charakterisiert.

ANY - Inhaltsmodell
Das Inhaltsmodell für `ANY` erlaubt dem Element, daß jedes in der DTD deklarierte Element als Unterelement in diesem Element zusammen mit Markup-freiem Text auftreten kann. Die Reihenfolge wird nicht vorgeschrieben.

Mixed - Inhaltsmodell

Besteht der Elementinhalt aus Elementen und aus Daten (`#PCDATA`- Zeichenfolgen ohne Markup-Symbole), wird von gemischtem Inhalt gesprochen. Laut der Regel `[51]` kann der Inhalt des Elements aus gemischtem Inhalt (`#PCDATA` und Elementen) oder nur aus Freitext (`#PCDATA`) bestehen. In der Regel `[51]` ist die Reihenfolge der Definition für den gemischten Elementinhalt genau vorgegeben. Der `#PCDATA`-Abschnitt wird zuerst definiert und dann folgen die Elemente.

Attribute:

Attribute sollen Elemente genauer spezifizieren, können aber nicht wie Elemente frei definiert werden. Aus diesem Grund werden Attribute in Attributlisten zusammengefaßt und durch die Angabe des Elementnamens an das Element gebunden. Hierdurch wird vermieden, daß ein Attribut bzw. eine Attributliste an mehrere Elemente gebunden werden kann. Es ist aber möglich, mehrere Attributlistendeklarationen durch Vereinigung an ein Element zu binden. Dabei ist darauf zu achten, daß ein Attribut nicht mehrfach definiert wird. Die Attributdeklaration wird in [W3C00b] in den Regeln `[52]` - `[60]` genau beschrieben.

Jedem Attribut wird bei der Deklaration ein Attribut-Typ zugewiesen. Es gibt insgesamt 10 Attribut-Typen, die man in die folgenden drei Gruppen entsprechend der Regel `[54]` einteilen kann:

Zeichenketten-Typ	–	(`[55]` – `CDATA`)
Token-Typen	–	(`[56]` – `ID`, `IDREF`, `IDREFS`, `ENTITY`, `ENTITIES`, `NMTOKEN` und `NMTOKENS`)
Aufzählungs-Typen	–	(`[57]` – `NotationType [58]` und `Enumeration [59]`)

Die einzelnen Attribut-Typen haben folgende Bedeutung:

CDATA-Attribute

Der Wert dieser Attribute ist eine beliebige Zeichenfolge. Im Gegensatz zum Schlüsselwort `#PCDATA` handelt es sich bei `CDATA` um eine Markup-freie Zeichenfolge, die nicht geparst wird.

ID-Attribute

Über die ID-Attribute können Elemente in einem XML-Dokument identifiziert werden. Aus diesem Grund müssen alle ID-Werte in dem XML-Dokument unterschiedliche Werte besitzen. Ist dies einmal nicht der Fall, so muß der validierende Parser einen Fehler melden.

IDREF- und IDREFS-Attribute

Zusammen bilden die ID-, IDREF- und IDREFS-Attribute einen Dokument-internen rudimentären Referenzierungsmechanismus. Der Wert des IDREF- bzw. die Werte der IDREFS-Attribute verweisen auf im XML-Dokument vorkommende ID-Attribute. Ist ein Verweisziel nicht vorhanden, so muß der validierende Parser einen Fehler melden. Die ID-Attribute sind demzufolge unabhängig von den IDREF- und IDREFS-Attributen, aber nicht umgekehrt. Der Unterschied zwischen dem IDREF- und dem IDREFS-Attribut liegt wie zu vermuten darin, daß man mit IDREF genau ein Verweisziel und mit IDREFS eine Menge von Verweiszielen angeben kann.

ENTITY- und ENTITIES-Attribute
Der Wert des ENTITY-Attributs enthält den Namen eines externen, nicht analysierten (unparsed) Entity, also externe Binärdaten. Extern bedeutet hier soviel, daß das Entity in der DTD definiert wurde. Da das Entity Binärdaten – z.B. eine Bilddatei – enthält, wird es nicht geparst. Die ENTITIES-Attribute verhalten sich gleich, erlauben nur die Angabe mehrerer Entity-Namen.

NMTOKEN- und NMTOKENS-Attribute
Der Wert eines NMTOKEN-Attributs entspricht prinzipiell dem eines CDATA-Abschnittes. Dieser Abschnitt enthält Markup-freie Zeichen. Im Unterschied zu CDATA muß der WERT des NMTOKEN-Attributs der Regel `[7]` und der Wert des NMTOKENS-Attributs der Regel `[8]` entsprechen.

NOTATION-Attribute
Der Wert des NOTATION-Attributs ist der Name einer Notation, welcher in der DTD deklariert ist. NOTATION-Attribute zählt man entgegen den ENTITIES- und NMTOKENS-Attributen zu den Aufzählungstypen, da in ihrer Attribut-Deklaration eine Auswahlliste von möglichen Werten angegeben werden kann, von denen als Attribut-Wert im Element einer ausgewählt wird. Der Wert der beiden anderen Attributtypen ist eine Liste von Werten, getrennt durch Leerzeichen.

Enumeration-Attribute
Der Wert des Enumeration-Attributs ist ein Name, welcher in der Auswahlliste der Attribut-Deklaration vorkommt. Die Namen in der Auswahlliste der Deklaration müssen gültigen XML-Namen entsprechen, ähnlichen der Namensspezifikation für NMTOKEN `[7]`.

Weiterhin können zu den Attributen Attribut-Vorgaben entsprechend der Regel `[53]` und `[60]` definiert werden, die folgende Bedeutung haben:

`#REQUIRED`	–	Das Attribut muß beim Auftreten des Elementes angegeben werden, ansonsten meldet der Parser einen Fehler (ungültiges Dokument).
`#IMPLIED`	–	Das Attribut muß nicht unbedingt beim Auftreten des Elementes angegeben werden (Attribut hat keinen Vorgabewert). Der Parser meldet lediglich das Fehlen des Attributes.
`#FIXED "WERT"`	–	Das Attribut besitzt einen festen Vorgabewert, der auch beim Weglassen des Attributes als Wert gesetzt wird. Falls der Attributwert nicht dem festen Vorgabewert entspricht, meldet der Parser einen Fehler.
`"WERT"`	–	Das Attribut besitzt einen Vorgabewert, der beim Weglassen des Attributes als Wert gesetzt wird.

Element versus Attribut:

Zunächst einmal unterscheiden sich Elemente und Attribute durch ihre Spezifikation in der XML-Empfehlung. Attribute werden Elementen zugeordnet und beschreiben gewisse Eigenschaften von Elementen.

Die Attributwerte sind Markup-frei, also ihr Inhalt wird nicht nach weiterem Markup durch den

Parser analysiert. Man kann allgemein sagen, daß Attribute nicht den eigentlichen Dokumentinhalt – z.B. bei Büchern die Kapitel, Abschnitte, Absätze – enthalten. Dagegen werden die Elemente analysiert, da sie den Dokumentinhalt mit weiterem Markup enthalten.

Die Attribut-Deklaration erlaubt die Einschränkung der gültigen Attributwerte. Elemente besitzen ein definiertes Inhaltsmodell und dadurch eine festgelegte logische Struktur. Ihr Inhalt läßt sich aber in der DTD nicht auf spezielle Werte einschränken.

Die Attribute charakterisieren die Elemente. Ihre Werte enthalten Metainformationen zum Element, die durch die XML-Anwendung verarbeitet werden. Das Element selbst enthält keine Metainformation, abgesehen vielleicht vom Elementnamen. Der Elementname unterstützt aber nur die Lesbarkeit des XML-Dokumentes durch einen menschlichen Betrachter und nicht die maschinelle Verarbeitung durch eine XML-Anwendung.

Ob eine Information als Element oder Attribut in der DTD definiert wird, muß hier genauso wie bei der konzeptuellen Modellierung, beispielsweise beim OODM oder RDM, im Entwurfsprozeß getroffen werden. Problematisch ist aber, daß es keine allgemeine Vorgehensweise bei der Datenmodellierung im DTD-Entwurf gibt. In [BM98] wird dem Entwurf einer DTD ein ganzes Kapitel gewidmet und dabei auch ausführlich auf den hier angesprochenen Unterschied eingegangen. Weiterhin finden sich im Internet unter "http://www.oasis-open.org/cover/elementsAndAttrs.html" Informationen zu diesem Thema. Gerade dieses Problem hat Auswirkungen auf die Bewertung von DTDs, da sich nur Eigenschaften bewerten lassen, die allgemein bestimmbar sind. Die Bewertung spezieller Element- und Attributbeziehungen lassen sich so nur aus anwendungsspezifischer Sicht bewerten, sofern dort eine allgemeingültige Datenmodellierung vorgegeben wird.

Entities:

Ein Entity kann in XML z.B. eine Datei, ein Teil einer Datei oder auch ein Zeichen bzw. eine Zeichenfolge sein. Prinzipiell bestehen alle Entities aus einem Namen und Inhalt. Auf ein Entity wird mittels einer Entity-Referenz verwiesen, wobei der Parser die Aufgabe hat, die Entity-Referenz durch den Entity-Inhalt zu ersetzen. Alle Entity-Typen werden in der DTD definiert. Der Unterschied in der Verwendung besteht darin, daß allgemeine Entities im XML-Dokument und Parameter-Entities in der DTD verwendet werden.

Laut der Spezifikation für Entities `[70]` gibt es zwei Typen, allgemeine Entities `[71]` und Parameter-Entities `[72]`. Die auffälligste Unterscheidung in der Deklaration ist, daß dem Namen des Parameter-Entity das Zeichen "`%`" vorangestellt wird. Allgemeine Entities werden mit "`&Entity-Name;`" und Parameter-Entities mit "`%Entity-Name;`" referenziert. Die Entities lassen sich nach den Eigenschaften intern oder extern und geparst oder nicht geparst unterscheiden. Ein allgemeines Entity kann als internes geparstes, externes geparstes oder externes nicht geparstes Entity (siehe Regel `[73]`) auftreten. Ein Parameter-Entity kann als internes geparstes oder externes geparstes Entity (siehe Regel `[74]`) vorkommen.

In XML sind fünf interne allgemeine Entities vordefiniert. Diese Entities stehen für spezielle Zeichen, welche im Zusammenhang mit XML eine besondere Bedeutung für das Markup haben und geschützt sind. Falls diese Zeichen als Zeichendaten innerhalb des Dokumentes vorkommen, müssen die folgenden Entity-Referenzen verwendet werden (siehe [W3C98] Abschnitt 2.4):

`<` – für das Zeichen ($<$)
`>` – für das Zeichen ($>$)
`&` – für das Zeichen (&)
`'` – für das Zeichen (')
`"` – für das Zeichen (")

Notationen:

Notationen werden innerhalb von DTDs deklariert und beschreiben das Format von Nicht-XML-Daten. Diese Daten werden wie bereits beschrieben in PIs (`[16]`), nicht geparsten Entities (`[71]`, `[73]` und `[76]`) und dem Notation-Attribut (`[54]`, `[57]` und `[58]`) in Elementen verwendet. Durch die Notations-Deklaration `[82]` wird der Notation ein Name und Bezeichner (URI) zugewiesen. Über den Namen der Notation wird durch den Parser aus der PI, dem Entity oder dem Notation-Attribut der Bezeichner ermittelt und die Nicht-XML-Daten an das im Bezeichner angegebene Programm zur Verarbeitung übergeben.

Weiteres:

Wie bereits am Anfang im Zusammenhang mit der DTD-Deklaration erwähnt (siehe auch Regel `[28]` - `[31]`), können neben Elementen, Attributen, Entities und Notationen innerhalb der DTD auch PIs und Kommentare (siehe auch `[29]`) eingefügt werden. Außerdem ist es in externen DTDs möglich, bedingte Abschnitte (siehe `[61]` - `[65]`) zu definieren.

Die PIs `[16]` werden in XML dafür verwendet (siehe auch Abschnitt 2.2.1), um in der DTD bzw. dem XML-Dokument spezifische Anweisungen, die nicht direkt vom Parser verarbeitet werden, zu integrieren. In HTML wurden dafür zum Teil die Kommentare mißbräuchlich benutzt, um beispielsweise Script-Anweisungen unterzubringen. In XML sollen Kommentare `[15]` reinen Dokumentationszwecken dienen und werden vom Parser bzw. sollen auch von der XML-Anwendung überlesen werden. Somit stellt ein XML-Dokument bzw. eine DTD mit oder ohne Kommentare für den Parser oder die Anwendung die gleiche Instanz dar.

Bedingte Abschnitte `[61]` - `[65]` sollen der Vollständigkeit halber an dieser Stelle auch erwähnt werden. Sie können nur in externen DTDs (siehe `[31]`) definiert werden. Ein denkbarer Einsatz wäre z.B. die Verwendung einer Gesamt-DTD, von der aus mehrere Teil-DTDs referenziert werden. Über den Wert der Parameter-Entities in der Gesamt-DTD könnte man bedingte – hier z.B. nicht benötigte – Abschnitte in den referenzierten Teil-DTDs ein- bzw. ausschließen.

Nachdem die wichtigen DTD-Bestandteile beschrieben wurden, werden im folgenden Abschnitt XML-DTDs und XML-Schema miteinander verglichen. Die DTD-Bestandteile bilden im Kapitel 4 die Grundlage für die zu entwickelnden Metriken.

2.3 XML-DTD versus XML-Schema

Im Abschnitt 2.1.2 wurde XML-Schema ([W3C01a], [W3C01b] und [W3C01f]) bereits erwähnt und in die aktuelle und zukünftige XML-Sprachfamilie eingeordnet. XML-Schema – kurz XSD – ist zunächst ein neuer und erweiterter Ansatz zur Definition von Dokument-Typen, also zur Definition neuer XML-Sprachen. Der DTD-Mechanismus bietet bisher eine Grammatik zur Beschreibung von Dokumentstrukturen, die aber stark Dokument-orientiert ist. XSD ist zusätzlich Daten-orientiert, da es spezielle Typdefinitionen der Elementinhalte ermöglicht. Dahinter steckt das Ziel, den *impedance mismatch* zwischen XML-Daten und Programm- oder auch Datenbankdaten bezüglich des unzureichenden und starren Datentypsystems zu überwinden. Die breite Verwendung von XML für beliebige Anwendungen (siehe auch Abschnitt 2.1.3 und Abbildung 2.4) offenbaren zunehmend Nachteile und Grenzen der DTDs, wie z.B. (aus [Jec01]):

- unzureichende Datentypunterstützung
  (begrenzt auf die vier Element-Inhaltsmodelle sowie vorgegebenen Attributtypen)

- starres Typsystem
  (keine Erweiterung durch den Anwender möglich)
- unzureichende Strukturierungsunterstützung
  (z.B. keine Beschränkung der Auftrittshäufigkeit (MIN, MAX) von Kindelementen)
- rudimentärer Referenzierungsmechanismus
  (nur lokale Referenzierung über IDREF und IDREFS)
- keine Unterstützung der Wiederverwendbarkeit
  (nur rudimentär über Parameter-Entities definierbar)
- keine Unterstützung von Namespaces
- XML- und DTD-Syntax nicht gleich

Diese Nachteile sollen durch XSD überwunden werden, wobei XSD die Ausdrucksmächtigkeit der DTDs erweitern soll. Die bisher beschriebenen Bestandteile der DTD werden im folgenden XSD gegenübergestellt.

In Abschnitt 2.1.2 wurde bereits auf die Einteilung des XSD-Sprachvorschlages in "Strukturteil" und "Datentypen" eingegangen. Der XSD-Part 1 basiert auf der Dokument-orientierten Sicht der DTD und erweitert diese an einigen Stellen. Die Erweiterungen sollen hier aber nicht Gegenstand des Vergleiches sein. Die in XSD-Part 2 definierten Datentypen basieren auf den DTD-Datentypen. Diese sind um die gebräuchlichsten von Programmiersprachen und Datenbanken her bekannten Typen erweitert.

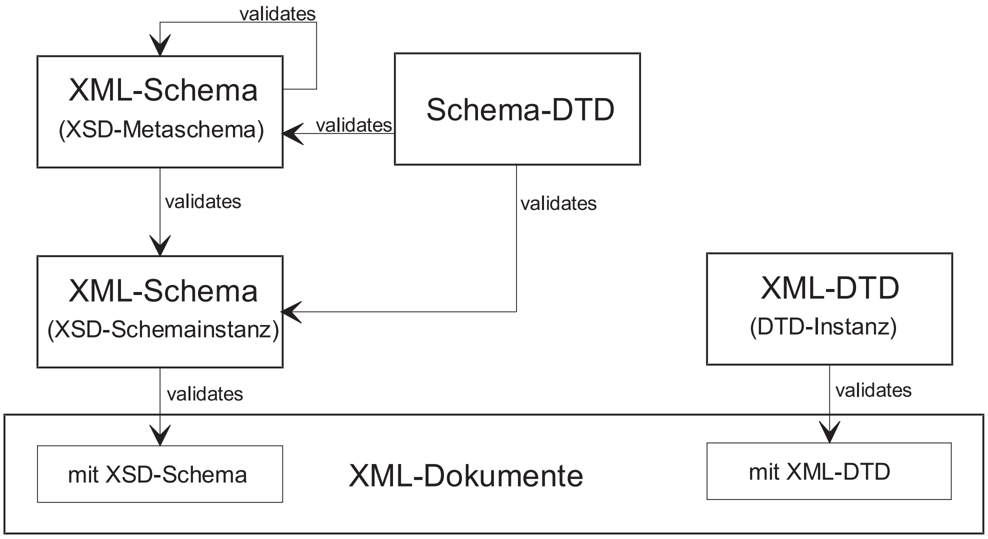


Abbildung 2.5: XSD- vs. DTD-Validierung (aus [Jec01])

Im Gegensatz zur DTD kann ein XML-Schema nicht als *internal subset* im XML-Dokument definiert werden, sondern nur extern. Über eine Schemareferenz im Wurzelelement des XML-Dokumentes wird das XML-Schema dem Dokument zugewiesen. Dabei werden durch XML-Schema ähnlich wie durch die DTD Regeln definiert, nach denen die XML-Dokumente auf Gültigkeit (schema validness) überprüft werden können. Im Gegensatz zur DTD ist das XML-Schema selbst ein XML-Dokument, da XSD als XML-Sprache realisiert wurde. Dies bedingt einen anderen Validierungsmechanismus als den von den DTDs her bekannten. Für XSD wurde ein Schema – Metaschema – geschrieben, so daß dies die Überprüfung der Gültigkeit einer Schema-Instanz ermöglicht, und zwar mit denselben Werkzeugen, mit denen auch die XML-Dokumente validiert werden. Zusätzlich hat die XSD-Arbeitsgruppe des W3C eine Schema-DTD entwickelt, so daß es für jede neue Schema-Instanz möglich ist, die Gültigkeit gegenüber der Schema-DTD mit den bisherigen Mitteln (validierende Parser) zu überprüfen. Eine DTD kann im Gegensatz zur Schema-Instanz nicht selbst validiert werden. In Abbildung 2.5 ist der Validierungsprozeß für XML-Dokumente mit einer Schema-Instanz denen mit DTDs gegenübergestellt.

In der gleichen Reihenfolge wie in Abschnitt 2.2.2 werden die DTD-Konstrukte denen des XML-Schemas gegenübergestellt, sofern diese eine Entsprechung in XSD haben. Erwähnt werden dabei auch die wesentlichen Unterschiede und eventuellen Vorteile der Schema-Konstrukte gegenüber den DTD-Konstrukten.

Elemente:

Die Element-Inhaltsmodelle der DTD werden durch XSD unterstützt, wobei die Elementdefinition um eine genauere Typdeklaration erweitert ist. Der Typ kann innerhalb – im Start-Tag – und außerhalb – zwischen Start- und End-Tag – der Tag-Deklaration des Elementes angegeben werden. Besonders der unstrukturierte Typ der DTD für Zeichenfolgen (#PCDATA) ist erweitert worden, wobei jetzt 44 einfache Typen zur Verfügung stehen. Darunter sind die bekannten Element- und Attribut-Typen sowie eine Menge neuer Typen.

Tabelle 2.1: DTD- versus XSD-Element-Inhaltsmodelle

XML-DTD	**XML-Schema**	**Bemerkung**
Elementwiederholungen und -verschachtelungen		
”?”; ”+”; ”*”	ersetzt durch Attribute minOccurs, maxOccurs	Angabe von der Minimal- und Maximalkardinalität ist flexibler
”|”	<choice>...</choice>	Alternative (bzw. Auswahl)
”,”	<sequence>...</sequence>	Sequenz
”(...)”	<group>...</group>	Gruppierung
keine direkte Entsprechung	<all>...</all>	spezielles Reihenfolge-unabhängiges Inhaltsmodell, in dem entweder keine oder alle Elemente vorkommen
Element-Inhaltsmodelle		
EMPTY	keine direkte Entsprechung	durch leeren complexType definierbar (Bsp. in XSD-Part 0 [W3C01f])
ANY	<element name=”anything” type=”anyType”/>	jeder Typ aus einfachen oder komplexen Typen ist einsetzbar (default Wert, falls kein Element-Typ definiert)
MIXED	<complexType mixed=”true”> ...</complexType>	in XSD vollständige Validierung , daher auch Reihenfolge und Auftrittshäufigkeit validiert
#PCDATA	ersetzt durch Typen aus simpleType	siehe XSD-Part 2 [W3C01b]

Attribute:

Die Attribute können in XSD einzeln über das XSD-Element attribute oder auch als Gruppe – entsprechend der DTD ATTLIST – mit attributegroup definiert werden. Desweiteren ist es möglich, sie innerhalb oder außerhalb der Elementdeklaration zu definieren. Falls ein Attribut außerhalb definiert wurde, wird innerhalb der Elementdeklaration durch eine Attributdeklaration über das `ref`-Attribut des `attribute`-Elementes das Attribut referenziert. Dieses Attribut kann außerdem von mehreren Elementen aus referenziert werden, was eine Wiederverwendung der Attribute ermöglicht. In der folgenden Tabelle werden die möglichen DTD-Attribute und Attributbedingungen denen von XSD gegenübergestellt. Es werden nicht alle Möglichkeiten der XSD-Attributdefinition genannt, wie z.B. die Schlüsselbeziehungen, Namespace-Integration mit der Möglichkeit der Verwendung extern definierter Attribute, weitere Bedingungsdefinitionen, aber gezeigt, wie die DTD-Attribute in XSD definiert werden können.

Tabelle 2.2: DTD- versus XSD-Attribute

XML-DTD	**XML-Schema**	**Bemerkung**
Zeichenketten-Typ		
`CDATA`	`type=”string”`	`simpleType` Attribut-Typ
Token-Typen		
`ID`	`type=”ID”`	`simpleType` Attribut-Typ
`IDREF` / `IDREFS`	`type=”IDREF”` o. `”IDREFS”`	`simpleType` Attribut-Typ
`ENTITY` / `ENTITIES`	`type=”ENTITY”` o. `”ENTITIES”`	`simpleType` Attribut-Typ
`NMTOKEN` / `NMTOKENS`	`type=”NMTOKEN”` o. `”NMTOKENS”`	`simpleType` Attribut-Typ
Aufzählungs-Typen		
`NOTATION`	`type=”NOTATION”`	`simpleType` Attribut-Typ
`ENUMERATION`	<simpleType name=”WO-TAGE”> <restriction base=”string”> <enumeration value=”MO”/> ... <enumeration value=”SO”/> </restriction> </simpleType>	Benutzer-definierter Typ, vollständige Aufzählung aller Werte, Werte entsprechen dem `simpleType` Attribut-Typ
Attributbedingungen bzw. -vorgaben		
`#REQUIRED`	`use=”required”`	Attributeigenschaft
`#IMPLIED`	`use=”optional”`	Attributeigenschaft
`#FIXED ”WERT”`	`fixed=”true”`	Attributeigenschaft mit Wert
`”WERT”`	`default=”false”`	Attributeigenschaft mit Wert

Entities und Notationen:

Die Entities wurden nicht in XSD übernommen, dafür stehen weitaus mächtigere Mechanismen zur Modularisierung von XML-Schemas zur Verfügung. Notation-Deklarationen in XSD sind in Aufbau und Funktion denen der DTD sehr ähnlich.

Weiteres:

Die Kommentare und PIs der DTD können in XSD mit dem `annotation`-Element definiert werden. Dieses Element besteht aus zwei Kindelementen. Zum einen aus dem `documentation`-Element für gewöhnliche Kommentare und zum anderen aus dem `appinfo`-Element, welches die Informationen für die verarbeitende Anwendung ähnlich den PIs enthält. Bedingte Abschnitte gibt es nicht, aber dafür andere Möglichkeiten der Strukturierung und Modularisierung.

Aus der Gegenüberstellung ist ersichtlich, daß XML-Schema die XML-DTDs ersetzen kann. Da hier nicht die Vorteile von XML-Schema gegenüber den XML-DTDs im Vordergrund standen, wurde nicht speziell auf die verbesserten Möglichkeiten zur internen Strukturierung und Modularisierung sowie der größeren Ausdrucksmächtigkeit, insbesondere durch die Beschreibung von semantischen Gültigkeitsbedingungen und Wertebereichen, eingegangen. Obwohl XML-Schema als "*Recommendation*" seit Mai 2001 verabschiedet ist, soll XML-DTD die Grundlage für weitere Betrachtungen in dieser Arbeit sein. Dies hat hier den Grund, daß es zur Zeit viele DTDs, aber nur wenige Schemas zu XML-Dokumenten gibt. Desweiteren ist die DTD übersichtlicher, für den menschlichen Betrachter lesbarer und somit erst einmal besser geeignet, um Bewertungen durchzuführen. Da XML-Schema eine Erweiterung darstellt, ist dies hier auch nicht als Nachteil zu sehen, da sich das Wissen über DTD-Metriken dann auch auf XML-Schema übertragen läßt.

2.4 Zusammenfassung

In diesem Kapitel wurde ein Überblick über XML gegeben. Dabei wurde zunächst beschrieben, welchen Hintergrund die Entwicklung von XML hat und wie XML sprachlich einzuordnen ist. Die stürmische Entwicklung von XML hat bereits innerhalb einer kurzen Zeit zu vielen XML-Sprachen bzw. -Anwendungen geführt, wobei für die Bewertung von XML-Dokumentkollektionen nur die aktuelle Recommendation [W3C00b] als Grundlage dient. Diese wurde zum besseren Verständnis der zu entwickelnden Metriken in diesem Kapitel umfassend beschrieben.

Der Einsatz von XML ist vielfältig und somit auch die Art der Dokumenttypen. Problematisch für die Bewertung ist in diesem Zusammenhang, daß nicht wie aus anderen bekannten Entwurfsmodellen – z.B. ER, EER, OOD – eine allgemeine Vorgehensweise bei der Datenmodellierung existiert. Dies schränkt die Bewertung der XML-Dokumentkollektionen ein wenig ein, wird aber zunächst nicht als großer Nachteil gesehen. Aus anwendungsspezifischer Sicht werden spezielle zusätzliche Betrachtungen notwendig sein, die bei der hier durchgeführten allgemeinen Betrachtung nicht relevant sind.

Auch wenn aktuell XML-Schema bereits als Empfehlung verabschiedet wurde, sollen die Metriken noch basierend auf der aktuellen XML-Empfehlung entwickelt werden. Die Gegenüberstellung der XML-DTD und XML-Schema hat gezeigt, daß die wesentlichen strukturellen Aspekte ähnlich sind. Aus diesem Grund wird es unter Beachtung der übernommen Deklarationen möglich sein, die DTD-Metriken auch prinzipiell auf XML-Schema anzuwenden.

Kapitel 3

Softwaremetrie

In diesem Kapitel wird ein Überblick über die Softwaremetrie, ein noch junges Teilgebiet der Softwaretechnik, gegeben. Die Softwaremetrie beschäftigt sich mit der Softwaremessung, wobei dies ein ziemlich weitläufiger Begriff ist. Softwaremessung reicht von einer einfachen experimentellen Messung einzelner Softwarekomponenten bis hin zur rechnergestützten Messung kompletter Softwareprozesse. Das primäre Ziel durch die Softwaremessung ist die Verbesserung der Qualität von Software und dadurch die Erhöhung der Produktivität bei der Erstellung und Verwendung von Software. Um das Ziel zu erreichen, wurden bereits unterschiedliche Standards – wie z.B. ISO 9000−3, ISO 9126, Capability Maturity Model (CMM) – geschaffen.

Der Begriff der *Softwarequalität* ist entsprechend der DIN 8402 definiert als Gesamtheit von Eigenschaften und Merkmalen eines Softwareproduktes, die geeignet sind, festgelegte und vorausgesetzte Anforderungen dieses Softwareproduktes zu erfüllen. Diese Definition ist zwar allgemein, fordert aber eine vollständige Übereinstimmung zwischen den Eigenschaften und Anforderungen der Software. Praktisch läßt sich der Grad der Übereinstimmung überprüfen, indem man die Softwarequalität meßbar und bewertbar macht. Softwarequalität ist zunächst einmal durch unterschiedliche *Qualitätsmerkmale* – wie Funktionalität, Zuverlässigkeit – gekennzeichnet. Diese Qualitätsmerkmale lassen sich in weitere Untermerkmale verfeinern, wobei ein Untermerkmal auch zu mehreren Merkmalen gehören kann. Die Aufteilung der Softwarequalität von allgemeinen in spezielle Qualitätsmerkmale hat das Ziel, die Qualitätseigenschaften durch Softwaremaße bzw. Softwaremetriken meßbar und bewertbar zu machen. Allgemein ist eine Metrik eine zahlenmäßige bzw. quantitative Beschreibung einer Eigenschaft. In dem IEEE Standard 1061 wird der Begriff der Softwaremetrik folgendermaßen definiert: ”Eine *Softwarequalitätsmetrik* ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.” In der Softwaretechnik werden die beiden Begriffe Softwaremaß und Softwaremetrik meistens synonym genutzt, auch wenn im mathematischen Sinn ein Unterschied zwischen einem Maß und einer Metrik besteht. Aus diesem Grund werden in der Diplomarbeit die beiden Begriffe auch gleichbedeutend benutzt.

Der oben beschriebene Prozeß, also von allgemeinen Qualitätsmerkmalen bis zu Softwarequalitätsmetriken, soll im folgenden Abschnitt an ausgewählten Normen und Modellen vorgestellt werden. Dabei wird zunächst ein kurzer Überblick über das Gebiet der Softwaremessung gegeben, bevor die allgemeinen Qualitätsmerkmale von Software entsprechend der ISO 9126 [ISO91a] beschrieben werden. Die Grundlage ist die IS0 9126, da hier nur die Qualitätsmerkmale primär von Interesse sind und nicht beispielsweise die Durchführung eines kompletten Qualitätsverbesserungsprozesses, wie er in der ISO 9000-3 beschrieben wird. Darauf aufbauend sollen Softwaremetriken und einige ihrer Merkmale beschrieben werden.

Im zweiten Abschnitt werden bekannte Softwaremetriken und Metriksammlungen vorgestellt. Die

Auswahl wurde hier auf Metriken beschränkt, die eventuell als Basismetriken für die DTD genutzt werden können. Die beiden vorgestellten Metriksammlungen enthalten Metriken zur Bewertung von unterschiedlichen konzeptuellen Modellen. Diese Metriksammlungen werden hier vorgestellt, da die DTD in gewisser Weise auch einer konzeptuellen Modellierung entspricht und die konzeptuellen Modelle in Anwendungsbereichen verwendet werden, welche im Abschnitt 2.1.3 mit der Beschreibung des XML-Einsatzes bereits erwähnt und auch in Abbildung 2.4 dargestellt wurden.

3.1 Softwarequalität

Die Softwarequalität läßt sich nur durch Softwaremessung bestimmen. Die Erkenntnis darüber existiert zwar schon länger, aber trotzdem ist die Softwaremessung im Bereich der Softwaretechnik noch eine junge Disziplin und gegenwärtiger Forschungsgegenstand. Ein Überblick über die Softwaremessung wird diesen Abschnitt einleiten.

Der Begriff der Softwarequalität wurde bereits in diesem Kapitel in der Motivation angelehnt an die DIN 8402 zitiert. Um die Softwarequalität bestimmen zu können, benötigt man konkrete Qualitätsmerkmale, die die Softwarequalität näher charakterisieren. Die Qualitätsmerkmale werden nach dem Abschnitt der Softwaremessung auf der Grundlage des ISO 9126 Standards [ISO91a] vorgestellt. Die ISO 9126 klassifiziert die Softwarequalität zuerst in sechs Qualitätsmerkmale, die dann in weitere weitgehend disjunkte Teilmerkmale gegliedert sind. Ein Vorteil ist hier die unabhängige Klassifizierung in allgemeingültige Qualitätsmerkmale. In anderen Normen bzw. auch Qualitätsmodellen wird die Qualität meist aus einer speziellen Sicht – z.B. Entwickler, Benutzer, ..., Prozeß, Produkt, ... – betrachtet, mit dem Sinn einer gezielten Qualitätssicherung.

Da zunächst nur die allgemeinen Qualitätsmerkmale beschrieben werden, sollen im folgenden Abschnitt Softwaremetriken und einige spezielle Klassifizierungen betrachtet werden. Softwaremetriken beschreiben quantitativ ein bestimmtes qualitatives Softwaremerkmal unter Berücksichtigung einer speziellen Sicht auf die Software. Beispielsweise sei hier erwähnt, daß es eine Menge von speziellen objektorientierten Softwaremaßen gibt, die nicht gleichermaßen zur Bewertung derselben Qualitätsmerkmale nichtobjektorientierter Programmiersprachen verwendet werden können. Das soll heißen, daß Softwaremetriken nicht wie die Qualitätsmerkmale als allgemeingültig betrachtet werden können. Aus diesem Grund sollen verschiedene Klassifizierungen bzw. Betrachtungsweisen für Softwaremetriken vorgestellt werden. Da bereits eine sehr große Menge an Softwaremetriken existiert, wird die Verwendung von bereits entwickelten und eventuell auch akzeptierten Metriken einer Neuentwicklung im Rahmen dieser Arbeit vorgezogen. Dies soll aber erst im Kapitel 4 im Zusammenhang mit der Entwicklung von DTD-Metriken betrachtet werden.

3.1.1 Softwaremessung

Messungen, daß heißt deren Durchführung und die Interpretation der Maße, sind auf anderen Gebieten wie z.B. der Physik, der Medizin und sogar im Alltag allgegenwärtig und gar nicht wegzudenken. Die Notwendigkeit der Messung auch im Bereich der Softwaretechnik wird durch folgende Zitate bekräftigt:

- "To measure is to know." Clerk Maxwell
- "You cannot control what you cannot measure." Tom DeMacro
- "You cannot predict what you cannot measure." Norman N. Fenton
- "Measurement is an excellent abstraction mechanism for learning what works and what doesn't." Victor Basili
- "A science is a mature as its measurement tools." Louis Pasteur

Auch wenn das Gebiet der Softwaremetrie noch ziemlich jung im Gegensatz zur Softwaretechnik ist, soll das nicht heißen, daß bisher wenige Standards, Qualitätsmodelle, Metriken etc. existieren. Einerseits bedingen die unterschiedlichsten Softwareaspekte geradezu eine Vielzahl z.B. an Qualitätsmodellen und auch Metriken. Andererseits erfordern die unterschiedlichen Zielausrichtungen bei der Messung unterschiedliche Meßstrategien auf der anderen Seite. Hierin besteht auch das Problem bei der Softwaremessung. Trotz unterschiedlicher Qualitätsmodelle, Metriken und Meßstrategien sollten die Meßergebnisse vergleichbar sein. Dies ist leider nur bedingt möglich, wie beispielsweise bei den Komplexitätsmaßen aufgrund der unterschiedlichen Sprachparadigmen und auch der verschiedenen Ansichten, was komplex ist. Außerdem kommt noch erschwerend hinzu, daß es kein festes Modell bzw. keine feste Vorgehensweise bei der Softwaremessung gibt. Die Softwaremessung muß den erforderlichen Gegebenheiten – z.B. Sprachparadigmen, konzeptuelles Modell, Zielausrichtung – angepaßt werden. Aus diesem Grund soll die Softwaremessung allgemein vorgestellt werden. Eine ausführliche Beschreibung zur Einführung in die Messung ist in [Fen91] zu finden. In diesem Buch definiert Norman Fenton den Prozeß des Messung folgendermaßen:

> "Measurement is the process by which numbers or symbols to attributes of entities in real world in such a way as to describe them according to clearly defined rules."

Das bedeutet für die Softwaremessung, daß geeignete Attribute der zu messenden Entities bestimmt werden müssen. In [Fen91] werden im Zusammenhang mit der Klassifikation von Softwaremaßen die Komponenten (Entities, Attribute) der Softwaremessung beschrieben. Ein Entity ist dabei das Meßobjekt und das Attribut ist die zu messende Eigenschaft. Diese Klassifikation wird im Abschnitt 3.1.3 noch näher erläutert. Um das Attribut bewerten, vergleichen oder einschätzen zu können, muß es quantitativ bewertbar sein. Dies wird erreicht durch ein Maß bzw. Metrik, die Norman Fenton folgendermaßen definiert:

> "A measure is an empirical objective assignment of a number (or symbol) to an entity to characterize a specific attribute."

Die objektive Zuordnung einer Zahl (oder eines Symbols) zu einem Entity wird als Maß bezeichnet. Dabei charakterisiert das Maß das spezifische Attribut des Entities. Das Maß ist hier nicht als eine Zahl zu sehen, sondern als eine Abbildung des Attributes in einen Wertebereich. Dabei muß das Maß der Abbildungsvorschrift entsprechen. Bei den Maßen unterscheidet man zwischen einem empirischen Maß und einem numerischen Maß. Empirische Maße sind z.B. Evaluierungen der Qualität, der Kosten etc. und werden durch empirische Relationen beschrieben, wie z.B. "ist größer, ist gleich, ist kleiner" oder ähnliche. Da die empirische Relation nicht direkt darstellbar ist, muß sie auf ein numerisches System abgebildet werden. Die numerischen Maße sind dagegen direkt darstellbar. Entsprechend der Vergleichbarkeit oder Ordnung (Reihenfolge z.B. totale Ordnung) der Werte lassen sich Maße auf einer Nominal-, Ordinal-, Intervall-, Verhältnis- oder Absolutskala abbilden. Bei der Definition von einem Maß sind nach [Dum92] die folgenden Probleme zu lösen:

Repräsentanzproblem
Ist die Forderung nach der Möglichkeit einer zahlenmäßigen Darstellung (als Meßwert bzw. numerisches Relativ) für die gemessene Eigenschaft.

Eindeutigkeitsproblem
Ist die Forderung von einem gleichartigen Verhalten bei mehreren Repräsentationen einer Eigenschaft.

Bedeutungsproblem
Ist die Forderung der Beibehaltung der richtigen Bedeutung im Sinne der Interpretation trotz zulässiger Meßwerttransformation.

Skalierungsproblem
Ist die Art und Weise der Bestimmung der jeweiligen Skalierung für das Maß, wobei die oben erwähnten Skalentypen zur Auswahl stehen.

Weiterhin werden in [Dum92] informale Anforderungen von Softwaremaßen aufgezählt. Diese sind wie folgt in Hauptgütekriterien und Nebengütekriterien eingeteilt.

Hauptgütekriterien

Objektivität	–	durch möglichst geringe subjektive Einflüsse
Verläßlichkeit	–	bei der Ermittlung des jeweiligen Maßes durch Vermeidung von Zufälligkeiten und zeitlich bedingte Einflüsse
Validität	–	um eine "allgemeine" Gültigkeit zu erreichen

Nebengütekriterien

Normierung	–	um durch eine Skalierung eine Zuordnung zu verbalen Bewertungen zu erreichen
Vergleichbarkeit	–	aufbauend auf einer Normierung sollte korrelatives Maßverhalten gewährleistet werden
Ökonomie	–	Kosten für die Softwaremessung sollten unter dem durch die Nutzung dieses Maßes zu erwartenden Nutzen liegen
Nützlichkeit	–	um wirklich praktische Bedürfnisse zu erfüllen

Nachdem die allgemeine Messung, Probleme der Maßdefinition und Anforderungen an die Maße beschrieben worden sind, soll jetzt noch eine allgemeine Durchführung der Softwaremessung erläutert werden. Wie bereits erwähnt, gibt es eine Vielzahl von unterschiedlichen Vorgehensmodellen bzw. auch Standards – z.B. ISO 9000, IEEE 1061, GQM, CMM, FCM – die auch zum Teil die Durchführung der Softwaremessung vorstellen. Diese Modelle bzw. Standards stellen die Softwaremessung meist aus einem bestimmten Blickwinkel dar und können selten direkt angewendet werden. Eine allgemeine Vorgehensweise (siehe Abbildung 3.1) wird in dem Standard IEEE 1061 [IEE93] definiert und soll kurz vorgestellt werden.

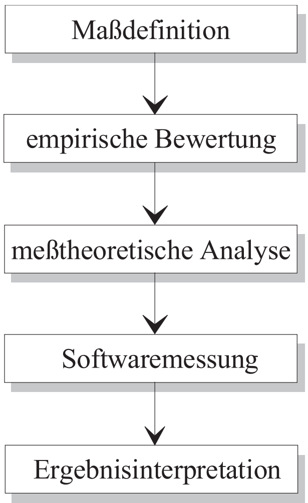


Abbildung 3.1: Vorgehensmodell bei der Softwaremessung durch IEEE 1061 (aus [Dum01])

Die in der Abbildung 3.1 dargestellte Vorgehensweise ist sehr allgemein. Sie berücksichtigt zunächst nicht die Betrachtung spezieller Softwaremeßstrategien – wie die Evaluierungen, Merkmalsabschätzungen, indirekte und direkte Softwaremessung – durch optionale Teilschritte. Aufgrund einiger

Meßstrategien entfallen eventuell Teilschritte oder sie bedingen spezielle Ausprägungen der einzelnen Meßschritte. Die einzelnen Meßstrategien werden nach der Beschreibung der Teilschritte vorgestellt. Desweiteren wird in dieser Darstellung auch nicht angedeutet, daß durch einen Rücksprung zur Maßdefinition die Modifizierung eines Maßes möglich ist. Dies kann z.B. nach der Validation, also hier bei der meßtheoretischen Analyse notwendig sein und wird in anderen Darstellungen auch nach der Ergebnisinterpretation ermöglicht.

Maßdefinition
: Der eigentlichen Maßdefinition geht eine informale Analyse des Softwaremeßzieles voraus. In dieser Analyse werden die Meßziele, z.B. durch Qualitätsmerkmale, das Meßobjekt (Entity) und die Attribute, welche die Meßziele charakterisieren, beschrieben. Das Meßobjekt ist im allgemeinen bekannt. Das eigentliche Problem besteht hier in der Zuordnung der Meßziele zu den charakterisierenden Attributen bzw. auch umgekehrt. Danach erfolgt die eigentliche Maßdefinition. Dabei besteht die Möglichkeit der Auswahl eines vorhanden Maßes bzw. auch dessen Modifikation oder die Neuentwicklung eines geeigneten Maßes. Die Verwendung eines passenden Maßes sollte dabei der Neuentwicklung vorgezogen werden.

empirische Bewertung
: Der Grund der empirischen Bewertung wurde bereits zuvor beschrieben. Dieser Punkt könnte eigentlich auch zur Maßdefinition gezählt werden, da es sich hier auch um die Definition eines Maßes, nämlich des empirischen Maßes, handelt. Dieser Meßschritt entfällt, wenn bei der Softwaremessung keine empirischen Maße verwendet werden, so z.B. bei der Merkmalsabschätzung oder auch der direkten Softwaremessung. Wenn bei der Softwaremessung nur empirische Maße verwendet werden, wie z.B. bei der Evaluierung, entfällt die Maßdefinition. Einer empirischen Bewertung geht ähnlich der Maßdefinition eine Analyse voraus bzw. fällt diese mit der empirischen Bewertung zusammen.

meßtheoretische Analyse
: In der meßtheoretischen Analyse soll die Validität des Maßes gezeigt werden, wobei die Meßtheorie (siehe [Zus98]) zugrunde gelegt wird. Es gibt hierbei aufgrund der unterschiedlichen Maße auch unterschiedliche Validitätsformen, die prinzipiell zeigen sollen, was das Maß wirklich mißt. Hierbei sind natürlich die zuvor genannten informalen Probleme der Maßdefinition und auch die Anforderungen an die Maße von Bedeutung. Die Validation eines Maßes kann zur Modifikation führen, aufgrund dessen das Maß erneut definiert werden muß. Ausführliche Beschreibungen zur Validation von Softwaremaßen sind in [Fen91] und [Dum92] zu finden.

Softwaremessung
: Durch die Softwaremessung werden die Eigenschaften (Attribute) des Meßobjektes (Entity) ermittelt. Bei der Durchführung der Messung gibt es unterschiedliche Verfahren, die vom Meßobjekt oder von der zu messenden Eigenschaft abhängen. Die Messung kann manuell, z.B. durch Codemusterung etc. oder auch automatisch durch Tools durchgeführt werden. Weiterhin kann die Messung statisch sein, daß heißt ohne die Ausführung der Software selbst oder auch dynamisch durch Ausführung z.B. des Programmcodes.

Ergebnisinterpretation
: Die Ergebnisinterpretation erfolgt durch den Nutzer der Softwaremessung. Hierbei interpretiert der Nutzer seinen Standpunkt bezüglich des Maßes, beispielsweise den Grad der Komplexität eines Programmes. Bei der Ergebnisinterpretation kann aufgrund der Einschätzung des Nutzers eine Maßmodifikation notwendig werden oder sich herausstellen, daß sich ein Maß für den Nutzerstandpunkt nicht eignet. Eine ausführliche Beschreibung von Maßen, welche auch die Ergebnisinterpretation unterstützen, sind z.B. in [Zus91] zu finden.

Nach der Beschreibung der allgemeinen Softwaremessung sollen jetzt die zuvor genannten speziellen Softwaremeßstrategien vorgestellt werden. Diese berücksichtigen durch spezielle Ausprägungen bzw. auch durch Weglassen einzelner Teilschritte eine konkretere Berücksichtigung unterschiedlicher Aspekte bei der Messung (siehe auch [Dum01]).

Evaluierungen
Die Evaluierungen basieren hauptsächlich auf Klassifikationen von einem oder auch mehreren Merkmalen, die durch empirische Bewertungen – z.B. "sehr gut, gut" – meist in Tabellenform dargestellt werden. Kennzeichnend für diese Kategorie der Softwaremessung sind das Aufstellen und Ausfüllen von Fragebögen, wobei die eigentliche Messung manuell vorgenommen wird. Aufgrund der Art der Messung entfällt der Teilschritt der Maßdefinition.

Merkmalsabschätzungen
Die Merkmalsabschätzungen erfolgen auf der Basis einer vorgegebenen Schätzformel, beispielsweise Schätzung des Entwicklungsaufwandes durch das Function-Point-Modell oder des Kostenmodell COCOMO. Diese Verfahren setzen eine große Erfahrung in Bezug auf die zu schätzenden Merkmale voraus und erfordern meist eine rückwirkende Modifikation der Schätzformel, z.B. aufgrund von spezifischen Anpassungen an die sich ändernden Voraussetzungen.

Indirekte Softwaremessung
Indirekte Softwaremessung wird auch modellbezogene Softwaremessung genannt. Die interessierenden Attribute werden über ein Modell oder über eine abgeleitete Zwischensprache – Metasprache, Syntaxtabelle – gemessen. Der Maßdefinition geht hier eine Modellbildung des Meßobjektes voraus und folgt eine empirische Bewertung. Diese Attribute – z.B. Änderbarkeit, Übertragbarkeit – lassen sich beispielsweise durch Strukturmetriken (Darstellung als Hierarchiebaum), Komplexitätsmetriken (für Datenfluß-, Steuerflußmessung) etc. messen.

Direkte Softwaremessung
Die direkte Softwaremessung erfolgt unmittelbar am Meßobjekt. Beispielsweise können Umfangsmaße, Programmausführungsmaße etc. direkt bestimmt werden. Eine empirische Bewertung kann bei dieser Kategorie der Softwaremessung entfallen, da die Attribute sich numerisch charakterisieren lassen.

Der Überblick über die Softwaremessung sollte die Durchführung von Softwaremessungen und die damit verbunden Aspekte veranschaulichen. Dabei wurde gezeigt, wie vielfältig die bei der Softwaremessung zu berücksichtigenden Aspekte sind, die einerseits unterschiedliche Standards und Modelle bedingen, aber andererseits die Auswahl eines geeigneten Modells schwierig machen. Aufgrund der Zielstellung (siehe Analyse bei der Maßdefinition) und auch Zielausrichtung durch den Nutzer (siehe Nutzersicht in Abschnitt 3.1.3), also denjenigen der an der Messung interessiert ist, wird meistens eine zielorientierte Anpassung der Softwaremessung notwendig sein.

3.1.2 Qualitätsmerkmale von Software nach ISO 9126

Die ISO9126 [ISO91a] legt Qualitätsmerkmale von Softwareprodukten fest. Die Qualitätsmerkmale sind Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. Jedes dieser Merkmale wird in weitere spezifischere Merkmale untergliedert. In Abbildung 3.2 sind die Hauptmerkmale mit ihren Submerkmalen entsprechend der ISO 9126 hierarchisch gegliedert.

In einigen Beschreibungen werden die Qualitätsmerkmale manchmal auch anders strukturiert. Dies sind aber eigentlich nur Umstrukturierungen der allgemeinen Qualitätsmerkmale nach besonderen Gesichtspunkten, wie z.B. für ein spezielles Qualitätsmodell. Barry Boehm unterteilt in [Boe80] die Qualitätsmerkmale zunächst nur nach Funktionalität und Wartbarkeit und ordnet diesen beiden Kategorien dann die wünschenswerten Eigenschaften zu. Die Priorisierung einiger Qualitätsmerkmale läßt sich durch die Anforderungen der Software erklären. Nicht jede Software wird jedes Qualitätsmerkmal gleichzeitig voll unterstützen können. Eine sehr effiziente Software wird vielleicht in Assembler geschrieben werden, aber widerspricht dann den Kriterien der Verständlichkeit, Änderbarkeit und Übertragbarkeit. Oder ein auf unterschiedliche Betriebssysteme übertragbares Programm kann das Effizienzkriterium nicht gleichzeitig erfüllen. Die Qualitätsmerkmale widersprechen sich teilweise bzw. sind nicht gleichzeitig vollständig realisierbar. Die Auswahl der wichtigen oder auch die Priorisierung einiger Qualitätsmerkmale ist abhängig von der Anwendung. Im folgenden sollen nun die allgemeinen Qualitätsmerkmale unabhängig von einem speziellen Anwendungshintergrund betrachtet werden.

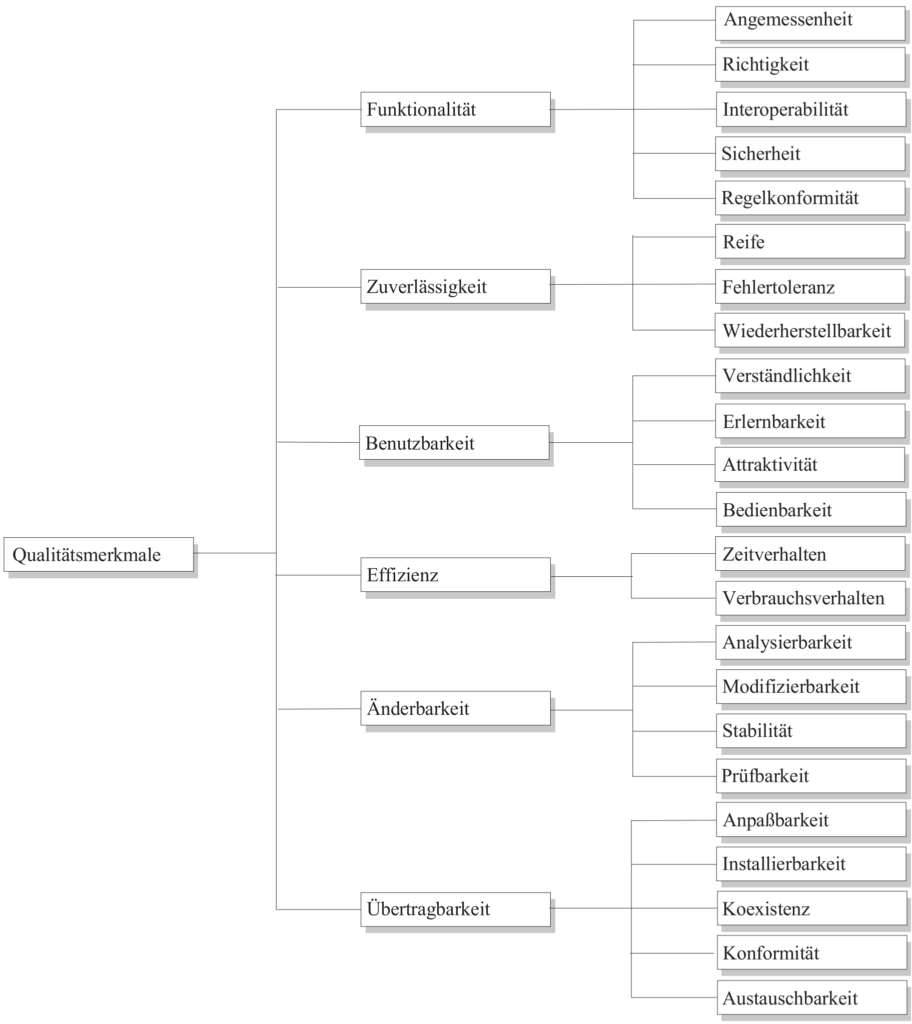


Abbildung 3.2: ISO 9126 Qualitätsmerkmale für Software

Funktionalität
Die Funktionalität von Software wird durch Funktionen mit festgelegten Eigenschaften beschrieben. Diese Funktionen müssen die definierten Anforderungen erfüllen. Weiterhin wird die Funktionalität beschrieben durch die Merkmale Angemessenheit, Richtigkeit, Interoperabilität, Sicherheit und Regelkonformität.

Angemessenheit	–	bedeutet für Software, daß sie genau die Funktionen enthält, die für die Erfüllung der Aufgabe erforderlich sind
Richtigkeit	–	beschreibt den Grad der Übereinstimmung (Exaktheit) der Software mit den Vorgaben
Interoperabilität	–	beschreibt die Fähigkeit, mit anderen (vorgegebenen) Systemen zusammenzuarbeiten
Sicherheit	–	wird bestimmt durch die Fähigkeit, beabsichtigte oder unbeabsichtigte unautorisierte Zugriffe auf Software (Programme, Daten) zu verhindern
Regelkonformität	–	Fähigkeit der Software regelkonform zu sein, sofern Regeln und Normen vorgegeben sind

Zuverlässigkeit

Die Zuverlässigkeit von Software wird dadurch bestimmt, daß ihr Leistungsniveau unter den festgelegten Bedingungen in einem ebenfalls festgelegtem Zeitraum eingehalten wird. Weiterhin kann die Zuverlässigkeit in die Qualitätsmerkmale Reife, Fehlertoleranz und Wiederherstellbarkeit unterteilt werden.

Reife	–	beschreibt die Häufigkeit von Ausfällen durch Fehler in der Software
Fehlertoleranz	–	bezeichnet die Robustheit der Software gegenüber falschen Eingaben
Wiederherstellbarkeit	–	Fähigkeit der Software, aus einem undefinierten Zustand (Fehler) über einen definierten Zustand wieder in Betrieb genommen zu werden

Benutzbarkeit

Die Benutzbarkeit beschreibt den benötigten Aufwand zur Benutzung der Software, der durch eine festgelegte Benutzergruppe erbracht werden muß, um mit dieser Software zu arbeiten. Problematisch bei diesem Qualitätsmerkmal ist die jeweils verschiedene subjektive Auffassung des Benutzers, ob die Software einfach zu nutzen ist. Dieses Qualitätsmerkmal ist deswegen auch schwer zu messen. Weitere charakteristische Merkmale sind die Verständlichkeit, Erlernbarkeit, Attraktivität und Bedienbarkeit.

Verständlichkeit, Erlernbarkeit, Bedienbarkeit	–	Diese Qualitätsmerkmale sollten vom Benutzungsstandpunkt aus möglichst einfach oder am besten intuitiv erfaßbar sein. Dies ist natürlich stark abhängig von der Komplexität der Software und betrifft unter anderem die Gestaltung der Benutzungsoberfläche, die konsistente Begriffswahl im Programm und in der Dokumentation sowie die Unterstützung des Benutzers z.B. durch eine Onlinehilfe und eine gute Programmbeschreibung.
Attraktivität	–	Ist vom Benutzungsstandpunkt aus dadurch bestimmt, die Software gegenüber anderen vorzuziehen. Bei dieser Beurteilung muß die Software in einigen Qualitätsmerkmalen – wie z.B. Funktionalität – übereinstimmen, da hier sonst kein reeller Vergleich möglich ist.

Effizienz

Die Effizienz beschreibt das Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen. Hierbei sind das Zeitverhalten und Verbrauchsverhalten in Bezug auf die Ressourcen von besonderer Bedeutung.

Zeitverhalten	–	bestimmt durch anforderungsmäßige Einhaltung vom Durchsatz, Antwortzeit- und Verarbeitungszeitverhalten
Verbrauchsverhalten	–	bestimmt durch Ressourcenverbrauch (Anzahl der verwendeten Ressourcen und Dauer der Verwendung dieser Ressourcen)

Änderbarkeit

Das Qualitätsmerkmal Änderbarkeit wird oft auch als Wartbarkeit bezeichnet. Änderungen werden aufgrund von unterschiedlichsten Gründen durchgeführt, wie z.B. Korrekturen, Verbesserungen oder Anpassungen an Veränderungen der Umgebung, Verbesserungen oder Anpassungen der funktionalen Spezifikation etc.. Hat die Software einen stabilen Zustand erreicht, so werden Änderungen nicht mehr so häufig durchgeführt. Hier ist der Aufwand zur Durchführung der Änderung an der Software von Interesse und wird weiterhin durch die Merkmale Analysierbarkeit, Modifizierbarkeit, Stabilität und Prüfbarkeit bestimmt. Diese Qualitätsmerkmale sind besonders abhängig von einer guten Softwaredokumentation sowie vom gut kommentierten und lesbaren Quelltext.

Analysierbarkeit	–	wird durch den Aufwand zur Lokalisierung von Mängeln, Fehlern oder zu modifizierenden Programmteilen bestimmt
Modifizierbarkeit	–	ist der Aufwand für Fehlerbeseitigungen, Verbesserungen oder Umgebungsänderungen
Stabilität	–	wird bestimmt durch die Gefahr von unerwarteten Auswirkungen durch Änderungen (z.B. durch Seiteneffekte, unkommentierte Programmiertricks)
Prüfbarkeit	–	ist der Aufwand zur Überprüfung modifizierter Software (z.B. geringe Schachtelungstiefe von Funktionen oder auch Aufruftiefe von Programmen)

Übertragbarkeit

Die Übertragbarkeit oder auch Portabilität beschreibt die Eignung der Software, von einer Umgebung in eine andere übertragen zu werden. Die Umgebung kann hier eine organisatorische Umgebung, Software- oder Hardware-Umgebung darstellen. Eine speziellere Klassifizierung dieses Aspektes wird durch die Qualitätsmerkmale Anpaßbarkeit, Installierbarkeit, Koexistenz, Konformität und Austauschbarkeit erreicht.

Anpaßbarkeit	–	an unterschiedliche Umgebungen
Installierbarkeit	–	ist der Aufwand zur Installation in spezieller Umgebung
Koexistenz	–	ist die Beeinflussung der Software auf und durch die Umgebung
Austauschbarkeit	–	ist die Ersetzbarkeit in Größe und Funktion von Software durch andere Software

Diese allgemeinen Qualitätsmerkmale sind im großen und ganzen in den anderen Standards oder auch Qualitätsmodellen in ähnlicher oder anderer Zusammenstellung wiederzufinden. Da diese Qualitätsmerkmale meßbar und bewertbar sein müssen, ist es daher denkbar, die Softwaremetriken direkt den Qualitätsmerkmalen, wie in Abbildung 3.3 dargestellt, zuzuordnen. Dies ist prinzipiell möglich, aber berücksichtigt nicht die Tatsache, daß Softwaremetriken nicht wie die Qualitätsmerkmale als allgemeingültig gesehen werden können. Softwaremetriken sind aus diesem Grund auch eher in sogenannten Metriksammlungen – wie z.B. der "Metric suite for Object-oriented Design" [CK94] – zusammengefaßt. Dabei werden nicht unbedingt alle Qualitätsmerkmale bewertet bzw. gleich priorisiert, aber die Besonderheiten des Entwurfsmodells und die Spracheigenschaften berücksichtigt. Die Abbildung 3.3 soll deshalb lediglich den Eindruck verstärken, daß es zu den Qualitätsmerkmalen eine große Auswahl an vorhandenen Qualitätsmetriken gibt, die aber nicht wie diese den jeweiligen Qualitätsmerkmalen zugeordnet werden. Man kann aber allgemein sagen, daß zu jedem Qualitätsmerkmal mehrere Metriken existieren und umgekehrt möglich ist, daß eine Metrik mehreren Qualitätsmerkmalen zugeordnet werden kann. Eine Zusammenstellung von ca. 1600 Softwaremetriken haben Horst Zuse und Karin Drabe in ihrem "Measure-Information-System" (siehe [ZD01]) realisiert, wobei die Metriken und ihr Einsatz beschrieben wurden und die Metriken nach unterschiedlichen Kriterien klassifiziert sind.

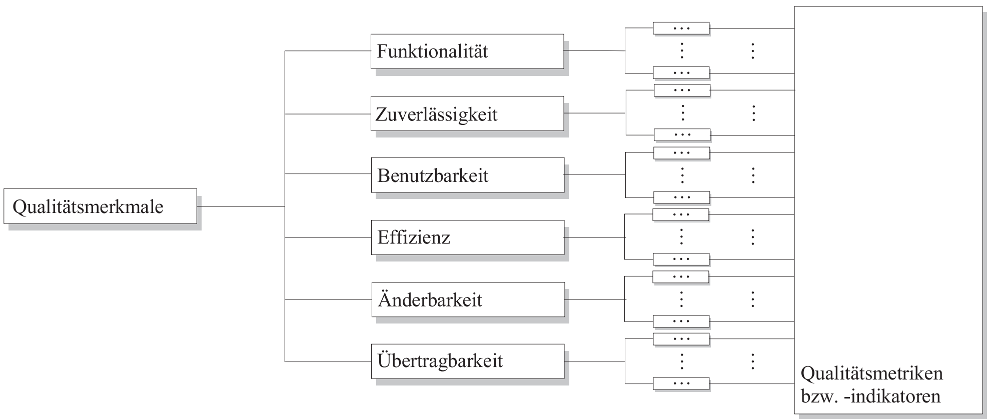


Abbildung 3.3: Qualitätsmerkmale und Qualitätsmetriken

Die Einteilungen von Softwaremetriken nach besonderen Kriterien werden im nächsten Abschnitt beschrieben. Spezielle Metriksammlungen, die eventuell auch bei der Metrikdefinition für DTDs von Bedeutung sein könnten, werden im zweiten Abschnitt dieses Kapitels beschrieben.

3.1.3 Klassifikationen von Softwaremetriken

In diesem Abschnitt sollen unterschiedliche Klassifikationsmöglichkeiten von Softwaremetriken sowie einige charakteristische Merkmale beschrieben werden. Dies soll dazu dienen, einen besseren Überblick über die bereits existierenden Softwaremetriken zu bekommen, wobei aber die Metriken hier noch nicht im einzelnen vorgestellt werden sollen. Weitere Klassifikationen und Beschreibungen zu Metriken sind in [Zus98], [ZD01] und auch [Zus91] zu finden.

Wie bereits am Ende des letzten Abschnittes erwähnt, gibt es eine Vielzahl an Softwaremetriken. In der Abbildung 3.3 ist bereits zuvor dargestellt worden, daß man die Softwaremetriken den Qualitätsmerkmalen zuordnen kann bzw. auch umgekehrt. Dies ist eine einfache und nicht eindeutige Klassifikationsmöglichkeit. Da jede Softwaremetrik mindestens ein Qualitätsmerkmal beschreibt, ist die Zuordnung zunächst einmal plausibel. Nicht berücksichtigt werden in dieser Klassifizierung spezielle Softwareaspekte, die für die Anwendung bzw. Auswahl der geeigneten Metriken wichtig sind. Bevor einige spezielle Klassifikationen vorgestellt werden, sollen zunächst einige der vielfältigen Einsatzgebiete von Softwaremetriken beschrieben werden (siehe auch [Fen91]).

Kosten- und Aufwandsbewertungsmodelle

Der Grund für diese Modelle ist die Notwendigkeit einer möglichst frühen Einschätzung von Kosten und Aufwand von einem Projekt. Von diesen Bewertungsmodellen werden bereits ab den ersten Projektphasen Ergebnisse erwartet, die z.B. den Projektumfang abschätzen durch das LOC-Maß (COCOMO-Modell von Barry Boehm) oder durch die Anzahl von "Function Points" (Function Point Modell von IBM).

Produktivitätsmodelle

Das Ziel der Produktivitätsmodelle ist die Einschätzung der Produktivität bei der Softwareherstellung. Hierbei wird davon ausgegangen, daß die Produktivität eine Funktion von Werten – messbare qualitative und quantitative Qualitätsmerkmale – und Kosten – Personal und Ressourcen – ist, die geeignet bestimmt werden müssen, um ein Maß für die Produktivität zu erhalten.

Datensammlungen

Datensammlungen sind hier eher als Mittel zum Zweck anzusehen. Viele Maße werden nicht direkt gemessen. Sie setzen sich entweder aus mehreren einzelnen Werten, die in einem bestimmten Zeitpunkt oder auch über einen Zeitraum gemessen werden, zusammen. Daher ist es beispielsweise für statistische Bewertungen nicht nur notwendig, gültige Maße zu verwenden, sondern auch gültige Daten zu sammeln.

Qualitätsmodelle

Die Notwendigkeit von Qualitätsmodellen wurde bereits in der Motivation dieses Kapitels und im vorherigen Abschnitt erwähnt. Nicht alle in den Qualitätsstandards – z.B. ISO 9126 – aufgeführten Qualitätsmerkmale sind gleichzeitig uneingeschränkt erfüllbar. Dies hängt unter anderem mit den Anforderungen an die Software zusammen. Aus diesem Grund gibt es auch kein allgemein gültiges Qualitätsmodell mit allen bekannten Qualitätsmerkmalen, sondern auf spezielle Softwareanforderungen zugeschnittene Modelle. In den Qualitätsmodellen werden die geforderten Qualitätskriterien festgelegt und Softwaremaße angegeben, die die Qualitätskriterien geeignet quantifizieren. Als Beispiel sei hier das Factor-Criteria-Metric Modell (kurz FCM-Modell) von McCall genannt, das diesem Aufbau entspricht.

Zuverlässigkeitsmodelle

Die Zuverlässigkeit ist ein Teilmerkmal der Softwarequalität und sollte von jeder Software erfüllt werden. In den Zuverlässigkeitsmodellen werden meist Maße beschrieben, die Fehlerhäufigkeit bewerten oder auch durch statische Modelle abschätzen, wie z.B. durch das Maß MTBF (mittlere Zeit zwischen Fehlern).

Leistungsbewertung

Ähnlich wie bei den Zuverlässigkeitsmodellen, wird bei der Leistungsbewertung auch nur ein Teilmerkmal (Effizienz) der Softwarequalität betrachtet. Dabei werden zum einen externe wahrnehmbare Parameter – z.B. das Antwortzeitverhalten, Durchsatz – bewertet. Zum anderen werden interne Parameter betrachtet, die die Effizienz beeinflussen, wie z.B. die berechenbare und algorithmische Komplexität.

Struktur- und Komplexitätsbewertung
Die Strukturmetriken basieren auf einer konstruktionsbezogenen Analyse der Software, wie z.B. von konzeptuellen Modellen (ER, UML, OOA, OOD, etc.) oder des Quellcodes. Da strukturelle Modelle bereits in den ersten Phasen der Softwareentwicklung vorhanden sind, kann die Strukturbewertung frühzeitig durchgeführt werden. Im Zusammenhang mit der Strukturbewertung wird oft auch von struktureller Komplexität gesprochen. Auch wenn Strukturmetriken teilweise getrennt klassifiziert werden, so kann man sie als Komplexitätsmetriken betrachten, die aber unbedingt geeignet interpretiert werden müssen. Beispielsweise bewerteten die Quellcodemetriken von Halstead oder auch McCabe (siehe [Dum92]) die strukturelle Komplexität. Durch die Komplexitätsbewertung wird versucht, die Kompliziertheit der Software als Maß festzulegen. Die Komplexitätsbewertung betrifft die Qualitätsmerkmale Änderbarkeit, Effizienz und auch Zuverlässigkeit.

Nachdem einige der Einsatzmöglichkeiten beschrieben wurden, sollen nun Klassifikationsmöglichkeiten vorgestellt werden. Ausgehend vom Lebenszyklus der Software gibt es zwei unterschiedliche Aspekte, nach denen die Metriken eingeteilt werden können. Zum einen sind es die unterschiedlichen Nutzersichten und zum anderen die jeweiligen Phasen des jeweiligen Lebenszyklus-Modells, z.B. entsprechend dem Wasserfallmodell von der Analyse- bis zur Nutzungsphase. Der zweite Aspekt wird in der Klassifikation von Fenton noch einmal vorkommen und soll hier erst einmal nicht weiter interessieren, da diese nachfolgend noch beschrieben wird. Aus der Nutzersicht werden nicht alle Qualitätsmerkmale gleichzeitig interessieren, so daß sich entsprechend der Nutzersicht unterschiedliche Qualitätszielausrichtungen ergeben.

Entwickler
Der Entwickler ist im Lebenszyklusmodell meist bis auf die Wartungsphase in allen Phasen beteiligt. Phasen-abhängig sind unterschiedliche Kriterien wichtig, am meisten aber Phasen-übergreifende Metriken, die die Verständlichkeit und die Zuverlässigkeit des Softwareentwicklungsprozesses beschreiben.

Manager
Der Projektleiter ist ähnlich wie der Entwickler an fast allen Phasen beteiligt, aber eher an Kosten- und Aufwandsmetriken interessiert, die Projektkosten und -fortschritt beschreiben.

Kunden
Der Anwender ist hauptsächlich an Kriterien wie Benutzbarkeit und Funktionalität interessiert, was aber nicht heißen soll, daß andere Merkmale durch ihn nicht auch priorisiert werden können.

Wiederverwender
Nach dem eigentlichen Softwareentwicklungsprozeß muß die Software gepflegt und gewartet werden, wobei hier die Qualitätsmerkmale Änderbarkeit und Übertragbarkeit die wichtigste Rolle spielen.

Weiterhin lassen sich die Metriken nach der Art und Weise der Ermittlung einteilen. Dies ist eher aus der Sicht der Softwaremessung eine pragmatische Einteilung.

direkte versus indirekte Metriken
Direkte Metriken sind durch die Messung einer Softwareeigenschaft direkt darstellbar, z.B. LOC-Maß. Indirekte (abgeleitete oder modellbezogene) Metriken sind entweder von mehreren Softwareeigenschaften abhängig oder werden durch geeignete Interpretation (z.B. empirische Bewertungen) aus einer Softwareeigenschaft abgeleitet.

subjektive versus objektive Metriken
Subjektive Metriken sind schwer quantifizierbar, wie z.B. die Attraktivität oder auch andere Benutzbarkeitsmerkmale. Diese Metriken werden verwendet, um Meinungen über die Qualität von Software zu erhalten und können beispielsweise durch Umfragen und Antwortsklassifikationen (sehr gut, gut, ...) bestimmt werden. Die Ergebnisse der Messung werden subjektiv beeinflußt, sind aber im Gegensatz zu objektiven Metriken leichter interpretierbar. Objektive Metriken sind dagegen quantifizierbar und leicht meßbar.

prediktive versus deskriptive Metriken
Ein prediktives Maß kann nicht gemessen werden, sondern wird geschätzt. Die deskriptiven Maße lassen sich als Eigenschaften des Softwareproduktes oder des Entwicklungsprozesses messen, bzw. sind allgemein aus der Analyse des Software-Lebenszyklus bestimmbar.

statisch versus dynamische Metriken
Statische Metriken sind dokumentationsbezogene Maße. Dokumentationsbezogen bedeutet hier, daß die Maße aus den Dokumenten – z.B. Quellcode, Dokumentation, konzeptuelle Modelle – die während des Entwicklungsprozeß der Software entstehen, bezogen werden können. Dynamische Metriken sind laufzeitbezogene Maße. Zu diesen Metriken zählen z.B. verschiedene Zeitaufwandsmaße, Komplexitätsmaße (z.B. Zweigüberdeckung, um den Testaufwand abzuschätzen) etc., die das Laufzeitverhalten quantifizieren.

Phasenmetriken versus Globalmetriken
Entsprechend den Software-Lebenszyklusmodellen lassen sich die anwendbaren Metriken in phasenbezogene und phasenübergreifende einteilen. Die Globalmetriken fassen mehrere Phasen zusammen und sollen die strategische Projektplanung unterstützen. Phasenmetriken werden für die taktische Unterstützung der jeweiligen Phase verwendet.

Eine sehr umfangreiche und oft benutzte Klassifikation wird von Norman Fenton in [Fen91] vorgestellt. Der Ausgangspunkt dieser Klassifikation sind die zu messenden Objekte (Entities) im Softwareentwicklungsprozeß. Dieser Entity-Klassifikation werden im zweiten Schritt die wünschenswerten Eigenschaften (Attribute) zugeordnet, die gemessen werden sollen. Die Entities werden zunächst grob in Prozesse, Produkte und Ressourcen unterschieden.

Prozesse – sind Meßobjekte, welche einen zeitlichen Bezug besitzen, so wie z.B. der Softwareentwicklungsprozeß.

Produkte – sind Meßobjekte, die im Softwareentwicklungsprozeß als Dokumente entstehen, wie z.B. konzeptuelle Modelle, Quellcode.

Ressourcen – sind Meßobjekte, die im Softwareentwicklungsprozeß als Input benötigt werden, wie z.B. Personal, Software, Hardware.

Die Eigenschaften der Entities sollen durch die Softwaremessung erfaßt werden. Norman Fenton unterscheidet diese Eigenschaften nach internen und externen Attributen.

Interne Attribute – der Prozesse, Produkte oder Ressourcen werden direkt an den Entities gemessen.

Externe Attribute – der Prozesse, Produkte oder Ressourcen werden unter Berücksichtigung gewisser Umgebungsbedingungen gemessen. Sie können nicht direkt am Meßobjekt bestimmt werden.

Die Klassifikation für Softwaremetriken in der Tabelle 3.1 entspricht der Aufteilung der Metrik-arten aus dem Vorlesungsskript [Dum01], die wiederum der Grundklassifikation aus [Fen91] entspricht. Der Vorteil dieser Aufteilung gegenüber anderen Klassifikationen ist, daß sie aufgrund der Berücksichtigung vieler Softwareaspekte die unterschiedlichen Metriken sehr praxisnah einteilt. Der Ausgangspunkt ist das Meßobjekt, welches vom Benutzerstandpunkt aus intuitiv bekannt ist und somit die Auswahl einer geeigneten Metrik besser unterstützt als beispielsweise die Metrikauswahl über die Qualitätsmerkmale.

Tabelle 3.1: Klassifikation von Softwaremetriken

Meßobjekt **Metrik-Kategorie**	**Metrik-Unterkategorie**
Prozesse	
Maturity-Metriken	Organisations-, Ressourcen-, Technologie-, Datenmanagement-, Prozeßsteuerungs-, Dokumentationsstandard-, Prozeßmetriken
Management-Metriken	Projektmanagement (Meilenstein-, Risiko-, Review-, Produktivitätsmetriken) Qualitätsmanagement (Effizienz-, Nutzerzufriedenheits-, Qualitätsmetriken) Konfigurationsmanagement (Änderungs- und Versionskontrollmetriken)
Life-Cycle-Metriken	Problemdefinitions-, Analyse-, Spezifikations-, Entwurfs-, Implementations-, Wartungsmetriken
Produkte	
Umfangsmetriken	Projektumfang (LOC-Metriken, Anzahl von Dokumentationsseiten, ...) Entwicklungsmetriken (Anzahl von Testfällen, ...) Komponentenumfang (Modulanzahl, durchschnittliche Klassenmethodenlänge, ...)
Architekturmetriken	Komponenten-, Sprach-, Paradigmenanzahl, ...
Strukturmetriken	Tiefen-, Breiten-, Kopplungsmetriken
Qualitätsmetriken	siehe Abbildung 3.2 (Quantifizierung der qualitativen Eigenschaften durch Metriken)
Komplexitätsmetriken	algorithmische Komplexitätsmetriken (Operationsanzahl,...) psychologische Komplexitätsmetriken (Steuerfluß-, Datenfluß-, Entropie-, Topologiemetriken)
Ressourcen	
Personalmetriken	Programmiererfahrung, Kommunikationsniveau, Produktivitätsmetriken, Teamstruktur
Softwaremetriken	Leistungs-, Paradigmen-, Ersetzungsmetriken
Hardwaremetriken	Leistungs-, Verfügbarkeits-, Zuverlässigkeitsmetriken

3.2 Softwaremetriken

In diesem Abschnitt sollen einige bekannte Metriken bzw. Metriksammlungen vorgestellt werden. Auf die Definition von Metriken und die Verwendung von Metriken bei der Softwaremessung wurde bereits in der Motivation des Kapitels sowie in dem Abschnitt 3.1.1 eingegangen, so daß hier nur Metriken vorgestellt werden.

Zusätzlich zu der oben erwähnten Einschränkung der Metriken kann man die hier vorgestellten

Metriken auch entsprechend der Klassifikation aus Tabelle 3.1 in Abschnitt 3.1.3 den Produktmetriken zuordnen. Speziell werden hier die Umfangs-, Struktur- und psychologischen Komplexitätsmetriken interessieren, da diese Art der Metriken auch den zu entwickelnden DTD-Metriken entspricht. Nicht relevant sind dabei die Produktmetriken, die die Architektur, die Qualität und die algorithmische Komplexität bewerten.

Zunächst sollen konventionelle Metriken vorgestellt werden. Damit sind hier Metriken gemeint, die mehr oder weniger allgemeine Spracheigenschaften bewerten und so auch breite Verwendung z.B. als Basismetriken finden. Das soll aber nicht bedeuten, daß sie ohne Einschränkung einsetzbar sind. Beispielsweise sei hier der mögliche Einsatz des LOC-Umfangmaßes auf Methodenebene der Objekt-orientierten Programmiersprachen genannt, da die Methoden als kleinste funktionale Einheit – abgesehen von Eintritts- und Austrittspunkten – intern keine Objekt-orientierte Struktur aufweisen. Dies zeigt, daß konventionelle Metriken auch zusätzlich zu speziellen Metriksammlungen zur Messung einiger Eigenschaften bzw. auch als Basismetriken verwendet werden können.

Neben den konventionellen Metriken sollen in den nächsten Abschnitten zwei Metriksammlungen für konzeptuelle Modelle vorgestellt werden. Konzeptuelle Modelle beschreiben Informationsstrukturen basierend auf Objekten und deren Beziehungen. Die DTD-Beschreibungen definieren Dokumentstrukturen und sind insofern auch mit den konzeptuellen Modellen vergleichbar. Die konzeptuellen Modelle lassen sich nicht ohne Einschränkungen durch DTDs beschreiben, da die konzeptuellen Modelle jeweils selbst und auch die DTD versuchen, einen speziellen Ausschnitt der realen Welt zu beschreiben. Das ist einer der Gründe, weswegen sich beispielsweise die Metriksammlungen der konzeptuellen Modelle nicht auf die DTD anwenden lassen. An dieser Stelle sei erwähnt, daß es hier um eine allgemeine Betrachtung der DTD geht und nicht um eine spezielle anwendungsspezifische Sicht. Die Betrachtung der Metriken ist hier aufgrund von zwei Aspekten interessant. Zum einen wird XML, wie bereits in Abschnitt 2.1.3 erwähnt, auf diesen Gebieten eingesetzt und zum anderen lassen sich durch die Betrachtung dieser Metriken Erfahrungen für die DTD-Metriken sammeln.

3.2.1 Konventionelle Softwaremetriken

Die hier vorgestellten Metriken sind Komplexitätsmetriken. Basili definiert die Komplexität als ein Maß für Ressourcen, die von einem System verwendet werden, während es mit der Software interagiert, um eine Aufgabe zu erfüllen. Ist der Computer das interagierende System, wird die Komplexität durch die Rechenzeit und den Speicherbedarf, der für die Berechnung benötigt wird, bestimmt. Die Komplexität wird auch als algorithmische Komplexität bezeichnet. Wenn der Programmierer das interagierende System darstellt, wird die Komplexität durch die Schwierigkeit bestimmt, die Software zu implementieren, zu testen und zu ändern und als psychologische Komplexität bezeichnet. Da diese Komplexität meist direkt am Quellcode bestimmt wird, lassen sich die Maße als Codemetriken bezeichnen. Die folgenden Metrikbeschreibungen beschränken sich auf Codemetriken, da diese zunächst am aufschlußreichsten für die DTD-Metriken sind.

Bei der Messung der psychologischen Komplexität wird versucht, Programmeigenschaften zu erfassen, die dem Programmierer ein Programm als kompliziert erscheinen lassen. Dabei werden nur die signifikanten Eigenschaften durch die konzeptuelle oder auch modellbezogene Komplexität betrachtet. Zu Gunsten einer einfachen und verständlichen Beschreibung der Komplexität werden einige Programmeigenschaften als unwesentlich eingestuft, auch wenn sie die Komplexität in geringem Umfang mit beeinflussen. Die psychologischen Komplexitätsmaße können nach den Eigenschaften, die sie bewerteten, wie folgt unterschieden werden:

- Umfangs- und Längenmaße (Größenmaße)

  Bei diesen Maßen wird davon ausgegangen, daß der Quellcodeumfang alleine ausschlaggebend für die Komplexität ist. Das LOC- und Halstead-Maß zählen zu dieser Kategorie der Komplexitätsmaße.

- Datenflußmaße

  Die Maße dieser Kategorie versuchen, den Informationsfluß zwischen den Programmkomponenten zu bewerten. Die Programmkomponenten können Module, Prozeduren, Funktionen oder auch die Anweisungen innerhalb eines Moduls etc. sein. Demnach wird die Komplexität in inter- und intra-modulare Messungen unterteilt. Die Henry-Kafura-Metrik ist ein Beispiel für ein Komplexitätsmaß dieser Kategorie.

- Steuerflußmaße

  Die Programmstruktur läßt sich durch Flußdiagramme darstellen. Die Idee dieser Maße ist die Bewertung der Programmkomplexität basierend auf diesen Flußdiagrammen, durch die Bewertung der Anzahl und Typen von Schleifen, Verzweigungen und Bedingungen. Andere Programmeigenschaften – z.B. die Programmgröße – werden hingegen ignoriert. Das McCabe-Maß ist einer der bekanntesten Vertreter dieser Kategorie.

- hybride Maße

  Diese Maße vereinen unterschiedliche Aspekte der Komplexitätsbetrachtung in einem Maß. Das Oviedo-Maß (siehe [Zus91]) sei hier beispielhaft genannt, welches als Summe aus den einzelnen Komplexitäten des Daten- und des Steuerflusses berechnet wird.

3.2.1.1 Lines of Code (LOC)

Dieses Maß wird sehr oft für die Messung der Codezeilen genutzt. Für die Verwendung dieses Maßes ist die konkrete Beschreibung der Verwendung insofern notwendig, da es unterschiedliche Möglichkeiten für die Interpretation von Codezeilen gibt.

1. nur Zeilen mit ausführbaren Code zählen
2. ausführbaren Code und Definition von Daten zählen
3. ausführbaren Code, Definitionen von Daten und Kommentare zählen
4. ausführbaren Code, Definitionen von Daten, Kommentare im Quellcode und zugehörige Jobsteueranweisungen zählen
5. Erfassung der physikalischen Zeilen des Codes, wie sie sich bei der Verwendung eines Editors am Bildschirm darstellen
6. nur Begrenzer (delimiter) der Sprachkonstrukte zählen

Ziel der Messung der LOC sind zum einen Aussagen hinsichtlich der Änderbarkeit in Form von direkten Werten oder auch als empirisch bewertetes Komplexitätsmaß (siehe [Zus91]). Bei der Interpretation des Maßes wird davon ausgegangen, daß der Aufwand für den Test oder die Einarbeitung bei der Wartung um so größer ist, je mehr Codezeilen vorhanden sind.

LOC sind einfach zu messen und werden oft als Basismetrik für andere Betrachtungen – z.B. Fehlerhäufigkeit pro 1000 LOC – genutzt (siehe [Tha00]). Dieses Maß ist sehr stark Programmiersprachenabhängig. Beispielsweise umfaßt ein Programm in einer höheren Programmiersprache weitaus weniger Codezeilen als ein Programm in Maschinencode, so daß es schwierig ist, vergleichbare und gleichzeitig auch zuverlässige Aussagen über mehrere Programmiersprachen zu bekommen. Die unterschiedliche Auslegungsmöglichkeit bei der Definition des Maßes ist auch als Nachteil zu sehen.

3.2.1.2 Halstead-Maße

M. Halstead stellte mehr als ein Dutzend Maße zur Bewertung der Eigenschaften von Programmen auf, von denen die bekanntesten die Maße *Length* (N), *Volume* (V), *Difficulty* (D) und *Effort* (E) sind. Die Maße basieren auf der grammatikalischen Analyse des Quellcode, wobei der Programmtext in Operanden – Variablen, Konstanten, Marken, etc. – und Operatoren – Funktionen, Prozeduren, Schleifen, Iterationen, Zuweisungen, etc. –, also allen durch den Compiler erkannten Token, unterteilt wird. Somit ergeben sich anhand des Quellcode die vier folgenden Grundeinheiten, welche direkt am Quellcode gemessen werden.

$n1,\ n2$	–	Anzahl unterschiedlicher Operatoren bzw. Operanden
$N1,\ N2$	–	Gesamtanzahl der Operatoren bzw. Operanden

Die oben aufgezählten Komplexitätsmaße sind basierend auf den Grundeinheiten wie folgt definiert:

$n = n1 + n2$	–	Anzahl unterschiedlicher Operatoren und Operanden (Vocabulary)
$N = N1 + N2$	–	Anzahl aller Operatoren und Operanden (Length)
$V = N(\log_2 n)$	–	Programmumfang (Volume)
$D = (n1/2)(N2/n2)$	–	Programmkomplexität (Difficulty)
$E = DV$	–	Programmieraufwand (Effort)

Die Codelänge (N) wird aus den Grundeinheiten berechnet und die anderen Maße werden basierend auf den Grundeinheiten geschätzt. Bei der Bestimmung der Maße aus den meßbaren Grundeinheiten des Quellcode wird davon ausgegangen, daß die Maße direkt von diesen Grundeinheiten abhängen. Dies ist sehr umstritten, was man auch an der Anzahl der Modifizierungen des Maßes erkennen kann (siehe [Dum92]). Da die Komplexität nur aus den Größeneigenschaften des Quellcode – hier Anzahlen der Operanden und Operatoren – abgeleitet wird, zählen die Halstead-Maße zu den Größenmaßen. Durch die Erhöhung der Anzahl der Operanden und Operatoren erhöht sich auch die Komplexität, wobei sich nur das Maß der Programmlänge verhältnisskaliert darstellen läßt (siehe [Zus91]).

Wie beim LOC-Maß kann man auch beim Halstead-Maß als Vorteil die einfache Messung und die Verwendung als Basismetrik in anderen Maßen nennen. Durch die Abstraktion des Quellcode auf Operanden und Operatoren bleiben wichtige Eigenschaften – z.B. Schachtelungstiefe, Deklarationen, Gültigkeitsbereiche von Namen – unberücksichtigt. Weiterhin gilt für die einzelnen Maße – ausgenommen eventuell das Codelängenmaß (L) –, daß sie den gemessenen Quellcode als Ganzes bewerten. Das bedeutet für ein Programm, daß der Code immer zusammenhängend gemessen werden muß, auch wenn er modular ist. Die Komplexitätsmaße sind nicht additiv. Bei der Aufwandsschätzung (E) wird nur die Codeerstellung als Attribut berücksichtigt.

3.2.1.3 Henry-Kafura-Maß

Die durch S. Henry und D. Kafura definierte Programmkomplexität [HK81] basiert auf dem Informationsfluß der einzelnen Module. Der Informationsfluß wird bestimmt durch den lokalen Informationsfluß in ein Modul (*fan-in*) und durch den lokalen Informationsfluß aus dem Modul (*fan-out*). Die Beziehungen zwischen den einzelnen Modulen des Programmes lassen sich durch einen Informationsflußgraphen darstellen (siehe [Fen91]), welcher im Gegensatz zum Kontrollflußgraphen keinen Anfangs- und Endknoten hat. Die Module werden im Informationsflußgraph durch Knoten repräsentiert und ihre Beziehungen mit anderen Modulen durch gerichtete Kanten. Die Informationsflußkomplexität eines Modules (M) ist folgendermaßen definiert:

$$V_1(M) = (\mathit{fan\text{-}in}(M) \cdot \mathit{fan\text{-}out}(M))^2$$

Die eigentliche Modulkomplexität besteht aus der Kombination der Informationsflußkomplexität sowie der Datenflußkomplexität mittels dem McCabe-Maß oder der Programmcodekomplexität mittels dem LOC-Maß bzw. dem Halstead-Maß (hier z.B. durch *length*) wie folgt:

$$V_2(M) = length \cdot (fan\text{-}in(M) \cdot fan\text{-}out(M))^2$$

An dieser Stelle sei erwähnt, daß dieses Maß zu den Hybrid-Maßen gezählt wird. Bei der Messung der Informationsflußkomplexität eines Modules wird davon ausgegangen, daß diese proportional von der Anzahl der *fan-in* und *fan-out* Größen abhängt. Problematisch ist der Fall, wenn einer der Werte von *fan-in* oder *fan-out* Null ist, so daß daraus geschlossen werden muß, daß die gesamte Modulkomplexität Null ist, auch wenn das Modul z.B. eine hohe Code- bzw. Datenflußkomplexität hat. Durch die Messung der Informationsflußkomplexität sollen Module erkannt werden, die aufgrund der hohen Anzahl der Beziehungen zu anderen Modulen schwerer zu ändern sind als andere. Weiterhin wurde eine Korrelation zwischen der Modulkomplexität in Bezug auf die Informationskomplexität und der Fehlerhäufigkeit festgestellt. Das bedeutet, daß durch dieses Maß Mängel in der Funktionalität – hier durch zu hohe *fan-in* oder *fan-out* Werte – festgestellt werden können, was Aussagen zur Änderbarkeit sowie Zuverlässigkeit – hier zur Reife – ermöglicht.

Dieses Maß hat ebenfalls eine breite Verwendung gefunden, was aber nicht bedeutet, daß es uneingeschränkt verwendet wird. So wurden von M. Shepperd einige Änderungen zu diesen Maß (siehe [Fen91])vorgeschlagen, wie z.B.:

- Rekursive Modulaufrufe sollen wie normale Modulaufrufe behandelt werden.
- Module, die der Compiler hinzugefügt hat oder die aus Bibliotheken stammen, sollen ignoriert werden.
- Das Maß soll ausschließlich den Informationsfluß bewerten.

Gerade die zuletzt genannte Forderung ist konform zur Maßtheorie, da hier auch gefordert wird, daß ein Maß der Charakterisierung einer Eigenschaft (Attribut) entsprechen soll. Die Hybrid-Maße widersprechen dieser Forderung. Diese Forderung hat aber auch einen großen Vorteil. Das Maß (V_1) kann bereits in der Entwurfsphase angewendet werden, wobei das Hybridmaß (V_2) erst in der Implementationsphase sinnvoll eingesetzt werden kann, da erst hier die Codebewertung möglich ist. Weiterhin wurde durch Shepperd festgestellt, daß das Maß (V_1) im Gegensatz zum Maß (V_2) mit der Entwicklungszeit korreliert.

3.2.1.4 McCabe-Maß (zyklomatische Komplexität)

Das McCabe-Maß [McC76] bewertet den Steuerfluß basierend auf dem Kontrollflußgraphen des Programmes bzw. Moduls, wobei die Kontrollstruktur des Programmes auf der Sequenz, Selektion und Iteration basiert. Die Komplexität entspricht der Anzahl der linear unabhängigen Pfade im Kontrollflußgraphen. Damit lassen sich Aussagen zur Testbarkeit und Wartbarkeit treffen. Das Maß mißt die Komplexität des Kontrollflußgraphen (G) mittels der Kanten (e), der Knoten (n) und der beteiligten Komponenten (p, wobei $p \geq 1$) – Teilgraphen – und ist folgendermaßen definiert:

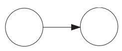
$$V(G) = e - n + 2p$$

Für die zyklomatische Komplexität $V(G)$ gelten die folgenden Eigenschaften:

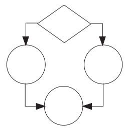
- $V(G) \geq 1$
- $V(G)$ entspricht der maximalen Anzahl linear unabhängiger Pfade in G
- $V(G)$ verändert sich nicht durch Hinzufügen oder Löschen einer Anweisung
- $V(G) = 1$, genau dann wenn G nur einen Pfad hat
- $V(G) = V(G) + 1$ beim Hinzufügen einer neuen Kante in G
- $V(G)$ hängt nur von der Entscheidungsstruktur (Steuer- bzw. Kontrollfluß) in G ab

In der folgenden Darstellung werden die im Kontrollflußgraphen üblichen Konstrukte mit ihrer Komplexität gezeigt. Auf die Darstellung der verketteten ELSE-Struktur (Switch-Anweisung) wird verzichtet, da sich diese Struktur von der IF-THEN-ELSE Struktur ableitet. Es sei hier nur erwähnt, daß das Hinzufügen einer Bedingung in der Switch-Anweisung die Komplexität $V(G)$ um 1 erhöht.

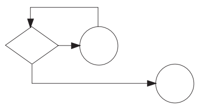
Sequenz
$V(G) = 1 - 2 + 2 = 1$



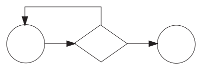
IF-THEN-ELSE
$V(G) = 4 - 4 + 2 = 2$



WHILE
$V(G) = 3 - 3 + 2 = 2$



DO-WHILE
$V(G) = 3 - 3 + 2 = 2$



Zum McCabe-Maß sei weiterhin erwähnt, daß durch Parameter p die Anzahl der beteiligten Komponenten des Programmes in $V(G)$ eingehen, wobei der Kontrollflußgraph des Programmes minimal aus einer verbundenen Komponente besteht. Besteht ein Programm aus einem Hauptprogramm und n Unterprogrammen, so wird $V(G)$ aus der Gesamtzahl aller Kanten, Knoten und $p = n + 1$ bestimmt. Dieselbe Komplexität erhält man aus der Summe der zyklomatischen Einzelkomplexitäten der Komponenten. Interessant an dieser Stelle ist auch die Möglichkeit der kompakten Darstellung des Kontrollflußgraphen in Form einer Erreichbarkeitsmatrix. In dieser $n \times n$ Matrix ist n die Anzahl der Knoten. Für jede Kante vom Knoten i zum Knoten j im Kontrollflußgraphen wird in der Matrix an der Stelle n_{ij} eine 1 eingetragen. Die Matrix kann zur Komplexitätsberechnung und als Datenstruktur z.B. für eine graphische Ausgabe genutzt werden.

Das McCabe-Maß ist ebenfalls ein sehr einfaches und sehr oft genutztes Maß für die Programmkomplexität auf der Grundlage der Graphentheorie. Kritisiert wird das Maß aber ebenso wie die zuvor genannten Maße. Es ist auf der einen Seite zu einfach, da es nur den Aspekt des Kontrollflusses zur Bewertung der Komplexität berücksichtigt. Andererseits ist es zu grob, da jede Verzweigung (siehe vorherige Beispiele) gleich bewertet wird und keine Bewertung der Komplexität der Datenstrukturen und auch der Anweisungen in das Maß mit einfließt. Das Maß korreliert zudem stark mit der Programmlänge.

3.2.2 Metriken für Objekt-orientierte Systeme

Objekt-orientierte Metriken gibt es schon eine Vielzahl. So sind zur Zeit über 200 Metriken bekannt. Da es keinen Sinn macht, die Metriken einzeln vorzustellen, sollen aus diesem Grund zunächst die Ziele und die charakteristischen Merkmale der Messung OO-Systeme beschrieben werden, bevor auf eine ausgewählte Metriksammlung eingegangen wird.

Zunächst einmal ist das Ziel der Messung mit OO-Metriken ähnlich der Messung prozeduraler Systeme. Man ist interessiert an der Quantifizierung der Qualitätsmerkmale wie z.B. Analysierbarkeit, Modifizierbarkeit und Prüfbarkeit aus der Sicht der Programmierer. Dies hängt mit den Anforderungen an die konzeptuellen Modelle bzw. Software zusammen, da sie verständlich, implementierbar, prüfbar, wartbar, erweiterbar und auch wiederverwendbar sein müssen. Die Metriken lassen sich den Metriken der psychologischen Komplexität zuordnen. Die Metriken korrelieren teilweise auch mit Eigenschaften anderer Nutzersichten, aber der Focus soll hier die Sicht des Programmierers sein. Bei der Bestimmung der psychologischen Komplexität ist man an statischen und strukturellen Aspekten der konzeptuellen Modelle bzw. auch des Quellcode interessiert.

Um OO-Systeme bewerten zu können, reichen die bekannten prozedurbasierten Metriken nicht aus, da sie Eigenschaften der Objektorientierung – wie Lokalität, Information Hiding, Vererbung, Abstraktion – nicht genügend berücksichtigen. Benötigt werden OO-Metriken, um die Methoden, Klassen und Beziehungen zwischen den Klassen bzw. deren Objekten untereinander geeignet bewerten zu können. Aufgrund der unterschiedlichen Eigenschaften, die zu bewerten sind, lassen sich die Metriken wie folgt klassifizieren in Metriken auf

– Methodenebene (Messen von Methodeneigenschaften z.B. durch konventionelle Metriken)
– Klassenebene (Messen von Klasseneigenschaften auf der Basis von Methodenmetriken)
– Vererbungsghierarchien (Messen der durch Vererbung erzielten Abstraktion)
– Aggregationshierarchien (Messen der Benutzt-Beziehungen zwischen den Klassen)

Wie sich zeigt, basieren einige Metriken auf bekannten konventionellen Metriken, da sich z.B. auf Methodenebene statische Maße (Codemaße) sowie strukturelle Aspekte (Daten- oder Informationsfluß) in ähnlicher Weise beschreiben lassen. Im Gegensatz zu den konventionellen Modellen sind die OO-Metriken meist in sogenannten Metriksammlungen zusammengefaßt. Ausschlaggebend dabei ist, daß die Bewertung der Qualität parallel zur Softwareentwicklung erfolgen soll, daß heißt beginnend mit der Analyse, aber spätestens beim Entwurf. Hierbei wird das Ziel verfolgt, daß die Metriken bzw. deren Werte über die einzelnen Phasen hinweg miteinander vergleichbar sind. Aus diesem Grund sollten auch keine unterschiedlichen bzw. modifizierten Metriken in den verschiedenen Phasen zur Anwendung kommen. In der Entwurfsphase werden bereits strukturelle Aspekte der späteren Software festgelegt und so lassen sich auch die verschiedenen OO-Metriksammlungen basierend auf unterschiedlichen OO-Entwurfsmodellen – wie z.B. OMT, OOD, OOA, UML – erklären. Die Metriken der speziellen Metriksammlungen bewerten die relevanten Aspekte der integrierten konzeptuellen Modelle der jeweiligen Entwurfsmethode. Das heißt nicht, daß die Metriksammlungen bzw. Metriken völlig unterschiedlich sind, sondern nur in einigen Punkten voneinander abweichen oder auch mehr bzw. weniger Metriken für die Bewertung der spezifischen Aspekte vorgeben. Zur Zeit werden auch noch nicht alle sprachspezifischen Konstrukte der unterschiedlichen OO-Programmiersprachen geeignet berücksichtigt, wie beispielsweise Templates, Friend-Classes, Exception-Handling von C++.

Als Beispiel für OO-Metriken sollen die Metriken der "Metric-Suite for OOD" [CK94] vorgestellt werden. Die Metriken dieser Metriksammlung werden oft als Basismetriken in OO-Qualitätsmodellen benutzt und sind bereits in Bezug auf die Eignung als Qualitätsindikatoren für OOD in [BBM95] validiert worden.

WMC – Weighted Methods per Class

Das Maß bestimmt die Komplexität der Klasse C aus der Summe der Einzelkomplexitäten der Methoden $c(m)$ die in der Klasse definiert sind, wobei n die Anzahl der Methoden in der Klasse ist. Zur Bestimmung der einzelnen Methodenkomplexität ist kein bestimmtes Maß vorgegeben.

$$WMC(C) = \sum_{i=1}^{n} c(m_i)$$

Unter der Annahme, daß alle Methoden dieselbe Komplexität besitzen, kann die Gewichtung der Komplexität der Methode mit $c(m) = 1$ festgelegt werden und entspricht dann der Anzahl der Methoden. Die Komplexität der Methoden kann auch durch die zyklomatische Komplexität von McCabe bestimmt werden. Eine hohe Klassenkomplexität läßt auf eine anwendungsspezifische Klasse schließen und bedingt zugleich eine schlechte Verständlichkeit und auch Wiederverwendbarkeit. Diese Kriterien korrelieren mit einer höheren Fehlerwahrscheinlichkeit der Klasse.

DIT – Depth in Inheritance Tree

Das Maß mißt den maximalen Weg in der Klassenhierarchie von der Oberklasse bis zur Wurzel.

Die Komplexität einer Klasse wird durch die Anzahl der Oberklassen beeinflußt, wobei tiefe Hierarchien schwerer wartbar und testbar sind. Da dieses Maß nur die Klassen unter der gegebenen Klasse betrachtet und nicht die gesamte Vererbungshierarchie, wird das Maß den Metriken der Klassenebene zugeordnet.

NOC – Number Of Children
Das Maß bestimmt die direkte Anzahl von Unterklassen.

Je mehr Unterklassen zu einer Klasse existieren, um so größer ist der Grad der Wiederverwendung in der Vererbungshierarchie. Außerdem zeigt das Maß, wie wichtig die jeweilige Klasse in der Vererbungshierarchie ist, wobei ein hoher Wert umfangreichere Tests erfordern kann.

CBO – Coupling Between Object classes
Das Maß bestimmt die Anzahl der Klassen, mit denen eine Klasse direkt kommuniziert. Dabei werden die Klassen gezählt, die von der Ausgangsklasse aus benutzt werden, als auch die Klassen, die die Ausgangsklasse nutzen. Kopplung zweier Klassen bedeutet hier, daß Methoden der einen Klasse Methoden oder Instanzvariablen der anderen Klasse aufrufen bzw. verwenden.

Ein hoher Kopplungsgrad einer Klasse widerspricht dem Konzept der Modularisierung und korreliert mit der Verständlichkeit der Klasse und somit auch mit der Wiederverwendbarkeit, Testbarkeit und Wartbarkeit.

RFC – Response For a Class
Das Maß $RFC(C)$ bestimmt die Anzahl der Methoden, die potentiell ausgeführt werden können, wenn ein Objekt der Klasse C auf eine eingegangene Nachricht reagiert. Dabei werden alle Methoden, die direkt aufgerufen werden können oder die durch Kopplung mit anderen Klassen erreichbar sind, gezählt.

Ein hoher Response-Wert der Klasse korreliert mit der Komplexität der Klasse und erhöht den Aufwand zum Testen der Methoden.

LCOM – Lack of COhesion in Methods
Das Maß $LCOM(C)$ bestimmt den Zusammenhalt der Klasse C, indem die Anzahl der Methoden von C, die keine gemeinsamen Instanzvariablen besitzen, von der Anzahl der Methoden von C, die gemeinsame Instanzvariablen besitzen, abgezogen werden. Im Fall, daß die Anzahl der Methoden mit gemeinsamen Instanzvariablen größer ist als die andere, wird $LCOM(C) = 0$ gesetzt.

Ein großer Zusammenhalt wird durch einen kleinen $LCOM$-Wert repräsentiert und ist wünschenswert, da er das Konzept der Einkapselung von Objekten widerspiegelt. Umgekehrt wird ein geringer Zusammenhalt durch einen hohen $LCOM$-Wert repräsentiert. Ein fehlender Zusammenhalt deutet auf Fehler im Entwurf hin, wobei die Klasse möglicherweise aufgeteilt werden sollte. Außerdem erhöht sich dadurch die Komplexität und die Fehlerwahrscheinlichkeit im Entwicklungsprozeß der Software.

Die Maße der "Metric-Suite for OOD" [CK94] charakterisieren die wesentlichen Eigenschaften der OO-Systeme, auch wenn sie durch andere Autoren erweitert oder teilweise verfeinert wurden. Eine gute Klassifikation von bekannten OO-Metriken ist in [Hog97] zu finden. An dieser Stelle sei noch einmal bemerkt, daß mit speziellen Metriken nicht das Problem der Vergleichbarkeit unterschiedlicher Entwicklungsprozesse bzw. Programmiersprachen gelöst werden kann, auch wenn man einige Basismetriken verwendet.

3.2.3 Metriken für Datenbanksysteme

Die Metriken für Datenbanksysteme sollen ähnlich den Metriken für Softwaresysteme den Entwicklungsprozeß bzw. den Lebenszyklus eines Datenbanksystems bewerten. Mit einem DBS ist hier die Kombination einer Datenbank mit einem DBMS gemeint. Die Phasen des Entwicklungsprozesses für ein DBS sind denen der Softwaresysteme sehr ähnlich. Die Datenbankqualität wird auch entscheidend durch die ersten Phasen im Entwicklungsprozeß beeinflußt. Hiermit ist die Erstellung des konzeptuellen Modells der Datenbank gemeint, welches die Grundlage für alle Phasen der Datenbankentwicklung bildet. Die Metriken, z.B. für OO-Systeme, setzen auch an diesem Punkt an und bewerten die im konzeptuellen Modell modellierten Eigenschaften. An dieser Stelle sei erwähnt, daß sich die Qualitätsmodelle für Software nicht direkt auf die Datenbanken übertragen lassen, da an die Datenbanken andere Anforderungen gestellt werden (siehe [SH97]), die zudem ein sehr komplexes Qualitätsmodell bedingen würden. Dies bedeutet insofern eine schlechtere Ausgangssituation für Datenbanksysteme, da mehr Eigenschaften berücksichtigt werden müssen und sich nicht alle Eigenschaften mit den herkömmlichen informalen Methoden der Softwaremessung erfassen lassen. Das ist aber andererseits auch nicht als großer Nachteil zu sehen, da sich Datenbanken im Gegensatz zu Softwaresystemen besser mit formalen Methoden – z.B. zur Erkennung funktionaler oder mehrwertiger Abhängigkeiten – auf ihre Eigenschaften hin überprüfen lassen.

Die Bewertung des konzeptuellen Modells von einem Datenbanksystem entspricht der modellbezogenen Softwaremessung. Bei der Bestimmung der wichtigen bzw. wesentlichen Qualitätsmerkmale für Datenmodelle existieren zur Zeit noch unterschiedliche Ansichten ([GJP00]), so daß es noch keinen gültigen Standard, z.B. ähnlich der ISO 9126 (siehe Abschnitt 3.1.2), gibt. Die folgend vorgestellten Metriken aus [GJP00] bewerten das ER-Modell für relationale DBS und orientieren sich dabei an dem Qualitätsmerkmal *Änderbarkeit* der ISO 9126. Ähnlich den Softwaremaßen wird versucht, über die Größe und Komplexität des ER-Modells das Qualitätsmerkmal Änderbarkeit mit seinen Teilmerkmalen bewertbar zu machen. Im ER-Modell (siehe [SH97]) werden Entities und deren Beziehungen dargestellt. Die Eigenschaften der Entities und auch der Beziehungen werden durch Attribute modelliert. Dies sind auch die Grundeinheiten, welche in den folgenden Metriken zur Bestimmung der Komplexität benutzt werden. Die Maße wurden noch nicht durch Anwendung in der Praxis validiert, so daß hier auch noch keine Eigenschaften angegeben werden können.

RvsE-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der Beziehungen und der Anzahl der Entities im ER-Diagramm.

$$RvsE = \left(\frac{N^R}{N^R+N^E}\right)^2 \quad \begin{cases} N^R \; \textit{Anzahl der Beziehungen im ER-Diagramm}, \\ N^E \; \textit{Anzahl der Entities im ER-Diagramm}, \\ N^R + N^E > 0 \end{cases}$$

Die Anzahl der Beziehungen pro Entity im ER-Diagramm beeinflußt diese Metrik. Dabei erhöht sich die Komplexität des ER-Diagramms mit steigender Anzahl der Beziehungen im Verhältnis zu den Entities. Die Komplexität ist Null, sofern keine Beziehungen im ER-Diagramm enthalten sind und tendiert gegen Eins, wenn sehr viele Beziehungen gegenüber wenigen Entities enthalten sind.

EAvsE-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der Entity-Attribute und der Anzahl der Entities im ER-Diagramm.

$$EAvsE = \left(\frac{N^{EA}}{N^{EA}+N^E}\right)^2 \quad \begin{cases} N^{EA} \; \textit{Anzahl der Entity-Attribute im ER-Diagramm}, \\ N^E \; \textit{Anzahl der Entities im ER-Diagramm}, \\ N^{EA} + N^E > 0 \end{cases}$$

Die Anzahl der Attribute pro Entity beeinflußt diese Metrik. Hätten die Entities keine Attribute im ER-Diagramm, wäre die Komplexität gleich Null. Die Komplexität tendiert gegen Eins mit steigender Anzahl der Attribute gegenüber den Entities.

RAvsR-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der Beziehungsattribute und der Anzahl der Beziehungen im ER-Diagramm.

$$RAvsR = \left(\frac{N^{RA}}{N^{RA}+N^{R}}\right)^2 \quad \begin{cases} N^{RA} \text{ Anzahl der Beziehungsattribute im ER-Diagramm}, \\ N^{R} \text{ Anzahl der Beziehungen im ER-Diagramm}, \\ N^{RA}+N^{R} > 0 \end{cases}$$

Die Is-a-Beziehungen und Aggregationsbeziehungen werden in dieser Metrik nicht berücksichtigt. Bis auf diese Ausnahmen wird diese Metrik durch die Anzahl der Beziehungsattribute und die Anzahl der Beziehungen beeinflußt. Das Verhalten der Metrik ist genauso wie das der zuvor beschriebenen *EAvsE* Metrik.

Die folgenden Metriken bewerten die unterschiedlichen Beziehungstypen, wobei die Is-a-Beziehungen und Aggregationsbeziehungen in diesen Metriken nicht berücksichtigt werden. Der Wert der folgenden Metriken ist Null, sofern der zu bewertende Beziehungstyp nicht im ER-Diagramm enthalten ist. Der Wert ist Eins, sofern nur dieser Beziehungstyp im ER-Diagramm enthalten ist. Ansonsten liegt der Wert zwischen Null und Eins.

M:N^{REL}-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der M:N Beziehungen und der Anzahl aller Beziehungen im ER-Diagramm.

$$M : N^{REL} = \frac{N^{M:NR}}{N^{R}} \quad \begin{cases} N^{M:NR} \text{ Anzahl der } M:N \text{ Beziehungen im ER-Diagramm}, \\ N^{R} \text{ Anzahl der Beziehungen im ER-Diagramm}, \\ N^{R} > 0 \end{cases}$$

1:N^{REL}-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der 1:N und 1:1 Beziehungen und der Anzahl aller Beziehungen im ER-Diagramm.

$$1 : N^{REL} = \frac{N^{1:NR}}{N^{R}} \quad \begin{cases} N^{1:NR} \text{ Anzahl der } 1:N \text{ und } 1:1 \text{ Beziehungen im} \\ \text{ER-Diagramm}, \\ N^{R} \text{ Anzahl der Beziehungen im ER-Diagramm}, \\ N^{R} > 0 \end{cases}$$

N–aryREL-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der n-stelligen ($n > 2$) Beziehungen und der Anzahl aller Beziehungen im ER-Diagramm.

$$N - aryREL = \frac{N^{N-aryR}}{N^{R}} \quad \begin{cases} N^{N-aryR} \text{ Anzahl der n-st. Beziehungen im} \\ \text{ER-Diagramm}, \\ N^{R} \text{ Anzahl der Beziehungen im ER-Diagramm}, \\ N^{R} > 0 \end{cases}$$

BinaryREL-Metrik

Metrik zur Bewertung des Verhältnisses zwischen der Anzahl der binären ($n = 2$) Beziehungen und der Anzahl aller Beziehungen im ER-Diagramm.

$$BinaryREL = \frac{N^{BinaryR}}{N^R} \quad \begin{cases} N^{BinaryR} \text{ Anzahl der binären Beziehungen im} \\ \text{ER-Diagramm}, \\ N^R \text{ Anzahl der Beziehungen im ER-Diagramm}, \\ N^R > 0 \end{cases}$$

Vorteil der einzelnen Metriken ist ihre Objektivität und einfache Berechenbarkeit. Für die Verwendung dieser Metriken in der Praxis fehlen noch Erkenntnisse, inwieweit die Maße wirklich mit den genannten Qualitätsmerkmalen korrelieren. Der Zusammenhang zwischen der Komplexität des Modells bezogen auf jeweils ein Maß ist intuitiv nachvollziehbar. Problematisch erscheint hier aber die Betrachtung der Gesamtkomplexität des ER-Modells, die sich natürlich nicht allein aus einem einzigen Maß ableiten läßt. In [GJP00] wird eine Möglichkeit vorgestellt, die Maße im Gesamtzusammenhang zu betrachten, um so die Gesamtkomplexität des ER-Modells zu ermitteln. Dazu ist eine empirische Bewertung der einzelnen Metriken notwendig, wobei die Gesamtkomplexität dann nicht mehr frei von subjektiven Einflüssen ist.

Im Gegensatz zu den Software-Metriken sind die Datenbank-Metriken erst seit kurzer Zeit Forschungsgegenstand, wodurch sich auch ihr geringer Bekanntheitsgrad erklärt. Die Metriken sind aber insofern interessant, da einige von ihnen Aufschluß über bestimmte Problemfälle bei der weiteren Verwendung der ER-Modellierung geben. Hier seien beispielsweise die M:N und auch die n-stelligen Beziehungen genannt, die nicht nur bei der Abbildung des ER-Modells auf das Relationenmodell die Komplexität erhöhen, sondern auch z.B. bei der Verwendung von XML als Datenaustauschformat.

3.3 Zusammenfassung

In diesem Kapitel wurde das Gebiet der Softwaremetrie überblicksweise vorgestellt und soll dabei zum besseren Verständnis der im nächsten Kapitel zu definierenden DTD-Metriken beitragen.

Im ersten Abschnitt wurde erläutert, wie bei der Softwaremessung vorgegangen wird. Besonders hervorgehoben wurde, welche Schritte für die Maßdefinition wichtig sind. Die ISO 9126 wurde vorgestellt, da sie allgemeingültige Qualitätsmerkmale für Software beschreibt. Die Qualitätsmerkmale definieren die qualitativen Eigenschaften der Software und sind nicht direkt meßbar. Aus diesem Grund müssen sie durch Softwaremetriken geeignet quantifiziert werden. Um einen Überblick über die vorhandenen Metriken und ihre Verwendung zu bekommen, wurden einige Metrikklassifikationen vorgestellt. Diese Klassifikationen werden bei den DTD-Metriken auch dazu dienen, sie aufgrund ihrer Eigenschaften in die vorhandenen Metriken einzuordnen. Andererseits war es notwendig, sich mit den bekannten Metriken auseinanderzusetzen, um bei der Definition der DTD-Metriken eventuell vorhandene Metriken oder auch nur deren Grundidee zu nutzen.

Im zweiten Abschnitt sind aus diesem Grund Metriken ausgewählt und vorgestellt worden, die als Basismetriken für die DTD verwendet werden können. Die vorgestellten konventionellen Metriken betrachten jeweils unterschiedliche Aspekte der Komplexität. Als Nachteil wird diesen Metriken angelastet, daß sie immer nur eine wesentliche Eigenschaft betrachten, um daraus die Komplexität zu bestimmen. Dies ist insofern richtig, da beispielsweise die Programmkomplexität nicht nur durch eine Eigenschaft geprägt wird, wie z.B. durch den Datenfluß oder den Informationsfluß. Andererseits wird durch die Maßtheorie gefordert, daß ein Maß die Charakterisierung einer Eigenschaft darstellen soll, was durch Hybridmaße nicht erfüllt wird.

Die Metriksammlungen für die konzeptuellen Modelle berücksichtigen die spezifischen Eigenschaften des jeweiligen Modells. Bei den vorgestellten Metriksammlungen wird durch jede Metrik jeweils eine charakteristische Eigenschaft des konzeptuellen Modells betrachtet, was auf der einen Seite die Gesamtbeurteilung der Komplexität für das konzeptuelle Modell aus den Einzelkomplexitäten aufwendiger macht. Andererseits hat dies den entscheidenden Vorteil, daß die differenziertere

Bewertung der Einzelmerkmale – z.B. in Bezug auf Funktionalitätsmängel, Strukturschwächen etc. – zu einem besseren Verständnis des Modells führt, da hierdurch festgestellt werden kann, welche Eigenschaft maßgeblich die Komplexität des gemessenen Modells beeinflußt. Bei der Betrachtung der Gesamtkomplexität gibt es verschiedene Möglichkeiten. Einfach scheint die Addition der Einzelkomplexitäten zu sein. Ohne Gewichtung der einzelnen Metriken bedeutet dies, daß jede Eigenschaft einen gleich großen Einfluß auf die Gesamtkomplexität hat, was in den meisten Fällen wohl eher nicht zutreffen wird. Auf der anderen Seite wird durch eine Gewichtung der Metriken die Bewertung subjektiv beeinflußt, was als Nachteil zu sehen ist.

Im Zusammenhang mit XML ist der Ausblick auf die OO-Metriken und Datenbank-Metriken insofern interessant, da sich XML als Austauschformat (siehe auch Abschnitt 2.1.3) für viele Daten anbietet. Probleme treten dabei insofern auf, da die konzeptuellen Modelle unterschiedlich ausdrucksstark sind und sich nicht direkt bzw. auch ohne Einschränkungen in XML abbilden lassen. Dies ist hier aber nicht Gegenstand der Betrachtung, denn im Mittelpunkt steht die Entwicklung von Metriken für XML-DTDs. Aus dieser Sicht sind die Metriken der betrachteten konzeptuellen Modelle aufschlußreich für die Entwicklung der DTD-Metriken.

Kapitel 4

XML-DTD-Metriken

In diesem Kapitel werden die entwickelten Metriken beschrieben. In Vorbereitung und zum besseren Verständnis dieser Metriken wurden in den vorangegangenen Kapiteln zum einen die aktuelle XML-Empfehlung [W3C00b] und zum anderen wichtige Aspekte der Softwaremetrie vorgestellt. Im Zusammenhang mit der Softwaremetrie sind auch bekannte konventionelle Metriken und Metriksammlungen vorgestellt worden, an deren Grundideen und Eigenschaften sich bei der Entwicklung der DTD-Metriken orientiert werden soll.

Zunächst wird in diesem Kapitel beschrieben, welche qualitativen Aspekte bei der Bewertung der DTD bzw. bei XML-Dokumentkollektionen wichtig erscheinen. Wie bereits bei der Beschreibung der Qualitätsmerkmale von Software im Abschnitt 3.1.2 erläutert, ist es nicht sinnvoll, ein spezielles Qualitätsmodell für XML zu entwickeln. Vielmehr sollen hier die allgemeinen Qualitätsmerkmale der ISO 9126 [ISO91a] evaluiert werden, inwieweit sie für die Bewertung von XML relevant sind oder auch nicht.

Danach werden die quantitativen Eigenschaften beschrieben, die die qualitativen Merkmale bewerten. Zuerst werden die Grundeinheiten der DTD definiert. Diese Grundeinheiten sind identisch mit den Deklarationen in der DTD oder auch den Auszeichnungen im XML-Dokument und können direkt gemessen werden. Zum einen repräsentieren diese Grundeinheiten einfache Quellcodeeigenschaften und sind weiterhin die Grundlage für die direkten Metriken. Aus diesen Grundeinheiten sind auch spezielle anwendungsspezifische Metriken definierbar, was hier aber nicht betrachtet werden soll. Die Beschreibung der Metriken folgt nach den Grundeinheiten. Sie werden nach direkten und indirekten Metriken unterschieden und auch in derselben Reihenfolge in diesem Kapitel vorgestellt. Für die indirekten Metriken ist notwendig ein Modell zu entwickeln, in dem die wesentlichen Eigenschaften der DTD auftreten und bewertbar sind. Dieses Modell dient hier ähnlich den konzeptuellen Modellen zur Darstellung und Bewertung der strukturellen Eigenschaften der DTD. Am Ende des Kapitels werden die Eigenschaften und Besonderheiten der XML-Bewertung noch einmal zusammengefaßt.

4.1 Qualitative Kriterien

Im Kapitel 3 wurde das Gebiet der Softwaremetrie vorgestellt. In diesem Zusammenhang wurde bereits erläutert, aus welchen Gründen eine Softwaremessung notwendig ist und welche unterschiedlichen Aspekte bzw. Ziele dabei verfolgt werden. In diesem Abschnitt wird beschrieben, welche qualitativen Ziele bei der Bewertung der XML-DTD im Mittelpunkt stehen und durch die DTD-Metriken quantitativ bewertet werden sollen.

In der XML-Empfehlung [W3C00b] sind die wesentlichen Entwurfsziele von XML aufgeführt. Ein Punkt ist dabei die Unterstützung eines breiten Anwendungsspektrums. Gerade dieser Punkt führt zu unterschiedlichen Entwurfszielen bei der Verwendung von XML und ist auch der Grund dafür, daß es kein allgemeingültiges Vorgehensmodell beim DTD-Entwurf gibt. Beim DTD-Entwurf wird eher intuitiv vorgegangen, was in [Arn00] und auch [Cho00] untersucht wurde. Dies hat zur Folge, daß hier nicht wie durch die Metriksammlungen ein konkretes Entwurfsmodell qualitativ und quantitativ bewertet werden kann. Hier können nur allgemeine Eigenschaften bewertet werden, die basierend auf der XML-Empfehlung in allen XML-Dokumenten bzw. auch DTDs unabhängig von der Art der Verwendung auftreten.

Im Abschnitt 3.1.2 wurde erwähnt, daß es unterschiedliche Qualitätsmodelle gibt und daß als Ausgangspunkt der qualitativen Betrachtung für XML das ISO 9126 Qualitätsmodell dienen soll. Dabei sind natürlich nicht alle Qualitätsmerkmale dieses Modells für die Bewertung relevant. Zu diesen Qualitätsmerkmalen zählen die Funktionalität, Effizienz, Zuverlässigkeit und Übertragbarkeit.

Funktionalität und Zuverlässigkeit sind hier zwei Aspekte, die durch unterschiedliche Testverfahren überprüft werden müssen und nicht Gegenstand dieser Bewertungen sind. Für die Bewertungen wird vorausgesetzt, daß die DTDs XML-konform und die betrachteten XML-Dokumente bezüglich ihrer DTD gültig sind. Die Effizienz scheidet als Qualitätsmerkmal aus Relevanzgründen aus. Aufgrund der Konzeption von XML für das Internet (siehe XML-Entwurfsziele in [W3C00b]) muß das Qualitätsmerkmal Übertragbarkeit hier auch nicht bewertet werden.

Ein weiterer Punkt der XML-Entwurfsziele ist die Lesbarkeit und angemessene Verständlichkeit der XML-Dokumente. Die angemessene Verständlichkeit kann bei gültigen Dokumenten gleichfalls auf die DTD bezogen werden und betrifft die Qualitätsmerkmale Änderbarkeit und Benutzbarkeit. Diese qualitativen Eigenschaften sind besonders bei der Änderung von DTDs und auch bei der Erstellung gültiger Dokumente relevant. Die Qualitätsmerkmale und ihre Unterkriterien werden im Abschnitt 3.1.2 beschrieben und sind Ausdruck für die Komplexität der DTD. Die Komplexität wird unter anderem durch die Größe beeinflußt. Nun ist aber nicht unbedingt gesagt, daß zwei gleich große Dokumente auch die dieselbe Komplexität besitzen. Ein weiteres Kriterium für die Komplexität ist die Dokumentstruktur, die bei der Erstellung von gültigen Dokumenten und auch bei Änderungen der DTD bekannt sein und berücksichtigt werden muß. Da die Dokumentstruktur durch die DTD festgelegt wird, ist die DTD Ausgangspunkt bei der Bewertung von XML-Dokumentkollektionen. Die quantitative Bewertung der DTD und der XML-Dokumente bezüglich der Kriterien Größe und Struktur wird in den folgenden Kapiteln beschrieben.

4.2 Grundeinheiten der XML-DTD

Im Abschnitt 2.2 wurde die aktuelle XML-Empfehlung XML 2^{nd} edition [W3C00b] bereits vorgestellt. Dabei wurden zunächst in Abschnitt 2.2.1 die Grundstruktur und der Aufbau der XML-Dokumente beschrieben und im folgenden Abschnitt 2.2.2 der Aufbau und die definierbaren Bestandteile der XML-DTD erläutert. Bei den Beschreibungen wurden die EBNF-Regeln der aktuellen XML-Empfehlung [W3C00b] zugrunde gelegt, die dafür in Anhang A kompakt aufgelistet sind. Anhand dieser Einführung sind die Bestandteile der XML-DTD – Elemente, Attribute, Entities, Notationen, Kommentare, PIs und bedingte Abschnitte – und ihre Verwendung in XML bereits bekannt. Diese Bestandteile werden benötigt, da sie direkt gemessen werden können und als Grundlage für die Metriken dienen. Im Zusammenhang mit der Softwaremessung (siehe Abschnitt 3.1.3) wurden sie als interne Attribute bezeichnet. Aufgrund der mehrfachen Verwendung des Begriffs "Attribut" sollen die internen Attribute der Softwaremessung hier als Grundeinheiten bezeichnet werden.

Die Wahl der Grundeinheiten baut auf der Beschreibung der DTD-Bestandteile im Abschnitt 2.2.2 auf. Hier wird beschrieben, welche DTD-Bestandteile als Grundeinheiten für die DTD-Metriken benötigt werden und welche nicht, wobei die DTD-Bestandteile in den folgenden Unterabschnitten separat beschrieben werden. Das hat den Hintergrund, daß die meisten DTD-Bestandteile nach speziellen Eigenschaften unterschieden werden können, so daß es für die Wahl von Grundeinheiten für die einzelnen DTD-Bestandteile auch mehrere Möglichkeiten gibt.

Die Grundeinheiten sind direkt meßbar und repräsentieren die Auftrittshäufigkeit des DTD-Bestandteils. In Abschnitt 3.2 wurde beschrieben, daß sich aufgrund von einfachen Eigenschaften des Quellcode, bzw. auch anhand der Grundeinheiten, bereits Aussagen zur Eigenschaft des Meßobjektes – hier der XML-DTD – aufstellen lassen. Aus diesem Grund sind diese Grundeinheiten nicht nur die Grundlage für die Metriken, sondern werden auch neben den Metriken als Quellcodeeigenschaften bei der Softwaremessung aufgeführt. Die durch das Vorhandensein oder auch die Größe der Auftrittshäufigkeit des Attributs eventuell ableitbare Eigenschaft wird auch in den folgenden Abschnitten beschrieben.

Im Zusammenhang mit der Beschreibung des XML-Dokumentes in Abschnitt 2.2.1 wurden die Möglichkeiten der DTD-Deklaration (siehe auch Anhang A, Regel `[28]` ff.) im XML-Dokument angegeben. Demnach kann eine DTD als interne, externe oder auch gemischte Teilmenge im Prolog eines XML-Dokumentes deklariert werden. Damit es bei der Messung der DTD-Grundeinheiten zu keinem Mißverständnis in Bezug auf eine Teilmenge oder eine Gesamt-DTD kommt, wird im folgenden von einer externen Gesamt-DTD ausgegangen. Dies hat den Hintergrund, daß die Untersuchungen der DTD zunächst ohne die Betrachtung des XML-Dokumentes erfolgen und ist insofern kein Nachteil, da es aus der Sicht eines XML-Dokumentes keine grundlegenden Unterschiede zwischen einer internen, externen oder auch gemischten Form der DTD gibt. Der Parser sorgt im Falle einer Gültigkeitsprüfung des XML-Dokumentes dafür, daß die einzelnen DTD-Teilmengen zu einer Gesamt-DTD zusammengefügt werden. Es sei hier auch bemerkt, daß eine Dokumenttyp-Deklaration aus Sicht des XML-Dokumentes nicht die vollständige DTD-Struktur referenzieren muß, z.B. wenn der im XML-Dokument gewählte Einstiegspunkt in der DTD nicht mit dem DTD-Wurzelelement übereinstimmt. Der Fall, daß eine externe DTD – z.B. mittels der Parameter-Entities – modularisiert ist, soll an dieser Stelle zunächst auch noch nicht betrachtet werden, sondern erst im Zusammenhang mit den Metriken. Ein Unterschied zwischen interner (Regel `[28b]`) und externer (Regel `[31]`) DTD-Teilmenge sei noch einmal erwähnt. Entsprechend der genannten Regeln können die bedingten Abschnitte (Regel `[61]` ff.) nur in der externen DTD-Teilmenge definiert werden (siehe auch Abschnitt 2.2.2).

4.2.1 Elemente

Die Elemente prägen den wesentlichsten Teil der Struktur für ein XML-Dokument. In der DTD werden die Elemente definiert, wobei zwischen Elementen mit Inhalt und leeren Elementen unterschieden wird. Die Elemente mit Inhalt können andere Elemente (Kindelemente) enthalten. Diese können ähnlich den regulären Ausdrücken miteinander kombiniert werden und beeinflussen so die Dokumentstruktur maßgeblich. Gefordert wird durch die Gültigkeitsbeschränkung der Regel `[45]` eine eindeutige Element-Deklaration. Demnach stimmt die Anzahl der Element-Deklarationen in einer *gültigen*[1] DTD mit der Anzahl der Elemente überein. Die Elemente können weiterhin noch nach ihren Inhaltsmodellen wie folgt unterschieden werden (siehe Regel `[46]`):

Element-Inhalt (Children)
: Im Inhalt dieser Elemente werden nur Elemente angegeben (siehe Regel `[47]`). Diese können ähnlich den regulären Ausdrücken kombiniert werden, wobei das Gegenstand der Betrachtungen bei der Definition der Metriken zur Strukturbewertung bzw. -komplexität ist. Dieses Inhaltsmodell ist gekennzeichnet durch seine besondere Strukturiertheit. Ein Beispiel dafür ist z.B. das Wurzelelement in einem XML-Dokument. Dieses Element dient als Container-Element für alle anderen Elemente.

[1] Hiermit sind die Restriktionen entsprechend der XML-Empfehlung [W3C00b] bei der Dokumenttyp-Deklaration gemeint, auch wenn diese nicht wie bei den XML-Dokumenten durch einen Parser validiert werden.

Gemischter Inhalt (Mixed Content)

Entsprechend der Regel `[51]` gibt es zwei Alternativen für die Definition des Inhaltsmodells. Entweder besteht der Inhalt nur aus Zeichendaten (`#PCDATA`) oder er ist gemischt aus Zeichendaten und Kindelementen (z.B. `(#PCDATA | E1 | E2 | ...)*`). Die Anzahl des Auftretens und die Reihenfolge der im Element definierten Kindelemente ist nicht restriktiv. Es wird einzig gefordert, daß der Name der Kindelemente im Inhalt nur einmal auftreten darf. Der Elementinhalt kann als semistrukturiert angesehen werden, da er aus unstrukturierten und strukturierten Inhalten gemischt ist. Bei der Verwendung dieses Elementinhalts als Grundeinheit ist eine getrennte Betrachtung entsprechend der oben genannten Alternativen denkbar.

EMPTY

Das Inhaltsmodell dieser Elemente enthält keine Kindelemente oder Zeichendaten. Die Eigenschaften dieses Elementes werden mehr oder weniger über seine Attribute (Metainformationen) bestimmt.

ANY

Dieses Inhaltsmodell bedeutet soviel, daß jedes beliebige in der DTD definierte Element als Kindelement im Inhalt des Elementes auftreten kann. Insofern ist der Inhalt als strukturiert zu sehen.

Da jedes Element durch eines der Inhaltsmodelle in der DTD charakterisiert wird, stimmt die Gesamtzahl aller in der DTD deklarierten Elemente mit der Summe der Elemente der unterschiedlichen Inhaltsmodelle überein. Eine weitere Unterscheidung der Inhaltsmodelle bezüglich der Kombination der Elemente mittels der regulären Ausdrücke soll an dieser Stelle nicht erfolgen, da dies bereits Kriterien für die Strukturkomplexität betrifft, die erst im Zusammenhang mit den Metriken interessieren werden.

4.2.2 Attribute

Attribute definieren Eigenschaften von Elementen. Aus diesem Grund ist bei der Attributdefinition die Zuordnung zu einem bestimmten Element erforderlich. Attribute können zusammengefaßt in sogenannten Attributlisten oder auch separat definiert werden (siehe Regel `[52]`). Wenn mehrere Attributdeklarationen zu einem Element vorhanden sind, werden diese bei der Verarbeitung zu einer vereinigt. Es gibt also Elemente mit und ohne Attributlisten und zu jedem Element können in der DTD beliebig viele Attribute definiert werden. Attribute sind nicht *wiederverwendbar*. Sie werden über die Attributdeklaration an ein Element gebunden und müssen für jedes Element separat definiert werden. Wenn Attribute für ein Element bzw. in einer Attributliste mehrfach definiert sind, so soll laut [W3C00b] (siehe auch Abschnitt 2.2.2) nur die erste Definition des Attributs gelten.

Entsprechend der Deklarationsmöglichkeiten für die Attribute gibt es für die Anzahl der Attribute unterschiedliche Auslegungsvarianten, welche alle in irgend einer Weise ihre Berechtigung haben, beispielsweise ähnlich der Diskussion um die Auslegung des LOC-Maßes (siehe Abschnitt 3.2.1.1).

Anzahl unterschiedlicher Attributlisten

Wenn die Anzahl der unterschiedlichen Attributlisten – hier in Bezug auf den Elementnamen in der Liste – angegeben wird, läßt sich daraus auch die Anzahl der Elemente mit und ohne Attributlisten ermitteln. Da die Attribute zu jedem Element separat deklariert werden müssen, ist es denkbar, daß es z.B. weniger unterschiedliche Attribute als Attributlisten in der DTD gibt und somit diese Anzahl wichtiger erscheint als die folgende.

Anzahl aller Attribute
Die Anzahl der Attribute kann interessant bei der Ermittlung der Größe (Umfang) der DTD sein. Da es unterschiedliche Attributtypen gibt, kann die Gesamtanzahl der Attribute auch zum Vergleich mit den unterschiedlichen Attributtypen genutzt bzw. in Relation mit der Anzahl des jeweiligen Attributtyps gesetzt werden. Bei der Angabe der Anzahl aller Attribute muß unbedingt angeben werden, wie die Anzahl bestimmt wurde. Beispielsweise ob die mehrfach definierten Attribute in den unterschiedlichen Attributlisten mitgezählt werden oder nicht. Mehrfach definierte Attribute einer Liste sollten dabei nur einmal gezählt werden, da für das XML-Dokument nur das erste Auftreten des Attributes gilt.

Es sind noch weitere Varianten denkbar, aber diese beiden Werte scheinen für die Betrachtung der DTD durch die folgenden Metriken erst einmal am wichtigsten. Wie bereits im Abschnitt 2.2.2 bei der Beschreibung der Attribute vorgestellt, lassen sich die Attributtypen zunächst in Zeichen-, Token- und Aufzählungs-Typen einteilen. Diese Einteilung sagt aber nicht allzu viel aus. Besser geeignet ist hier natürlich die Angabe der Anzahl der jeweiligen Attributtypen. Da für die folgenden Metriken eine differenzierte Bewertung der Attribute nach Typen nicht in Betracht gezogen wird, soll an dieser Stelle auch nicht weiter darauf eingegangen werden. In der folgenden Zusammenfassung zu den Grundeinheiten werden die einzelnen Attributtypen trotzdem als mögliche Kandidaten mit aufgezählt. Es ist nämlich denkbar, daß die DTD aufgrund spezieller Verwendungszwecke oder Eigenschaften auf das Vorhandensein einzelner Attributtypen geprüft werden muß, z.B. nach der Eigenschaft der vorhandenen Referenzierungen durch die `ID`- und `IDREF`-/`IDREFS`-Attribute. Wenn eine Betrachtung der unterschiedlichen Attributtypen vorgenommen wird, ähnlich wie bei den Elementen nach den unterschiedlichen Inhaltsmodellen, sollte die Anzahl aller Attribute mit der Summe der Anzahl der einzelnen Attributtypen übereinstimmen.

4.2.3 Entities

Die Entities werden in der DTD definiert. Auf die unterschiedlichen Eigenschaften wurde bereits in Abschnitt 2.2.2 eingegangen. Entsprechend der Verwendung der Entity-Referenz im XML-Dokument oder in der DTD werden die Entities in allgemeine Entities und Parameter-Entities unterteilt. Diese Unterteilung wird bei den folgenden Metriken für die DTDs nicht unbedingt benötigt, soll aber trotzdem an dieser Stelle als Grundeinheit erwähnt werden.

Allgemeine Entities
Diese Entities können innerhalb des XML-Dokumentes verwendet werden.

Parameter-Entities
Diese Entities werden in der DTD verwendet. Einerseits können interne Entity-Referenzen in der DTD dazu dienen, die Lesbarkeit zu erhöhen, indem sich oft wiederholende Textabschnitte als Entity definiert werden. Als externe Entity-Referenzen können sie andererseits für die Modularisierung von großen DTDs genutzt werden. Denkbar ist auch die Wiederverwendung von DTDs, bzw. auch bestimmter Teile von DTDs.

Wenn die Auftrittshäufigkeit der beiden genannten Entity-Typen von Interesse ist, sollte auch hier die Summe der Anzahl der beiden Entities mit der Anzahl aller Entities übereinstimmen. Entities können noch nach weiteren Eigenschaften – z.B. intern, extern, geparst – unterschieden werden, doch scheint eine weitere Unterscheidung für eine Grundeinheit hier erst einmal nicht notwendig.

4.2.4 Notationen

Notationen können genauso wie die Entities nur in der DTD definiert werden (siehe Regel `[82]` ff.). Mit Hilfe der Notationen werden Textersetzungsmuster für den Zugriff auf externe nicht XML-codierte Daten beschrieben, beispielsweise Binärdaten (siehe auch Abschnitt 2.2.2). Dies ist ein

Grund, weswegen die Notationen bei der Bewertung der DTD durch Metriken ausgeklammert werden könnten. Die Anzahl der in der DTD auftretenden Notationen wird in der folgenden Zusammenfassung trotzdem als Grundeinheit aufgenommen, da sie beispielsweise auch die Größe (Umfang) der DTD mit beeinflussen.

4.2.5 Kommentare

Kommentare sind einerseits wichtig, besonders im Zusammenhang mit großen und komplizierten DTDs. Das Messen der Anzahl der Kommentare läßt sich aber in keiner Weise in Bezug mit einer *guten* Kommentierung setzen, welche die Verständlichkeit und somit die Wartbarkeit, Änderbarkeit und Wiederverwendbarkeit der DTD erhöhen soll. Ein weiterer Grund ist, daß nicht jeder Kommentar auch wirklich ein solcher ist, beispielsweise die Auskommentierungen von Deklarationen etc.. Hier scheint eher das generelle Fehlen und eventuell eine zu geringe Anzahl von Kommentaren als Manko bewertbar zu sein, als daß die Höhe der Anzahl der Kommentare auf eine gute Kommentierung schließen läßt. Aus diesem Grund werden die Kommentare nicht als Grundeinheit aufgenommen.

4.2.6 PIs

Die PIs dienen in ähnlicher Weise wie die Kommentare als Zusatzinformationen, hier aber nicht für den Leser des Dokumentes, sondern als Informationen in Form von Anweisungen für andere Anwendungsprogramme. Laut [W3C00b] zählen die PIs nicht zu den Zeichendaten des Dokumentes bzw. der DTD, so daß sie auch nicht bei den Grundeinheiten berücksichtigt werden.

4.2.7 Bedingte Abschnitte

Bedingte Abschnitte können nur in einer externen DTD definiert werden (siehe Regel `[28]` ff.). Wie bereits in Abschnitt 2.2.2 erwähnt, ist ihre Verwendung auf wenige Einsatzmöglichkeiten begrenzt, weswegen die bedingten Abschnitte auch selten verwendet werden. Da ihre Anwendung keinen erkennbaren Einfluß auf das Qualitätsmerkmal Änderbarkeit z.B. durch Erhöhung der Verständlichkeit, Wartbarkeit, Modularisierbarkeit hat, soll es auch nicht als Grundeinheit aufgenommen werden. Ein weiterer Grund ist, daß die bedingten Abschnitte nicht in XML-Schema übernommen worden sind (siehe auch Abschnitt 2.3).

4.2.8 Fazit

In der folgenden Tabelle werden die in den vorherigen Abschnitten beschriebenen Grundeinheiten der DTD zusammengefaßt. Hierbei werden alle wichtigen Grundeinheiten erwähnt, also auch Grundeinheiten, welche nicht in der folgenden Metrikdefinition benötigt werden. Anhand der Markierung in der letzten Spalte der Tabelle ist zu erkennen, welche der Grundeinheiten bei den Metriken verwendet werden und welche nicht oder daß eine Verwendung für spezielle Metriken möglich ist.

Tabelle 4.1: XML-DTD-Grundeinheiten

DTD-Grundeinheiten	**Beschreibung**	**M.**
zu 4.2.1 Elementen		
Elemente	Anzahl der Elementdeklarationen	$\oplus$
weitere mögliche Einteilung nach Element-Inhaltsmodell		
Children	Anzahl der Elemente mit Elementinhalt	$\otimes$
Mixed Content	Anzahl der Elemente mit Mixed Content *oder getrennt nach*	$\otimes$
Mixed Content	Anzahl der Elemente mit Mixed Content	$\otimes$
Fortsetzung der Tabelle auf der nächsten Seite		

Tabelle 4.1: Fortsetzung der Tabelle 4.1

DTD-Grundeinheiten	**Beschreibung**	**M.**
`#PCDATA`	Anzahl der Elemente mit Zeichendaten	⊗
`EMPTY`	Anzahl der Elemente mit `EMPTY`-Inhaltsmodell	⊗
`ANY`	Anzahl der Elemente mit `ANY`-Inhaltsmodell	⊗
zu 4.2.2 Attributen		
Attributlisten	Anzahl der Attributlisten *(auch Anzahl der Elemente mit Attributen)*	⊗
oder auch		
Attribute	Anzahl aller Attribute *(Erläuterung der Anzahl notwendig!)*	⊕
weitere mögliche Einteilung nach Attribut-Typen		
`CDATA`-Attribute	Anzahl aller String-Attribut-Typen	⊗
`ID`-Attribute	Anzahl der `ID`-Attribut-Typen	⊗
`IDREF`-/`IDREFS`-Attribute	Anzahl der `IDREF`-/`IDREFS`-Attribut-Typen	⊗
`ENTITY`-/`ENTITIES`-Attribute	Anzahl der `ENTITY`-/`ENTITIES`-Attribut-Typen	⊗
`NMTOKEN`-/`NMTOKENS`-Attribute	Anzahl der `NMTOKEN`-/`NMTOKENS`-Attribut-Typen	⊗
`NOTATION`-Attribute	Anzahl der `NOTATION`-Attribut-Typen	⊗
Enumeration-Attribute	Anzahl der Enumeration-Attribut-Typen	⊗
zu 4.2.3 Entities		
Entities	Anzahl aller Entities	⊕
oder Unterscheidung nach		
allgemeine Entities	Anzahl der Entities für XML-Dokument-Referenzen	⊗
Parameter-Entities	Anzahl der Entities für XML-DTD-Referenzen	⊗
zu 4.2.4 Notationen		
Notationen	Anzahl der Notationen	⊕
zu 4.2.5 Kommentare		
nicht als Grundeinheit betrachtet		⊖
zu 4.2.6 PIs		
nicht als Grundeinheit betrachtet		⊖
zu 4.2.7 Bedingte Abschnitte		
nicht als Grundeinheit betrachtet		⊖

⊕ : verwendet -, ⊖ : nicht verwendet -, ⊗ : verwenden möglich – in den DTD-Metriken

Nicht alle Grundeinheiten werden in den DTD-Metriken verwendet. Dies hat den bereits erwähnten Grund, daß nicht alle Deklarationen in XML-Schema übernommen wurden. Die Metriken sollten möglichst ohne große Änderungen auf XML-Schema angewendet werden können oder zumindest mit denen von XML-Schema vergleichbar sein. Deshalb werden DTD-Konstrukte, die keinen großen Einfluß auf die Metriken haben und nicht in XML-Schema übernommen werden, nicht bei der Metrikdefinition berücksichtigt. Im Abschnitt 2.3 wurden dazu die DTD-Konstrukte denen des XML-Schema gegenübergestellt. Eine Ausnahme stellen die Entities dar, welche zwar nicht in XML-Schema übernommen wurden, deren Funktion aber mittels anderer Mechanismen realisiert wird.

4.3 Definition der DTD-Metriken

In diesem Abschnitt werden die entwickelten Metriken vorgestellt. Dabei lassen sich die Metriken in zwei Gruppen einteilen. Die direkten Metriken basieren auf den Grundeinheiten und lassen sich beispielsweise direkt durch Parsen der DTD ermitteln. Zu dieser Metrikkategorie zählt die Umfangsmetrik, die als erstes vorgestellt wird. Die anderen Metriken zählen zu den indirekten oder auch modellbezogenen Metriken. Im Zusammenhang mit diesen Metriken wurde ein spezieller DTD-Graph entworfen, an dem die wesentlichen strukturellen Eigenschaften der DTD beschrieben werden. Der DTD-Graph wird im Abschnitt 4.3.2 zusammen mit der Strukturkomplexität beschrieben. Bewerten lassen sich die Eigenschaften mittels Algorithmen der Graphentheorie (siehe [Tur96]).

In den folgenden Unterabschnitten wird jeweils eine Metrik vorgestellt. Die Abschnitte werden mit einer allgemeinen Beschreibung der DTD-Eigenschaft und einem Vergleich zu ähnlichen Metriken eingeleitet. Danach erfolgt eine genaue Beschreibung der speziellen DTD-Eigenschaft, die bewertet werden soll, und die Definition der Metrik. Nach der Definition wird die Metrik in Bezug auf ihre Verwendung bewertet. Hierbei wird auf die Besonderheiten der Metrik und auch auf das Verhalten der Metrik bei einer modularen DTD eingegangen. Abschließend wird die Anwendung der Metrik auf die XML-Dokumente diskutiert.

Im letzten Unterabschnitt werden die Eigenschaften der Metriken beschrieben, um die Metriken entsprechend der Klassifikationen der vorhandenen Metriken (siehe Abschnitt 3.1.3) einordnen zu können. Weitere Aspekte zu den Metriken sind in der Zusammenfassung am Ende des Kapitels zu finden, da diese die Grundeinheiten und Metriken gleichfalls betreffen.

4.3.1 Umfang (U)

Die Angaben im Quellcode sollen ähnlich dem LOC-Maß zur Bestimmung der Größe der DTD dienen. Dabei wird nicht einfach die Anzahl der Zeilen als Bewertungsmaßstab genommen, beispielsweise die physikalischen Zeilen des Quellcode im Editor, sondern die Anzahl der relevanten XML-Deklarationen. Diese Bewertung soll der Messung der Anweisungen im Quellcode eines Programmes entsprechen. Um die Anweisungen bzw. Deklarationen in der DTD bewerten zu können, muß erst einmal geklärt werden, welche Anweisungen für die Größe der DTD relevant sind.

Relevant für die Bestimmung der Größe der DTD sollen alle Anweisungen sein, welche die Größe des XML-Dokumentes beeinflussen. Dazu sollen alle Deklarationen der DTD gezählt werden, die direkt und indirekt im XML-Dokument verwendet werden, bzw. in einem auf Gültigkeit überprüften XML-Dokument unbedingt in der DTD deklariert sein müssen. Dies hat den Hintergrund, daß die Größe der DTD unter gewissen Bedingungen mit der Größe des XML-Dokumentes vergleichbar ist. Darauf wird am Ende des Abschnitts eingegangen.

In der DTD werden, wie bereits in den Abschnitten 2.2 und 4.2 beschrieben, die für ein XML-Dokument gültigen logischen Strukturen – durch Element-, Attribut-Deklarationen und bedingte Abschnitte – und physikalischen Strukturen – durch Entity- und Notations-Deklarationen – festgelegt. Diese Deklarationen nehmen unmittelbar Einfluß auf die Größe des XML-Dokumentes, wobei die bedingten Abschnitte nicht als relevant betrachtet werden. Die Gründe wurden bereits im Abschnitt 4.2.7 genannt. Bei den Entities sollen sowohl die allgemeinen Entities und die Parameter-Entities berücksichtigt werden, auch wenn die Parameter-Entities nur innerhalb der DTD verwendet werden. Die Notationen beeinflussen die Größe der DTD direkt und werden deshalb hier auch als relevante Eigenschaft mit aufgenommen.

Weiterhin können in DTDs sowie auch in XML-Dokumenten an *beliebiger* Stelle – siehe dazu [W3C00b] – Kommentare und PIs eingefügt werden. Diese Anweisungen innerhalb der DTD besitzen keine Relevanz für das XML-Dokument und sollen aus diesem Grund nicht bei der Größenbestimmung – wie in den Abschnitten 4.2.5 und 4.2.6 erwähnt – berücksichtigt werden.

Der Umfang einer DTD ($U_{DTD}(DTD)$) wird demnach bestimmt durch die Summe der XML-Deklarationen für Elemente (EL), Attribute (A), Entities (EN) und Notationen (N). Dieser

Zusammenhang wird durch die folgende Metrik beschrieben:

$$U_{DTD}(DTD) = EL + A + EN + N$$

Diese Umfangsmetrik stellt einen einfachen Indikator für die Größe der DTD dar und kann ähnlich dem LOC-Maß eingesetzt werden. Mit Hilfe dieser Metrik kann sehr einfach bestimmt werden, ob der Umfang einer DTD sich nach der Änderung ebenfalls geändert hat. Beispielsweise könnte man anhand der HTML-DTDs – hier SGML-DTD und nicht XML-DTD – von HTML 1.0 bis 4.2 nachträglich erkennen, ob eine DTD *nur* korrigiert oder auch durch neue Deklarationen erweitert wurde. Dieses Maß kann nicht als Indikator für die Komplexität einer DTD eingesetzt werden, da hier andere Eigenschaften – wie z.B. Struktur, Datenfluß – bewertet werden müssen, als nur die einfache Anzahl der DTD-Deklarationen. Dies wird deutlich durch die Beispiele im Anhang B gezeigt. Die Beispiele 1 und 2 haben einen ähnlichen Umfang, unterscheiden sich aber sehr stark in der Strukturkomplexität und auch Strukturtiefe.

Das Maß kann gleichfalls für interne DTDs und für externe DTDs, die mit Entity-Referenzen modularisiert sind, verwendet werden. Bei den modularisierten DTDs wird die Summe aller DTD-Konstrukte der einzelnen DTDs gebildet, wobei zu beachten ist, daß hierbei für jede Teil-DTD die Definition eines Parameter-Entities notwendig ist, so daß im Endeffekt die Metrik für jede Teil-DTD um ein Entity größer wird. Umgehen kann man das, indem man die Parameter-Entities grundsätzlich nicht mit in die Metrik einbezieht.

Um das Umfangsmaß für die DTD vergleichbar auf die XML-Dokumente bzw. auch XML-Dokumentkollektionen anwenden zu können, müssen einige Einschränkungen getroffen werden. Voraussetzung für den Vergleich ist, daß das XML-Dokument gültig bezüglich der zu vergleichenden DTD ist. Aufgrund der Elementdeklaration in der DTD können die in der DTD deklarierten Konstrukte wiederholt innerhalb des XML-Dokumentes auftreten, so daß der Vergleich, ob die in der DTD deklarierten Konstrukte auch im Dokument verwendet werden, auf einer einfachen Überprüfung basiert. Demzufolge kann die Größe eines XML-Dokumentes minimal 1 sein, denn es muß laut [W3C00b] mindestens ein Wurzelelement enthalten sein oder das XML-Dokument ist maximal so groß wie die DTD, wenn alle in der DTD deklarierten Konstrukte mindestens einmal im XML-Dokument auftreten. Um eine Vergleichbarkeit zu erreichen, muß das zuvor definierte Umfangsmaß angepaßt werden. Die Notationen dienen der Identifizierung von Nicht-XML-Daten und werden innerhalb von nicht geparsten Entities zur Bestimmung des Elementformats durch den Notation-Attribut-Typ oder in PIs verwendet. Da hier nur XML-Daten betrachtet werden sollen und die Notation nur innerhalb der Auszeichnung (Mark-up) auftritt, soll die Notation nicht im Umfangsmaß berücksichtigt werden, wenn es zum Vergleich genutzt wird. Die Parameter-Entities werden nur in der DTD und nicht im XML-Dokument verwendet und können deshalb auch nicht berücksichtigt werden. Durch diesen Vergleich zwischen dem Umfang eines XML-Dokumentes bzw. einer XML-Dokumentkollektion läßt sich feststellen, welche in der DTD deklarierten Konstrukte im XML-Dokument verwendet werden und welche nicht. Demnach ergibt sich ein verändertes Umfangsmaß. Die Notationen (N) und die Parameter-Entities (EN_P) gehen nicht mehr mit in den Umfang ein. Dafür werden nur noch die allgemeinen Entities (EN_A) berücksichtigt. Die geänderte Metrik für den Umfang $U_D(D)$, wobei D allgemein für Dokument steht, welche gleichzeitig auf XML-Dokumente und DTDs unter Beachtung der beschriebenen Einschränkungen angewendet werden kann, lautet somit folgendermaßen:

$$U_D(D) = EL + A + EN_A$$

4.3.2 Strukturkomplexität (SK)

Im Abschnitt 3.2.1 wurde bereits im Zusammenhang mit den konventionellen Metriken erläutert, was unter Komplexität zu verstehen ist. Die Metrik für die Strukturkomplexität der DTD soll hier die psychologische Komplexität einer DTD beschreiben. Zwar hat die Struktur auch Einfluß auf die Verarbeitung eines gültigen Dokumentes, also die algorithmische Komplexität, aber das ist nicht Gegenstand dieser Arbeit. Vielmehr steht die in der DTD definierte Dokumentstruktur als Maß für die Verständlichkeit in Bezug auf die Änderbarkeit und Benutzbarkeit der DTD im Mittelpunkt der hier betrachteten Komplexität.

Die Strukturkomplexität der DTD ist nicht direkt mit einer der in Abschnitt 3.2 vorgestellten Metriken realisierbar. Einfach wäre beispielsweise die Anwendung des Halstead-Maßes (siehe Abschnitt 3.2.1.2) durch Aufteilung der DTD-Grundeinheiten in Operatoren und Operanden, um damit die Komplexität zu berechnen. Der Nachteil, daß die Struktur durch dieses Maß in keiner Weise berücksichtigt wird, müßte hier auch in Kauf genommen werden. Aus diesem Grund ist die Verwendung des McCabe-Maßes (siehe Abschnitt 3.2.1.4) naheliegender, da es den Steuerfluß basierend auf der Programmstruktur bewertet. Problematisch dabei ist, daß die in der DTD definierte Grundstruktur nicht mit der Ablaufstruktur eines Programmes vergleichbar ist. Zunächst sollen die wesentlichen Gründe beschrieben werden, die ein Problem bei der Verwendung des McCabe-Maßes für die DTD-Strukturkomplexität darstellen, bevor erläutert wird, warum und wie die Grundidee des Strukturgraphen trotzdem genutzt werden kann. Danach werden die wesentlichen Eigenschaften der durch die DTD definierbaren Dokumentstruktur vorgestellt und basierend darauf die Strukturkomplexität definiert. Im Anschluß daran erfolgt die Bewertung und Beschreibung der Besonderheiten bei der Verwendung der Strukturkomplexität.

Der beim McCabe-Maß benutzte Steuerflußgraph besitzt einen einzigen Start- und Endknoten. Die XML-Dokumentstruktur wird dagegen in Form eines Baumes dargestellt, wobei das Wurzelelement mit dem Startknoten vergleichbar ist und durch die Blattknoten mehrere Endknoten existieren. Durch die Anwendung des McCabe-Maßes auf eine Baumstruktur ergibt sich eine Komplexität von $V(G) = 1$. Begründet ist dies darin, daß in einem Baum immer genau eine Kante weniger als Knoten vorhanden ist und laut Definition auch keine Kreise vorkommen können (siehe [Tur96]). Die zyklomatische Komplexität von McCabe beruht aber gerade darauf, daß Verzweigungen und Kreise im Graphen, die letztendlich auf einen einzigen Endknoten führen, die Strukturkomplexität erhöhen. Ein sequentielles Programm ohne diese Eigenschaften hat bei McCabe die gleiche Komplexität wie eine Baumstruktur. Eine Anwendung des McCabe-Maßes auf XML-Dokumente würde bedeuten, daß alle XML-Dokumente dieselbe Komplexität besitzen. Dies soll an dieser Stelle auch nicht das Problem sein, denn die Grundstruktur des XML-Dokumentes wird in der DTD nicht restriktiv durch genau eine Baumform vorgegeben. Die in der DTD definierte Grundstruktur für ein XML-Dokument ist durch einen DTD-Graphen darstellbar. Der DTD-Graph basiert auf einer Baum-ähnlichen Struktur, ist aber kein Baum. Es wird sich zeigen, daß es hier genauso wie beim Steuerflußgraphen ohne Verzweigungen von Vorteil ist, wenn die Komplexität des Baumes immer gleich groß ist.

Ein Baum wird laut Definition (siehe [Tur96]) durch einen zusammenhängenden ungerichteten Graphen dargestellt. Die Richtung im Baum ist durch die Wurzeldefinition vorgegeben. Der DTD-Graph muß auf Grund der Kreise als gerichteter Graph dargestellt werden. Bei der folgenden Problembeschreibung der DTD-Grundstruktur im Vergleich zur Baumstruktur wird von diesem Unterschied einmal abgesehen, da ein Baum auch durch einen gerichteten Graphen darstellbar ist.

Die Grundidee der Strukturkomplexität für die DTD ist der vom McCabe-Maß sehr ähnlich und basiert zunächst auf der Elementdeklaration in der DTD als Pedant zu den Anweisungen eines Programms. Wird beispielsweise in der DTD eine einfache Elementverschachtelung definiert, so daß der DTD-Graph einem Baum entspricht, so soll die Strukturkomplexität Eins sein. Demzufolge erhöhen alle einfachen Kanten zwischen den Knoten die Strukturkomplexität nicht, ähnlich wie in einem sequentiellen Programm. Würde beispielsweise die Anzahl aller Kanten oder eventuell auch noch die Anzahl der Knoten die Strukturkomplexität beeinflussen, so wäre das Maß zu stark abhängig von der Größe bzw. dem Umfang der DTD. Die Struktur des Baumes ist einfach erfaßbar und ist hier der Grund dafür, daß die Komplexität des Baumes immer gleich groß ist und dadurch die Strukturkomplexität nicht wesentlich beeinflußt. Eine DTD mit dieser Komplexität ist für XML-Dokumente sehr restriktiv. Jede zusätzliche Kante, die einen Kreis bzw. geschlossenen Weg im DTD-Graph erzeugt, soll die Strukturkomplexität um Eins erhöhen. Der DTD-Graph entspricht nicht mehr einem Baum, was hier als Indiz für eine erhöhte Strukturkomplexität aufgefaßt werden soll. Eine DTD mit einer größeren Strukturkomplexität ist in Bezug auf die mögliche XML-Dokument-Ausprägung flexibler. Da die DTD-Struktur nicht durch einen Baum beschrieben werden kann, sollen zunächst die Deklarationen vorgestellt werden, die der Grund dafür sind, daß der DTD-Graph nicht der Baumstruktur entspricht.

Mehrere Wurzelelemente
: Wenn die hierarchische Gliederung innerhalb der Elementdeklarationen nicht eingehalten ist, gibt es mehrere Kandidaten für Wurzelelemente. Diese Grundstruktur der DTD kann nicht durch einen zusammenhängenden Graphen repräsentiert werden. Somit widerspricht die Gesamtstruktur des DTD-Graphen der Definition vom Baum, der genau durch einen zusammenhängenden Graphen repräsentiert wird. Diese Deklaration ist in der XML-Empfehlung nicht durch eine Gültigkeitsbeschränkung verboten und stellt bei Beachtung einiger Einschränkungen für die Metrikanwendung kein Problem dar. Der Fall wird deshalb erst einmal nicht bei der Definition der Strukturkomplexität berücksichtigt. Die erwähnten Einschränkungen werden im Zusammenhang mit der Modularisierung nach der Definition der Metrik erläutert.

Rekursiver Elementaufruf
: Ausgehend von einer hierarchischen Elementdeklaration ist eine rekursive Elementschachtelung möglich. Diese ermöglicht in Bezug auf die Struktur des XML-Dokumentbaums die Erzeugung von rekursiven Nachkommen und führt zu einer Expansion in der Strukturtiefe des XML-Dokumentbaums. Im DTD-Graphen wird dies durch einen Kreis bzw. geschlossenen Weg zwischen den Knoten dargestellt. Der Kreis kann sich dabei minimal auf dasselbe Element beziehen, wird aber meist zwischen mindestens zwei Knoten existieren. Das widerspricht der Definition des Baumes. Diese zusätzliche Verzweigung erhöht die Strukturkomplexität. Angenommen diese Verzweigung könnte in der Baumstruktur auftreten, so würde sie die zuvor erwähnte Baumkomplexität basierend auf McCabe-Maß um Eins erhöhen.

Wiederholungsoperatoren
: Die Wiederholungsoperatoren ($^+$, *) sind bei der Betrachtung der Strukturkomplexität sehr wichtig. Gerade die Möglichkeit der Wiederholung von Elementen, gruppierten Elementen (Sequenzen, Alternativen) oder auch ganzen Inhaltsmodellen (beispielsweise Mixed Content) bei der Elementdeklaration machen die XML-Struktur auf der einen Seite flexibel, erhöhen andererseits die Strukturkomplexität. In Bezug auf die Struktur des XML-Dokumentbaums erweitert sich dieser in der Breite, also durch direkte Geschwisterknoten. Die Wiederholungen sollen im DTD-Graphen durch Kreise dargestellt werden und sollen die Strukturkomplexität wie die Rekursion gleichermaßen erhöhen.

Der DTD-Graph soll ähnlich dem Steuerflußgraphen vom McCabe-Maß dazu genutzt werden, die in der DTD definierte Struktur darzustellen. Anhand der Modellstruktur des DTD-Graphen läßt sich die Strukturkomplexität mit der gleichen Formel wie beim McCabe-Maß genauso einfach berechnen. Da die Richtung der Kreise im DTD-Graphen von Bedeutung ist, wird der DTD-Graph durch einen gerichteten Graphen dargestellt. Die Knoten im DTD-Graphen repräsentieren die Elemente der DTD und die durch die Inhaltsmodelle realisierten Elementverknüpfungen entsprechen den Kanten im DTD-Graphen. Die Beschreibung und Bewertung der möglichen Inhaltsmodelle erfolgt nach der Metrikdefinition. Bei der Definition der Metrik wird davon ausgegangen, daß in der DTD ein Element existiert, von dem aus alle anderen Elemente zu erreichen sind. Dies ist das Start- oder auch Wurzelelement der DTD und wird im DTD-Graph als Wurzelknoten bezeichnet. Dadurch wird erreicht, daß der betrachtete DTD-Graph ein zusammenhängender Graph ist. Die

Beschreibung der Modularisierung folgt später, gleichfalls die Erklärung, wie bei mehreren Wurzelknoten in der DTD vorgegangen werden kann. Die Strukturkomplexität (SK) einer DTD kann – basierend auf dem DTD-Graphen (G) – folgendermaßen berechnet werden:

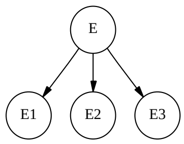
$$SK(G) = e - n + 2 \quad \begin{cases} e \text{ – } Anzahl\ der\ Kanten\ im\ Graph, \\ n \text{ – } Anzahl\ der\ Knoten\ im\ Graph, \\ n \geq 1 \end{cases}$$

Bei der Berechnung der Strukturkomplexität (kurz SK) wird davon ausgegangen, daß in der DTD ein Knoten (Wurzelelement) vorhanden ist. Die SK einer solchen DTD ist gleich Eins. Ein DTD-Graph ohne Kreise hat dieselbe SK, wie eine DTD mit nur einem Element. Die Beschreibung der Elementdeklaration kann in Abschnitt 2.2.2 nachgelesen werden. Basierend darauf wird jetzt die Darstellung der Elementdeklaration mit den möglichen Inhaltsmodellen durch den DTD-Graphen vorgestellt. Dabei wird noch auf einige Besonderheiten hingewiesen, sowie die jeweilige SK des Beispiels angegeben.

Children

Das deklarierte Element (`E`) sowie die Elemente im Inhaltsmodell (`E1, E2, E3`) werden als Knoten im DTD-Graph dargestellt. Die Elemente im Inhaltsmodell sind direkte Nachkommen von `E` im Graph, was durch die von `E` ausgehenden gerichteten Kanten repräsentiert wird. Desweiteren sind die Besonderheiten zu berücksichtigen, die bei den Elementverschachtelungen beschrieben werden.

```
<!ELEMENT E (E1,E2,E3)>
```

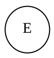


$SK(G) = 3 - 4 + 2 = 1$

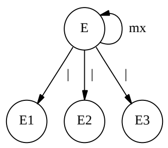
Mixed Content

In Abschnitt 2.2.2 wurde bereits beschrieben, daß es bei diesem Inhaltsmodell zwei Alternativen gibt. Sofern nur Zeichendaten und keine weiteren Elemente im Inhaltsmodell auftreten, wird nur das Element (`E`) durch einen Knoten im DTD-Graph repräsentiert. Die Zeichendaten beeinflussen die Struktur des DTD-Graphen nicht und werden aus diesem Grund nicht im Graph dargestellt. Wenn im Inhaltsmodell weitere Elemente (`E1|E2|E3`) angegeben sind, werden diese entsprechend dem *Children*-Inhaltsmodell im Graphen aufgenommen und durch Kanten vom deklarierten Element ausgehend verbunden. Die Kanten können zusätzlich gekennzeichnet werden, damit man die Art der Gruppierung erkennt, obwohl diese keinen Einfluß auf die SK hat. In diesem Inhaltsmodell tritt keine weitere Elementgruppierung auf, so daß der Kreis für den Quantor am deklarierten Element eingezeichnet werden kann. Dieser wird mit "*mx*" für das Inhaltsmodell gekennzeichnet.

```
<!ELEMENT E (#PCDATA)>
und
<!ELEMENT E (#PCDATA|E1|E2|E3)*>
```



und

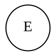


$SK(G) = 0 - 1 + 2 = 1$ $\qquad$ $SK(G) = 4 - 4 + 2 = 2$

EMPTY

Zu diesem Inhaltsmodell gibt es bezüglich der Struktur nichts zu erklären. Hier wird einzig das Element (`E`) als Knoten im DTD-Graph aufgenommen. Eine besondere Kennzeichnung des Knotens ist an dieser Stelle für die SK nicht angedacht, aber bei der Verwendung des DTD-Graphen für andere Zwecke denkbar.

`<!ELEMENT E EMPTY>`

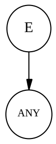


$SK(G) = 0 - 1 + 2 = 1$

ANY

Ein ANY-Element könnte ähnlich wie ein EMPTY-Element durch einen Knoten im Graphen repräsentiert werden und hat ebenfalls keinen Einfluß auf die SK. Da beide Inhaltsmodelle strukturell unterschiedlich sind, ist dies auch in der Darstellung berücksichtigt.

`<!ELEMENT E ANY>`



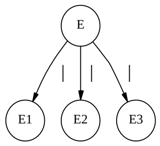
$SK(G) = 1 - 2 + 2 = 1$

Wie schon in Abschnitt 2.2.2 beschrieben, sind die Inhaltsmodelle für *Mixed Content*, *EMPTY* und *ANY* fest vorgegeben. Innerhalb des *Children*-Inhaltsmodells lassen sich die Elemente ähnlich den regulären Ausdrücken durch Wiederholungsoperatoren (?,$^+$,*), die Alternative, Sequenz und Gruppierung miteinander kombinieren. Gerade durch diese Operatoren wird die Flexibilität und auch zugleich Komplexität der DTD-Struktur erreicht. Im folgenden sollen die Besonderheiten der Inhaltsmodelle mit diesen Operatoren sowie die Rekursion beschrieben werden.

Alternative und Sequenz

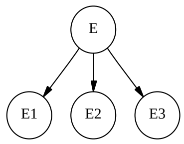
Die Struktur der beiden Inhaltsmodelle ist bei derselben Anzahl von Elementen gleich. Es gilt dasselbe für die Darstellung und SK wie beim *Children*-Inhaltsmodell. Einzig die Kanten sind bei der Alternative zur Unterscheidung zusätzlich gekennzeichnet.

`<!ELEMENT E (E1|E2|E3)>`
und
`<!ELEMENT E (E1,E2,E3)>`



$SK(G) = 3 - 4 + 2 = 1$

und

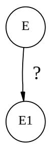


$SK(G) = 3 - 4 + 2 = 1$

Optionalität (?)

Die Optionalität eines Elements (bzw. einer Gruppe) hat auf die SK im DTD-Graphen keinen Einfluß. Um die Optionalität eines Elementes hervorzuheben, kann die Kante gekennzeichnet werden. Die Darstellung einer optionalen Elementgruppe wird bei der Gruppierung der Elemente beschrieben.

`<!ELEMENT E (E1?)>`
oder auch
`<!ELEMENT E (E1)?>`

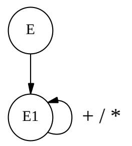


$SK(G) = 1 - 2 + 2 = 1$

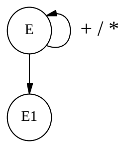
Wiederholungsoperatoren ($^+$,*)

Für die Darstellung der Elemente dieses Inhaltsmodells durch einen DTD-Graphen gilt dasselbe wie bei dem *Children*-Inhaltsmodell. Die Wiederholungsoperatoren an dem Element im Inhaltsmodell werden im DTD-Graphen durch einen Kreis am Element mit der entsprechenden Kennzeichnung dargestellt. Für die richtige Zuordnung des Kreises zum entsprechenden Element ist danach zu unterscheiden, wo der Wiederholungsoperator definiert ist. Im unten gezeigten Beispiel wäre die Unterscheidung in Bezug auf die SK zwar irrelevant, aber wenn es sich im zweiten Fall des Beispiels um eine Element-Gruppierung handelt, ist diese Unterscheidung wichtig. Die zusätzliche Kante im DTD-Graphen erhöht die SK.

<!ELEMENT E (E1$^+$)>/
<!ELEMENT E (E1*)>
und
<!ELEMENT E (E1)$^+$>/
<!ELEMENT E (E1)*>



und



$SK(G) = 2 - 2 + 2 = 2$ $SK(G) = 2 - 2 + 2 = 2$

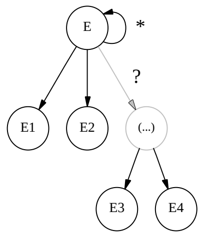
Gruppierungen von Elementen

Bei der einfachen Gruppierung von Elementen mit Wiederholungsoperatoren – z.B. Mixed Content – wird die zusätzliche Kante am Knoten des deklarierten Elements dargestellt. Bei der Darstellung einer Gruppierung innerhalb einer Gruppe mit oder ohne einen Wiederholungsoperator muß im DTD-Graph eine zusätzliche Kante mit Knoten eingeführt werden. Nur so läßt sich in diesem Fall die spezielle Kombination der Elemente in der Gruppe und die Wiederholung der Gruppe innerhalb des Inhaltsmodells von den anderen Elementen unterscheiden. Die Optionalität oder Wiederholung einer Gruppe wird dabei wie zuvor beschrieben dargestellt. Das gleiche gilt für die Optionalität, Wiederholung oder unterschiedliche Kombination der Elemente durch die Alternative oder Sequenz innerhalb der Gruppe. Die durch die eingefügte Gruppe entstandene Kante und Knoten beeinflussen die SK nicht. Im Fall der SK verhält sich das Mitzählen der Gruppenknoten und -kanten in Bezug auf das Ergebnis neutral, sollte aber trotzdem nicht erfolgen. Falls der DTD-Graph für andere Betrachtungen – z.B. Strukturtiefe – genutzt wird, muß das unbedingt beachtet werden. Aus diesem Grund sind die zusätzlichen Komponenten anders dargestellt.

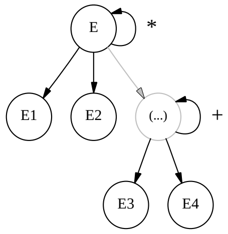
<!ELEMENT E (E1,E2,(E3,E4)?)*>

und

<!ELEMENT E (E1,E2,(E3,E4)$^+$)*>



und

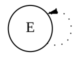


$SK(G) = 6 - 6 + 2 = 2$ $SK(G) = 7 - 6 + 2 = 3$

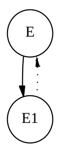
Elementrekursion

Die rekursive Verwendung von Elementen in der hierarchischen Grundstruktur der DTD sowie die Bewertung für die SK wurde zuvor schon beschrieben. Die Kante im DTD-Graphen, welche die Rekursion darstellt, ist hervorgehoben. Dies hat den Grund, daß eine Rekursion strukturell anders geartet ist als die Elementwiederholung durch die Wiederholungsoperatoren. Auch wenn die Elementrekursion komplexer erscheinen mag als die einfache Elementwiederholung, soll sie genauso bewertet werden. Eine Gewichtung der Kanten im DTD-Graph ist aber möglich. Beispielsweise sei hier noch erwähnt, daß bei einer mit einem Wiederholungsoperator gekennzeichneten rekursiven Elementdeklaration (siehe DTD-Graph im Anhang B.2 und B.4) im DTD-Graphen gleich zwei zusätzliche Kanten am Elementknoten entstehen.

<!ELEMENT E (E)>
oder
<!ELEMENT E (E1)>
<!ELEMENT E1 (E)>



oder



$SK(G) = 1 - 1 + 2 = 2$ $SK(G) = 2 - 2 + 2 = 2$

Anhand dieser Beschreibungen ist es möglich, für jede DTD einen DTD-Graphen aufzustellen und an diesem Modell die SK zu bestimmen. Im Anhang B sind Beispiel-DTDs und ihre DTD-Graphen enthalten. Bei der Darstellung dieser DTD-Graphen wurde auch die Reihenfolge der Elementdeklarationen eingehalten. Dies wurde bisher noch nicht erwähnt, ist aber unbedingt zu berücksichtigen. Bei der Sequenz ergibt sich diese Forderung schon aus der XML-Empfehlung [W3C00b] und bei der Alternative sollte man aus Gründen der Übersichtlichkeit der Strukturierung in der DTD folgen.

Die SK berücksichtigt einen weiteren Aspekt in der DTD. Innerhalb der DTD werden einige Elementdeklarationen sozusagen wiederverwendet, was daran zu erkennen ist, daß einige Elemente mehrfach im Inhaltsmodell eines anderen Modells vorkommen. Im DTD-Graph ist dies daran zu erkennen, daß einige Knoten bzw. Teilgraphen mehrfach auftreten und somit ihre Teilstrukturkomplexität ebenfalls mehrfach in die gesamte SK eingeht. Dies ist bei der SK der DTD beabsichtigt, da die Summe der einzelnen Teilstrukturkomplexitäten jeder Elementdeklaration nicht die SK der DTD darstellt. Das Wiederverwenden von Elementdeklarationen wird durch die $FAN\text{-}OUT$ Metrik noch separat betrachtet.

Wie an den Beispielen im Anhang unschwer zu erkennen ist, haben vom Umfang her gleich große DTDs eine sehr unterschiedliche SK. Dabei hat die bereits zuvor erwähnte Grundstruktur des Baumes keinen Einfluß auf das Komplexitätsmaß. Dies ist bei der DTD-Strukturmetrik ein Vorteil. Für die Bearbeitung der Daten eines Baumes gibt es effiziente Algorithmen, so daß es hier nicht als Nachteil zu sehen ist, die Baumstruktur einer sequentiellen Programmstruktur gleichzusetzen. Demzufolge werden nur Wiederholungen und Rekursionen, also Kreise im DTD-Graphen, als ausschlaggebend für die SK betrachtet. Gerade diese machen die XML-Struktur flexibel und auch komplex in Bezug auf die eingangs erwähnten Qualitätsattribute wie Verständlichkeit und Benutzbarkeit. Da die SK nicht direkt von der Größe der DTD abhängig ist, ist es denkbar, das eine vom Umfang her kleinere DTD eine größere Komplexität besitzt als eine große DTD. Zwar ist es wahrscheinlicher, daß eine große DTD auch eine größere SK besitzt als eine kleine DTD, aber dies liegt eher in der Natur der Sache und widerspricht nicht der zuvor genannten Tatsache. Gerade diese Unabhängigkeit, daß ein Maß nicht durch mehrere Eigenschaften beeinflußt wird, ist eine Forderung der Maßtheorie.

Die SK einer DTD kann als Indikator für ihre Flexibilität und zugleich auch Komplexität genutzt werden. Angenommen die SK der DTD ist gleich Eins. Die Struktur der DTD, welche durch den DTD-Graphen repräsentiert wird, entspricht in diesem Fall der eines Baums. Die für die XML-Dokumente vorgegebene Struktur ist sehr restriktiv, aber zugleich einfach, wie auch das Beispiel 1 im Anhang B.1 zeigt. Erhöht sich die SK dagegen, wird die DTD flexibler. Bei einer höheren SK unterscheidet sich die Datenstruktur von gültigen XML-Dokumenten stärker voneinander als bei einer kleineren SK. Die Größe der SK hat demnach Auswirkungen auf die eingangs erwähnten Qualitätsattribute Änderbarkeit und Benutzbarkeit (siehe auch Abschnitt 3.1.2), da mit der steigenden Komplexität die Verständlichkeit sinkt und dies mit der Fehlerwahrscheinlichkeit korreliert.

Interessant ist hier auch der Zusammenhang zwischen der Änderung der SK und den Auswirkungen auf die XML-Dokumente. Änderungen von Quantoren und rekursiven Elementverknüpfungen haben Einfluß auf die SK und somit auch Auswirkungen auf die XML-Dokumente. Dabei läßt sich die Änderung eines Quantors in Bezug auf die Strukturkomplexität mit dem Verhalten der Informationskapazität vergleichen, was in der Tabelle 4.2 dargestellt wird (siehe auch [Zei01] Tabelle 3.3). Hierbei spielt es keine Rolle, ob der Quantor an eine Elementgruppierung oder an ein einzelnes Element gebunden ist. Anhand dieses Vergleichs ist folgendes zu erkennen:

- Die Änderung eines Quantors in der DTD verhält sich in Bezug auf die SK genauso wie die Informationskapazität
- Änderungen eines Quantors, welche eine Reduktion der SK zur Folge haben, bedeuten gleichfalls eine Reduktion des Informationsgehalts im XML-Dokument.

Tabelle 4.2: Strukturkomplexität versus Kapazitäts- und Informationsverhalten

	Strukturkomplexität				Kapazitätsverhalten				Informationsverhalten			
	1	?	+	*	1	?	+	*	1	?	+	*
1		→	↗	↗		→	↗	↗		✓	✓	✓
?	→		↗	↗	→		↗	↗[2]	✓ ⫽		✓ ⫽	✓
+	↘	↘		→	↘	↘		→	✓ ⧹	✓ ⧹		✓
*	↘	↘	→		↘	↘	→		✓ ⧹	✓ ⧹	✓ ⫽	

→ : SK- bzw. Kapazitätserhaltung, ↗ : -erweiterung, ↘ : -reduktion
✓ : Informationserhaltung, ⫽ : -erweiterung, ⧹ : -reduktion

Das Löschen von rekursiven Elementaufrufen in der DTD führt zu einer Verkleinerung der SK und zieht Änderungen der XML-Dokumente nach sich. Werden dagegen rekursive Elementaufrufe in anderen Element-Inhaltsmodellen hinzugefügt, erhöht sich die SK. Eine Änderung der XML-Dokumente ist nur unter bestimmten Umständen notwendig. Das ist z.B. abhängig vom Inhaltsmodell des jeweiligen Elements, in dem die rekursive Elementdeklaration enthalten ist und dem Quantor, der an das Element gebunden ist. Dies wird in [Zei01] unter Umformungen auf Elementebene beschrieben. Als Fazit dieser Betrachtungen läßt sich folgendes festhalten:

- Einzelne Änderungen in der DTD an Quantoren und rekursiven Elementaufrufen, die zu einer Verkleinerung der SK führen, ziehen Änderungen der XML-Dokumente nach sich.
- Änderungen einzelner Quantoren in der DTD, die keine Änderung oder eine Erhöhung der SK zur Folge haben, haben keine Auswirkungen auf die XML-Dokumente.
- Eine Erhöhung der SK durch Hinzufügen von rekursiven Elementaufrufen hat nur unter bestimmten Bedingungen Auswirkungen auf die XML-Dokumente.

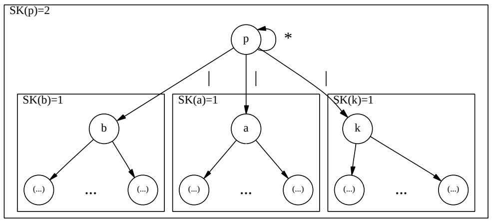
Bei der Problembeschreibung der DTD-Grundstruktur wurde am Anfang des Abschnitts erwähnt, daß die DTD unter Umständen nicht durch einen zusammenhängenden Graph darstellbar ist. Voraussetzung für die Bewertung der SK ist aber ein zusammenhängender Graph. In diesem Fall muß ähnlich der Bestimmung des Wurzelelements in einem XML-Dokument ein Element als Startknoten des DTD-Graphen angegeben werden, von dem aus die Komplexität der Struktur der erreichbaren Elemente berechnet werden soll.

Ist die DTD durch Parameter-Entities modularisiert und soll die SK der Gesamt-DTD bewertet werden, so kann die gleiche Metrik genutzt werden. Hierbei ist wiederum ein Element als Startknoten des DTD-Graphen zu bestimmen, von dem aus der Gesamt-Graph aufgespannt werden soll. Aufgrund der Definition der Metrik ist es nicht möglich, die Einzel-DTDs (Teil- und Haupt-DTD) zu bewerten und ihre Summe als SK der Gesamt-DTD anzugeben, was am folgenden Beispiel (siehe Abbildung 4.1) beschrieben wird. Angenommen wird, daß alle Teil-DTDs (buch.dtd, artikel.dtd und konferenz.dtd) eine SK von Eins haben und jeweils in einer Gesamt-DTD über ihre Wurzelelemente (b, a, k) integriert werden. Basierend auf einem DTD-Graph, welcher die Gesamt-DTD umfaßt, ergibt sich im Beispiel eine SK von Zwei. Dies ist richtig, da in der Haupt-DTD ein Quantor verwendet wird und somit in der Gesamt-DTD ein Kreis vorhanden ist. Im Gegensatz dazu würde sich aus der Summe der Teil-DTDs bereits eine Teilkomplexität von Drei ergeben, zu der noch die Teilkomplexität der Haupt-DTD von Zwei hinzugerechnet werden müßte.

Anhand dieses einfachen Beispiels ist zu sehen, daß bei der Modularisierung von DTDs von einem Gesamt-DTD-Graph ausgegangen werden muß, da sich ansonsten eine falsche SK ergibt. Die SK

[2] Diese Änderung bewirkt eine Kapazitätserweiterung, auch wenn in [Zei01] auf Seite 31 in der Tabelle 3.3 fälschlicher Weise eine Kapazitätsreduktion angegeben ist.

```
<!ENTITY % buch SYSTEM ”buch.dtd”>
<!ENTITY % artikel SYSTEM ”artikel.dtd”>
<!ENTITY % konferenz SYSTEM ”konferenz.dtd”>
%buch; %artikel; %konferenz;
   <!-- b Wurzelelement in buch.dtd -->
   <!-- a Wurzelelement in artikel.dtd -->
   <!-- k Wurzelelement in konferenz.dtd -->
<!ELEMENT p ( b | a | k )*>
```



$SK(dtd) = SK(p) = 2$

Abbildung 4.1: Strukturkomplexität und Modularisierung einer DTD

soll nicht durch die Modularisierung erhöht werden. Dies entspricht auch der Tatsache, daß eine Modularisierung eher die Komplexität verringert als erhöht. Es gibt aber noch weitere Gründe, die ausschlaggebend dafür sind, daß auch im Fall einer modularisierten DTD von einem Gesamt-DTD-Graph ausgegangen werden muß. Die Teil-DTDs müssen nicht, wie im Beispiel konstruiert, komplett über ihre Wurzelelemente in der Haupt-DTD integriert sein. Weiterhin ist es möglich, daß Teilgraphen der Teil-DTDs mehrfach in der Haupt-DTD auftreten etc.. Daher ist es für die SK einer DTD immer notwendig, den DTD-Graph vollständig zu betrachten. Beim McCabe-Maß erhöht dagegen jedes Modul die Komplexität um Eins.

Die SK ist in dieser Form nicht auf XML-Dokumente anwendbar, da die dem XML-Dokument zugrunde liegende Struktur ein Baum ist. Das XML Information Set [W3C01c] beschreibt den XML-Dokumentbaum, wobei alle in der DTD vorkommenden Bestandteile in sogenannten *Information Items* dargestellt werden können. Würde man in diesem Dokumentbaum nur die *Element Information Items* betrachten und einen gerichteten Graph ausgehend von der Dokumentwurzel, dem *Dokument Information Item*, darstellen, so würde dieser Graph in Bezug auf die DTD-Strukturkomplexität für jedes XML-Dokument dieselbe Komplexität besitzen. Man kann zwar von der SK der DTD auf die Flexibilität und Komplexität der XML-Dokumente schließen. Umgekehrt ist das jedoch nicht möglich.

4.3.3 Strukturtiefe (ST)

Die Darstellung der DTD in Form eines DTD-Graphen (siehe Abschnitt 4.3.2) führt zu weiteren strukturellen Eigenschaften, die für die Bewertung der DTD von Bedeutung sind. Basierend auf der baumförmigen Grundstruktur, ist die Auswertung der Strukturtiefe (kurz ST) interessant. Die ST beeinflußt auch die psychologische Komplexität in Bezug auf die Qualitätsattribute Änderbarkeit und Benutzbarkeit, wobei die ST der DTD von den anderen Maßen der DTD unabhängig ist und dementsprechend separat betrachtet wird.

Für die Bewertung der ST von anderen Modellen gibt es bereits Metriken, wie beispielsweise die DIT-Metrik (siehe Abschnitt 3.2.2) für die Bestimmung der Tiefe der Klassenhierarchie im OOD. Hierbei ist aber zu beachten, daß sich die Modelle für die Klassenhierarchie und den DTD-Graphen voneinander unterscheiden, auch wenn prinzipiell die maximale ST zwischen der Wurzel und den Blattknoten bzw. umgekehrt bestimmt wird.

Entsprechend dem DTD-Graphen soll die ST von der Wurzel ausgehend betrachtet werden. Da ein DTD-Graph nur eine Wurzel hat, wird auch der Wurzelknoten als einziger Knoten die maximale ST aufweisen. Interessant ist aber auch die ST der anderen Knoten. Im DTD-Graph wird jede Element-Deklaration durch einen Knoten repräsentiert, welcher immer dieselbe ST besitzt, auch wenn er im DTD-Graph mehrfach auftritt. Daraus läßt sich z.B. die Schlußfolgerung ziehen, daß ein hoher Wert der ST die Komplexität in Bezug auf die Änderbarkeit des Elements erhöht, wohingegen bei einem Blattknoten mit der ST von Null bei einer Änderung zumindest keine anderen Elemente betroffen sein können. Das entspricht auch der vorangegangenen Beschreibung der Informationskapazität der Elemente bei der Strukturkomplexität. Hier kann man sagen, daß ein Element mit einer höheren ST auch eine größere Informationskapazität besitzt und somit komplexer ist gegenüber einem Element mit einer geringeren ST. Die Bewertung der ST wird nach der Definition getrennt nach Gesamt-ST der DTD und der Element-ST erläutert.

Die DTD-ST entspricht in der DTD der maximalen Tiefe der Elementverschachtelung und ist einfach durch die maximale Anzahl von Kanten, ausgehend von der Wurzel bis zu den Blattknoten, anhand des Modells des DTD-Graphen zu bestimmen. Dies kann z.B. auch an den Beispielen im Anhang B nachvollzogen werden, wobei die zusätzlichen Kanten, die durch die Gruppierungen von Elementen entstanden sind, nicht mitgezählt werden dürfen. Das gleiche gilt für die Kanten, die die Wiederholungsquantoren und rekursiven Elementverschachtelungen im DTD-Graph repräsentieren. Aus dieser Sicht entspricht der DTD-Graph einem Baum, in dem jeder Ast genau ein Blatt enthält und die maximale ST bestimmbar ist. Die ST entspricht im DTD-Graph somit nur den vorwärts (bzw. nach unten) gerichteten Kanten von der Wurzel aus gesehen. Im DTD-Graph werden nur diese Kanten berücksichtigt, da DTDs das Schema für Dokumente darstellen und Dokumente endlich sind. Die Problematik von endlosen Programmen aufgrund von Schleifen oder Rekursionen muß hier also nicht betrachtet werden.

Eine andere Möglichkeit ist die Beschreibung über das Inhaltsmodell der Elemente, wobei hier nur die Elemente im Inhaltsmodell als nachfolgende Knoten in Betracht kommen, über die die ST definiert ist. Bei der Ermittlung der ST wird davon ausgegangen, daß eine direkte oder indirekte rekursive Elementverschachtelung wie ein Blattknoten betrachtet wird. Im folgenden werden die Definitionen der ST zur Ermittlung aus dem Inhaltsmodell der Elemente der DTD angegeben und erläutert:

Children

Dieses Inhaltsmodell würde im DTD-Graph durch zwei Knoten und eine gerichtete Kante von e nach e_1 dargestellt werden und soll die ST um Eins erhöhen, wobei weiterhin die ST von e_1 zu betrachten ist.

$$ST(e(e_1)) = max(ST(e_1)) + 1$$

Mixed Content

Sofern keine Elemente im Inhaltsmodell enthalten sind, handelt es sich um einen Blatt- bzw. Endknoten im DTD-Graph. Dieser Knoten besitzt keine weiterführende Kante zu einem anderen Knoten im DTD-Graph und erhöht somit die maximale ST der DTD nicht. Anders verhält es sich dabei, wenn Elemente enthalten sind. Da alle folgenden Elemente jeweils genau über eine Kante erreichbar sind, erhöht dieses Inhaltsmodell die ST um Eins, wobei weiterhin die ST der weiteren Elemente $(e_1; \ldots; e_n)$ zu betrachten ist. Die zusätzliche – durch "*mx*" gekennzeichnete – Kante im DTD-Graph für den Wiederholungsquantor wird bei der ST nicht betrachtet.

$$ST(e(\#PCDATA)) = 0$$

$$ST(e(\#PCDATA|e_1|\ldots|e_n)*) = max(ST(e_1; \ldots; e_n)) + 1$$

EMPTY

Für die ST von EMPTY gilt das gleiche wie bei der zuvor beschriebenen ersten Variante des Inhaltsmodells für Mixed Content.

$$ST(e(EMPTY)) = 0$$

ANY

Bei der Definition der ST dieses Inhaltsmodells mag es unterschiedliche Ansichten geben. Im DTD-Graph wird das Element, welches dieses Inhaltsmodell enthält, durch zwei Knoten und eine Kante dargestellt. Gerade dies bringt die mögliche minimale ST von Eins für dieses Inhaltsmodell zum Ausdruck, beispielsweise wenn ANY durch ein Element der DTD mit einer ST von NULL – z.B. EMPTY – ersetzt wird. Eine maximale Abschätzung ist an dieser Stelle zu ungewiß.

$$ST(e(ANY)) = 1$$

Nachdem die ST für die vier möglichen Inhaltsmodelle definiert wurde, folgt nun noch die Definition der speziellen Inhaltsmodelle, bei denen die Elemente durch Wiederholungsoperatoren, die Alternative oder Sequenz sowie rekursive Elementverschachtelungen kombiniert werden können.

Wiederholungsoperatoren

Die Quantoren haben keinen Einfluß auf die ST und werden deshalb bei den Elementen sowie auch den Gruppierungen nicht berücksichtigt. Einzig die Elemente im Inhaltsmodell bestimmen die ST.

$$ST(e(e_1)?) = ST(e(e_1)^+) = ST(e(e_1)^*) = ST(e(e_1)) = max(ST(e_1)) + 1$$

Gruppierungen von Elementen durch Sequenz und Alternative

Bei den Gruppierungen von Elementen durch die Sequenz oder Alternative wird ähnlich der zweiten Variante des Inhaltsmodells für Mixed Content vorgegangen. Da vom deklarierten Element (e) alle Elemente im Inhaltsmodell ($e_1; \ldots; e_n$) über jeweils eine Kante im DTD-Graph erreichbar sind, erhöht sich die ST um Eins. Zur Bestimmung der ST von e müssen weiterhin noch die ST von e_1 bis e_n bestimmt werden, wobei die Art der Gruppierung keine Rolle spielt.

$$ST(e(e_1, \ldots, e_n)) = ST(e(e_1| \ldots |e_n)) = max(ST(e_1; \ldots; e_n)) + 1$$

Elementrekursion

Die Elementrekursion kann nach direkter oder indirekter Rekursion unterschieden werden und soll die ST nicht erhöhen. Bei der direkten Rekursion ist die Bestimmung der ST einfach und in der DTD im Beispiel 2 im Anhang B.2 enthalten. Hingegen muß bei der indirekten Rekursion zunächst einmal das rekursive Element erkannt werden, damit die Bestimmung ST beim entsprechenden Element abgebrochen werden kann. Eine DTD mit indirekter Elementrekursion ist im Beispiel 4 im Anhang B.4 enthalten.

$$ST(e(e)) = 0$$

$$ST(e(e_1(\ldots(e_n(e))))) = n$$

Nachdem die ST definiert wurde, soll nun die Bewertung für die DTD beschrieben werden. Eingangs wurde erwähnt, daß hierbei einmal die maximale ST der DTD und die ST für jedes in der DTD deklarierte Element bestimmt werden soll. Dies wird in den beiden folgenden Unterabschnitten separat beschrieben. Da die Definition der Berechnung für die ST auf den deklarierten Elementen erfolgt, kann sie gleichermaßen für die maximale ST der DTD und die ST der Elemente angewendet werden.

4.3.3.1 DTD-Strukturtiefe (ST_{DTD})

Die ST der DTD entspricht der einfachen Tiefe des DTD-Graphen. Es werden demnach keine Elementrekursionen und Elementwiederholungen berücksichtigt. Die maximale ST bestimmt die

ST des Wurzelelements der DTD, wobei hier davon ausgegangen wird, daß es genau ein Wurzelelement in der DTD gibt. Wenn es mehrere mögliche Wurzelelemente in der DTD gibt, muß das relevante Element ausgewählt werden, von dem aus die ST der DTD ermittelt werden soll. Entsprechend den vorherigen Definitionen gilt für die maximale ST der DTD die folgende Metrik:

$$ST(DTD) = max(ST(e)) \ , \quad mit\ e\ \text{-}\ Wurzelelement\ der\ DTD$$

Die Höhe der ST der DTD ist als Indikator für die psychologische Komplexität zu sehen. Dabei ist eine DTD mit einer höheren ST in Bezug auf die Qualitätsattribute Verständlichkeit und Benutzbarkeit komplexer anzusehen, als eine DTD mit einer geringeren ST. Zwar ist dieses Maß unabhängig von anderen Maßen der DTD, sollte aber ebenso wenig wie die anderen Maße auch als allein ausschlaggebend für die Gesamtkomplexität einer DTD angesehen werden.

Eine DTD mit einer ST ab Neun soll hier als komplex in Bezug auf ihre ST gelten. Dabei entspricht die Zahl Neun dem Maximalwert der anthropologischen Konstante von G.A. Miller "*Sieben plus/minus Zwei*" in [Mil56]. In seinem Experiment hat G.A. Miller gezeigt, daß das menschliche Kurzzeitgedächtnis in Bezug auf die Informationsverarbeitung nur begrenzt aufnahmefähig ist. Es vermag nur ungefähr Sieben plus/minus Zwei kognitive Einheiten gleichzeitig aufzunehmen, zu speichern, an die man sich im nachhinein noch erinnern kann. Da diese Zahl oft im Zusammenhang mit der psychologischen Komplexität zitiert wird, soll sie hier auch als Ausgangspunkt einer Bewertung gelten, ohne daß dies an dieser Stelle experimentell nachgewiesen wird.

Entgegen der Bestimmung der Strukturkomplexität ist die Ermittlung der ST unbeeinflußt durch die Modularisierung einer DTD. Demzufolge ist es denkbar, daß, wie im Beispiel der Modularisierung bei der Strukturkomplexität (siehe Abbildung 4.1), im Fall einer kompletten Verwendung einer Teil-DTD dem Element in der Haupt-DTD die maximale ST der Teil-DTD bei der Ermittlung der ST hinzugerechnet wird. Wenn statt dem Wurzelelement aus der Teil-DTD ein anderes Element verwendet wird und die ST des Elements bekannt ist, so wird diese zur ST des Elements in der Haupt-DTD hinzugerechnet.

Die ST der DTD ist auf die XML-Dokumente anwendbar. Wie bereits im Zusammenhang mit der Strukturkomplexität beschrieben, läßt sich ein XML-Dokument in Form eines Baums (Information Set [W3C01c]) darstellen. Wenn in dieser Struktur nur die Elemente berücksichtigt werden und die rekursiven Elementverknüpfungen außer acht gelassen werden, so ist die maximale ST der DTD mit der ST der XML-Dokumente vergleichbar. Dabei kann die ST des XML-Dokuments kleiner als die ST der DTD sein, entspricht aber maximal der ST der DTD. Bei einem mehrfachen Auftreten von Elementen wird dabei dessen maximale ST im XML-Dokument bestimmt.

4.3.3.2 Element-Strukturtiefe (ST_E)

Genauso wie für das Wurzelelement läßt sich für jedes einzelne Element in der DTD die ST bestimmen. Dabei werden die Werte kleiner sein als die des Wurzelelements. End- bzw. Blattknoten im DTD-Graphen haben eine ST von Null. Auch wenn die ST der einzelnen Elemente kleiner ist als die maximale ST der DTD bzw. des Wurzelelements, so läßt sich in Bezug auf die Komplexität hier das gleiche feststellen. Elemente mit einer geringeren ST sind weniger komplex als Elemente mit einer höheren ST, was bereits im vorherigen Abschnitt auf DTD-Ebene beschrieben wurde.

Um einen besseren Überblick über die Verteilung der STn in der DTD zu erhalten, können die Werte aller Elemente in einem Diagramm dargestellt werden (siehe Beispiele im Anhang B). Dabei werden zunächst die STn aller Werte ermittelt und sind in den Beispielen in einer Tabelle zusammengefaßt. Im zweiten Schritt wird die Anzahl der Elemente für jede ST ermittelt und im Diagramm zur ST auf der Hauptachse (x-Achse) entsprechend zur Größenachse (y-Achse) eingetragen. Die Punkte bilden einen Kurvenverlauf, der die Verteilung der unterschiedlichen STn darstellt. In der Zusammenstellung der Beispiele im Anhang B.5 sind die Kurvenverläufe der

Beispiele in einem Diagramm zusammengefaßt. Zu erkennen ist, daß jede DTD genau ein Wurzelelement besitzt, wobei die ST unterschiedlich ist. Außerdem ist in den DTDs mindestens ein Element mit einer ST von Null deklariert, welches den bzw. die Endknoten der DTD repräsentiert. Ist der Kurvenverlauf nicht nur fallend, wie im Beispiel 1, so ist daran zu erkennen, daß die Elemente mit einer geringeren ST häufiger im Inhaltsmodell der Elemente mit höherer ST wiederverwendet werden. Dies ist besonders stark im Beispiel 2 zu erkennen.

Auch die Element-ST ist auf die XML-Dokumente anwendbar. Hierbei gilt das gleiche, was zuvor schon bei der maximalen ST der DTD beschrieben wurde. Da ein in der DTD deklariertes Element nicht unbedingt im XML-Dokument auftreten muß, ergibt sich hier auch ein anderer Verlauf der STn-Verteilung. In der graphischen Darstellung wird der Verlauf der Kurve des XML-Dokuments meist unterhalb der DTD-Kurve liegen und sich nur im Falle einer vollständigen Verwendung aller in der DTD deklarierten Elemente mit dieser Kurve decken.

4.3.4 Strukturbreite (FAN-IN)

Aus der Darstellung der DTD durch den DTD-Graphen ist eine weitere Struktureigenschaft ersichtlich, welche als Maß für die Komplexität der Elementdeklaration gelten soll. Die bisherigen Metriken, lediglich mit Ausnahme der Element-ST, haben bisher immer Eigenschaften der Gesamt-DTD betrachtet. Hier wird eine Eigenschaft auf Element-Ebene betrachtet, da die Strukturbreite der DTD aufgrund der größtenteils vorkommenden Unausgeglichenheit des DTD-Graphen weniger relevant erscheint.

Das Maß ist prinzipiell mit der Definition der NOC-Metrik (siehe Abschnitt 3.2.2) vergleichbar. Die NOC-Metrik bestimmt die Anzahl der direkten Unterklassen einer Klasse und dient bei den OO-Metriken als Maß für die Wiederverwendbarkeit. Hingegen wird Strukturbreite hier ähnlich der zuvor beschriebenen ST als Maß für die Komplexität auf Elementebene betrachtet. Die Wiederverwendbarkeit von Elementen wird durch die FAN-OUT-Metrik beschrieben. Ein Vergleich an dieser Stelle mit dem Henry-Kafura-Maß (siehe Abschnitt 3.2.1.3) ist nicht sinnvoll, auch wenn dort der Informationsfluß in eine Prozedur ähnlich bezeichnet wird. Die Verwendung und auch die Modelle sind hier zu unterschiedlich.

Bei der Definition des Maßes werden nur die möglichen Elemente bewertet, welche im Inhaltsmodell des deklarierten Elements vorkommen. Ihre Verknüpfung in Form von regulären Ausdrücken wird nicht berücksichtigt. Dadurch mag das Maß ziemlich einfach und ungenau erscheinen, aber dies ist gerade das Ziel der Darstellung des $FAN\text{-}IN$ der Elemente (siehe auch [CS00]). Ähnlich der Beschreibung der ST soll zunächst die Ermittlung informal über den DTD-Graphen beschrieben werden, bevor die formalere Definition basierend auf der Elementdeklaration der DTD erfolgt. In diesem Zusammenhang werden auch die Besonderheiten beschrieben, die bei der Ermittlung der Strukturbreite zu berücksichtigen sind, bevor die Bewertung der Metrik beschrieben wird.

Bei der Betrachtung des DTD-Graphen ist die Strukturbreite jedes Elements der DTD anhand der ausgehenden Kanten aus dem Knoten zu erkennen. Jede Elementdeklaration in der DTD wird im DTD-Graph durch einen Teil-DTD-Graphen repräsentiert. Auch wenn der Teil-Graph eines Elements mehrfach im DTD-Graphen vorkommt, hat er immer dieselbe Struktur. Die zusätzlichen Kanten und Knoten, welche die Elementgruppierungen im DTD-Graph darstellen, werden bei der Betrachtung ignoriert. Dafür werden die direkten Elemente der Elementgruppierung mitgezählt. Die zusätzlichen Kanten, welche die Elementwiederholungen im DTD-Graph repräsentieren, werden bei Ermittlung der Strukturbreite der Elemente ebenfalls ignoriert. Hingegen werden die Kanten, die die direkte oder auch indirekte Elementrekursion darstellen, bei der Strukturbreite berücksichtigt. Entsprechend der Definition der anderen Metriken soll hier auch die Definition auf der Grundlage der DTD Elementdeklarationen erfolgen.

Die Strukturbreite eines deklarierten Elementes entspricht der Anzahl der unterschiedlichen Elemente im Inhaltsmodell und wird basierend auf der Elementdeklaration folgendermaßen definiert:

Children

Das Element e_1 ist das einzige Element im Inhaltsmodell von e und somit ist der $FAN\text{-}IN$ von e Eins.

$$FAN\text{-}IN(e(e_1)) = 1$$

Mixed Content

In der ersten Variante des Inhaltsmodells für Mixed Content sind keine Elemente im Inhaltsmodell von e enthalten und aus diesem Grund ist der $FAN\text{-}IN$ gleich Null. In der zweiten Variante könnten n unterschiedliche Elemente im Inhaltsmodell von e auftreten, wobei zum Zeitpunkt der Bewertung der DTD noch nicht festgelegt werden kann, welches der Elemente im XML-Dokument auftritt und mit welcher Häufigkeit. Aus diesem Grund beträgt der $FAN\text{-}IN$ von e in der zweiten Variante n.

$$FAN\text{-}IN(e(\#PCDATA)) = 0$$

$$FAN\text{-}IN(e(\#PCDATA|e_1|\ldots|e_n)^*) = n$$

EMPTY

In diesem Inhaltsmodell sind ähnlich der ersten Variante des vorherigen Inhaltsmodells auch keine Elemente enthalten und somit ist der $FAN\text{-}IN$ ebenfalls gleich Null.

$$FAN\text{-}IN(e(EMPTY)) = 0$$

ANY

Laut Definition für das Inhaltsmodell ANY kann jedes beliebige Element der DTD eingesetzt werden. Demnach müßte der $FAN\text{-}IN$ des Inhaltsmodells ANY mindestens Eins sein, was mit der Darstellung im DTD-Graphen übereinstimmen würde. Da hier aber nicht bestimmt werden kann welches der Elemente und ob ein Element enthalten ist, soll der $FAN\text{-}IN$ für ANY Null sein. Dies ermöglicht weiterhin auch die Vergleichbarkeit mit dem $FAN\text{-}OUT$, was in den folgenden Abschnitten beschrieben wird.

$$FAN\text{-}IN(e(ANY)) = 0$$

Nachdem $FAN\text{-}IN$ für die vier möglichen Inhaltsmodelle definiert wurde, müssen noch die Besonderheiten der möglichen Kombinationen der Elemente durch die regulären Ausdrücke für das Children-Inhaltsmodell definiert werden.

Wiederholungsoperatoren

Die Wiederholungsoperatoren haben keinen Einfluß auf den Wert des $FAN\text{-}IN$, sondern lediglich die Anzahl der unterschiedlichen Elemente im Inhaltsmodell des deklarierten Elements e.

$$FAN\text{-}IN(e(e_1)?) = FAN\text{-}IN(e(e_1)^+) = FAN\text{-}IN(e(e_1)^*) = FAN\text{-}IN(e(e_1)) = 1$$

Gruppierungen von Elementen durch Sequenz und Alternative

Die Gruppierung bzw. die Kombination der Elemente mittels der Sequenz oder der Alternative haben keinen Einfluß auf den Wert von $FAN\text{-}IN$, sondern lediglich die Anzahl der unterschiedlichen Elemente im Inhaltsmodell des deklarierten Elements e. Hier sind weiterhin Beispiele aufgeführt, in denen ein Element (e_1) mehrfach vorkommt, wobei dieses im $FAN\text{-}IN$ nur einmal mitgezählt wird.

$FAN\text{-}IN(e(e_1, \ldots, e_n)) = FAN\text{-}IN(e(e_1| \ldots |e_n)) = n$

Beispiele von Besonderheiten bei der Gruppierung von Elementen:

$FAN\text{-}IN(e((e_1, e_2)|(e_1, e_3))) = FAN\text{-}IN(e(e_1, (e_2|e_3))) = 3$

$FAN\text{-}IN(e(e_1*, e_2*, e_3*, e_1*)) = 3$

Elementrekursion

Die Elementrekursion wird beim *FAN-IN* berücksichtigt. An dieser Stelle sei nur noch einmal die direkte Rekursion aufgeführt, da sich die indirekte Rekursion bereits aus den anderen Definitionen ergibt.

$FAN\text{-}IN(e(e)) = 1$

Nachdem die Strukturbreite als *FAN-IN* definiert wurde, wird nun beschrieben, wozu dieses Maß genutzt werden kann. Weiterhin wird erläutert, wie die FAN-IN-Werte der Elemente der DTD im Diagramm dargestellt und interpretiert werden.

Ähnlich den anderen Komplexitätsmaßen gilt für ein Element mit einem hohen FAN-IN-Wert, daß dieses komplexer ist als ein Element mit einem geringeren FAN-IN-Wert. Mit einer steigenden Anzahl von Elementen im Inhaltsmodell steigt gleichfalls die Informationskapazität des Elements und dadurch dessen Komplexität. Die Höhe der FAN-IN-Komplexität einer Elementdeklaration hat hier ebenfalls Einfluß auf die Qualitätsattribute Änderbarkeit und Benutzbarkeit. Wie bei der ST auch, soll als Maximalwert des FAN-IN Neun als Ausgangswert für die Komplexitätsbetrachtung festgelegt werden und entspricht der anthropologischen Konstante von G.A. Miller "*Sieben plus/minus Zwei*" in [Mil56]. Wie bereits bei den anderen Metriken erwähnt, ist für die Gesamtkomplexität der DTD eine Betrachtung aller Metriken notwendig und bei FAN-IN zu berücksichtigen, daß diese Metrik auf Elementebene definiert ist. Dies hat zur Folge, daß nicht ein besonders hoher FAN-IN eines Elements ausschlaggebend für die Gesamtkomplexität sein wird, sondern sich nur auf das jeweilige Element bezieht. Dies zeigt sich besonders gut am Beispiels 1 im Anhang B.1, welches den höchsten FAN-IN-Wert aller Beispiele hat. Betrachtet man das arithmetische Mittel der FAN-INs aller Beispiele im Anhang B.5 in der Tabelle B.1, so relativiert sich der Wert des Beispiel 1 wieder und deckt sich mit den anderen Komplexitätswerten, welche gegenüber den anderen Beispielen ebenfalls am niedrigsten sind.

Die FAN-IN Werte der Elemente können in einem Diagramm (siehe Anhang B und auch [CS00]) veranschaulicht werden. Dazu wurden die Elemente nach ihren FAN-IN-Werten absteigend sortiert und danach elementweise im Diagramm eingetragen. Die Sortierung der FAN-IN-Werte spiegelt sich im Kurvenverlauf wieder. Die Einteilung der Elementachse (x-Achse) entspricht nicht der Element-ID aus der Wertetabelle. Hieran ist lediglich die Gesamtzahl der Elemente der DTD und die Anzahl der Elemente mit gleichem FAN-IN ablesbar. Elemente mit einem FAN-IN von NULL sind als End- bzw. Blattknoten des DTD-Graph zu erkennen. In Beispiel 2 im Anhang B.2 gibt es keine Elemente mit einem FAN-IN von NULL, was auf die direkte Elementrekursion des eigentlichen Endknotens im DTD-Graph zurückzuführen ist.

Da bei der Bewertung des FAN-IN nur die einzelnen Elementdeklarationen betrachtet werden, ist es an dieser Stelle völlig egal, ob die DTD modularisiert ist oder nicht. Es muß lediglich bestimmt werden, welche Element-Deklarationen in der DTD relevant sind. Betrachtet man den DTD-Graphen, so sind die Elemente relevant, welche vom Wurzelknoten aus erreichbar sind. Für diese Elemente muß der FAN-IN bestimmt werden.

Der FAN-IN für ein Element im XML-Dokument wird unterschiedlich sein und kann dabei höchstens den maximalen FAN-IN-Wert der Elementdeklaration in der DTD annehmen. Durch das mehrfache Auftreten eines Elements im Dokumentbaum wird es vorkommen, daß zu einem Element unterschiedliche FAN-IN-Werte gemessen werden. Aus diesem Grund wird zum Vergleich der FAN-IN-Werte zwischen der DTD und den Dokumenten der Maximalwert des Elements im jeweiligen XML-Dokument betrachtet.

4.3.5 Elementwiederverwendung (FAN-OUT)

Durch die FAN-OUT-Metrik sollen ähnlich wie durch den FAN-IN die Elementdeklarationen bewertet werden. Im Gegensatz zum FAN-IN wird nicht das Inhaltsmodell des jeweiligen Elements betrachtet, sondern ermittelt, wie oft das deklarierte Element im Inhalt von anderen Elementen wiederverwendet wird. Genauso wie durch die FAN-IN-Metrik werden auch durch den FAN-OUT Eigenschaften auf der Element-Ebene betrachtet, wobei die betrachteten Eigenschaften unabhängig voneinander sind. Trotz der Unabhängigkeit der Werte auf der Element-Ebene besteht ein Zusammenhang zwischen FAN-IN und FAN-OUT. Dieser wird nach der Bewertung der Metrik in einem Unterabschnitt beschrieben.

Zunächst einmal soll der Begriff "*Elementwiederverwendung*" relativiert werden. Die Wiederverwendbarkeit im Sinne der Softwaremetrie ist ein weitläufiger Begriff, wobei hiermit die Wiederverwendung von Methoden, einzelnen Programmen oder auch ganzen Softwaresystemen gemeint sein kann. Durch die Messung der genannten Eigenschaft soll festgestellt werden, ob die genannten Komponenten wiederverwendet werden können oder neuentwickelt werden müssen. Diese Art der Wiederverwendung ist nicht das Ziel der Messung des FAN-OUTs der Elemente in der DTD. Es soll hier lediglich festgestellt werden, ob und wie oft ein Element in einer anderen Elementdeklaration vorkommt. Somit entspricht FAN-OUT im Prinzip der Wiederverwendbarkeit der NOC-Metrik, wobei aber berücksichtigt werden muß, daß sich das Modell des DTD-Graphen und der Klassenhierarchie unterscheiden.

Der FAN-OUT eines Elements ist ein wenig umständlicher zu definieren. Der DTD-Graph müßte zur Bestimmung des FAN-OUT rückwärts durchlaufen werden, wobei nur die unterschiedlichen und direkten Vorgängerknoten zu jedem Element notiert werden. Die Kanten und Knoten der Elementgruppierungen sowie die Kanten der Elementwiederholungen werden gleichfalls wie beim FAN-IN ignoriert. Hingegen werden die rekursiven Kanten ebenfalls betrachtet und die Knoten als FAN-OUT bewertet. Da ein Element in mehr als einer Elementdeklaration vorkommen kann, muß jeder direkte Vorgängerknoten des Elements betrachtet werden. Gezählt werden dabei aber lediglich die unterschiedlichen Vorgängerknoten bzw. Elemente.

Der FAN-OUT wird an dieser Stelle nicht formal definiert. Dafür wird nach diesem Abschnitt eine Möglichkeit vorgestellt, wie aus den Daten für die Bestimmung des FAN-IN der Elemente gleichfalls der FAN-OUT der Elemente ermittelt werden kann.

Der FAN-OUT eines Elements soll auch als ein Maß für die Komplexität auf Element-Ebene betrachtet werden. In Bezug auf die Änderbarkeit in der DTD mag ein hoher FAN-OUT-Wert zunächst wünschenswert und von Vorteil sein, doch in Bezug auf ein XML-Dokument wird die Änderung eines Elements mit einem hohen FAN-OUT umfangreicher sein, sofern die Änderung Auswirkung auf die Gültigkeit des Dokuments hat. Somit hat der FAN-OUT auch Einfluß auf die Änderbarkeit. Das Wiederverwenden von Elementen ist für die Benutzbarkeit eher von Vorteil. Dies wird an dieser Stelle aber eher als irrelevant betrachtet, da die Benutzbarkeit mehr durch das jeweilige Inhaltsmodell und die gesamte Strukturkomplexität der DTD beeinflußt wird.

Ebenso wie die Werte für den FAN-IN, können die FAN-OUT-Werte in einem Diagramm (siehe Anhang B) dargestellt werden. Diese werden aber nicht absteigend sondern aufsteigend sortiert. Das hat hier den Hintergrund, daß ein Element mit einem niedrigen FAN-IN eher einen hohen FAN-OUT besitzt und umgekehrt. Diese Tatsache wird durch die beiden unterschiedlichen Kurvenverläufe hervorgehoben. Anhand des FAN-OUT-Diagramms ist ebenso die Anzahl der Elemente der DTD und die Anzahl der Elemente mit gleichem FAN-OUT ablesbar. In der DTD sollte nur ein Element mit einem FAN-OUT von Null vorkommen. Der FAN-OUT von Null zeigt, daß dieses Element nicht im Inhalt eines anderen Elements vorkommt. Das ist in der DTD in der Regel bei einem Wurzelelement der Fall. Sollten mehrere Elemente mit einem FAN-OUT von Null in der DTD vorhanden sein, so spricht dies für mehrere Wurzelkandidaten. In Beispiel 4 im Anhang B.4 existiert ein Ausnahmefall. In diesem Beispiel gibt es eine rekursive Elementverschachtelung, so daß das Wurzelelement auch im Inhalt eines anderen Elements enthalten ist. Aus diesem Grund ist das Wurzelelement nicht eindeutig aus dem FAN-OUT zu bestimmen. Wie schon erwähnt, verhalten sich FAN-IN und FAN-OUT zwar nicht grundsätzlich umgekehrt, aber in den überwiegenden Fällen. Während die Blattknoten des DTD-Graphen einen geringen FAN-IN haben, haben sie meistens einen hohen bzw. auch den höchsten FAN-OUT-Wert.

Bei der Bestimmung des FAN-OUT einer modularisierten DTD muß von der Gesamt-DTD ausgegangen werden, da die Erreichbarkeit der Elemente die Voraussetzung für die Bestimmung des FAN-OUT ist. Hier ist es nicht möglich, zuerst den FAN-OUT der Elemente der Teil-DTDs zu bestimmen, denn bei der einzelnen Betrachtung ist die Wiederverwendung der Elemente außerhalb dieser Teil-DTDs nicht erkennbar.

Die FAN-OUT-Metrik kann ähnlich der FAN-IN-Metrik auf die XML-Dokumente angewendet werden. Hiermit kann beispielsweise festgestellt werden, ob das zu überprüfende Element im XML-Dokument verwendet wird und in welchen anderen Elementen es tatsächlich auftritt. Um die Werte zwischen der DTD und dem XML-Dokument vergleichbar zu machen, ist hier ebenfalls nicht die Häufigkeit des Auftretens in den anderen Elementen, sondern nur das Vorhandensein zu bewerten. Somit würde der FAN-OUT eines Elements maximal dem Wert der DTD entsprechen, falls es tatsächlich in allen möglichen Elementen mindestens einmal auftritt.

4.3.5.1 Zusammenhang zwischen FAN-IN und FAN-OUT

Der Zusammenhang zwischen den beiden Metriken wird anhand des Beispiels 3 aus dem Anhang B.3 erläutert. Die Werte für den FAN-IN und FAN-OUT dieser DTD sind bereits im Anhang in einer Tabelle zusammengestellt. Der FAN-IN der Elemente läßt sich direkt aus den Elementdeklarationen in der DTD bestimmen. Bei der Bestimmung des FAN-OUT eines Elements müßte jedes andere Element durchsucht werden. Demnach erscheint zunächst die Ermittlung des FAN-OUT aufwendiger.

Einfacher gestaltet sich die Ermittlung des FAN-OUT durch die Erreichbarkeits- oder auch Adjazenzmatrix (siehe [Tur96]). Die Erreichbarkeitsmatrix für das Beispiel 3 aus dem Anhang B.3 ist in der Abbildung 4.2 dargestellt. Diese Matrix kann direkt aus den einzelnen Elementdeklarationen aufgestellt werden. Zur besseren Übersicht wurden die Elemente der Beispiel-DTD von 0 bis n durchnummeriert. Die Element-ID wird in der Abbildung 4.2 als Zeilen- und Spaltennummer genutzt. Die Matrix ist quadratisch, wobei die Anzahl der Zeilen und Spalten mit der Anzahl der Elementdeklarationen in der DTD übereinstimmt. Die Zeilen dieser Matrix entsprechen dabei den Elementdeklarationen, wobei hier genau die i-te Zeile der i-ten Elementdeklaration entspricht. Die Spalten stehen für die Elemente im Inhaltsmodell. Eine 1 in der Matrix an der Stelle ij bedeutet, daß im Inhaltsmodell des i-ten Elements das j-te Element enthalten ist. Das Auftreten jedes Elements wird entsprechend der FAN-IN-Definition dabei nur einmal notiert, auch wenn das Element mehrfach im Inhaltsmodell vorkommt.

In dieser Abbildung ist zu sehen, daß die Zeilensummen den jeweiligen FAN-IN des Elements wiedergeben, der in der letzten Spalte notiert ist. Wenn in der Matrix die Spaltensumme gebildet wird, erhält man den FAN-OUT für die Elemente. Die Summe aller FAN-IN- und aller FAN-OUT-Werte ist gleich und entspricht damit dem Gradsummensatz der Graphentheorie.

Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	**FAN-IN**
0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	9
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
5	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	6
11	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	5
12	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	5
13	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	4
14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	4
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	4
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FAN-OUT	0	7	1	1	1	1	2	1	1	1	1	1	2	2	2	4	1	1	6	5	3	**44**

Abbildung 4.2: Erreichbarkeitsmatrix Beispiel 3 (aus Anhang B.3)

Diese Summe ist mit der Anzahl aller Elemente in den Inhaltsmodellen identisch, wobei das mehrfache Auftreten eines Elements im Inhaltsmodell einer Elementdeklaration nur einfach bewertet wird. An diesem Beispiel ist zu erkennen, daß sich aus den erstellten Daten für den FAN-IN gleichfalls der FAN-OUT ermitteln läßt.

Weiterhin ist zu sehen, daß die beiden Metriken unabhängig sind. Im Anhang B sind für die Beispiele die jeweiligen Werte für jedes Element gegenübergestellt und damit direkt vergleichbar. Zusätzlich wurden die Daten vom FAN-IN und FAN-OUT in einem Diagramm dargestellt. In diesem Diagramm entspricht die Elementachse (x-Achse) der Element-ID, denen die FAN-Werte entsprechend zugeordnet werden. An dem Kurvenverlauf in allen Beispielen ist deutlich zu erkennen, daß ein hoher FAN-IN immer mit einem niedrigen FAN-OUT zusammenfällt und umgekehrt.

4.3.6 Fazit

In der Zusammenfassung des Kapitels 4.4 in der Tabelle 4.3 sind alle Metriken noch einmal aufgelistet. Jede dieser Metriken beschreibt eine wesentliche direkt oder indirekt meßbare Eigenschaft der DTD, wobei die bewerteten DTD-Eigenschaften unabhängig voneinander sind.

Nach der ausführlichen Betrachtung der Softwaremetrie in Kapitel 3 soll die Einordnung der DTD-Metriken an dieser Stelle nicht fehlen. Die DTD-Metriken lassen sich wie erwähnt durch direkte und indirekte (modellbezogene) Softwaremessung ermitteln. Metriken, die auf dem DTD-Graph basieren, sind hierbei indirekt ermittelbar. Die Maße sind dabei objektiv, deskriptiv und statisch. Nach der im Abschnitt 3.1.3 beschriebenen Klassifikation der Metriken (siehe Tabelle 3.1 auf Seite 32) sind die DTD-Metriken den Produktmetriken zuzuordnen. Die durch die DTD-Metriken bewerteten Eigenschaften zählen nach der Einteilung von Norman Fenton (siehe Abschnitt 3.1.3) zu den internen Attributen.

4.4 Zusammenfassung

In diesem Abschnitt sollen spezielle Hintergründe der entwickelten Metriken und auch der Grundeinheiten zusammenfassend beschrieben werden. In der Tabelle 4.3 sind die wichtigsten Grundeinheiten sowie alle definierten Metriken zusammenhängend aufgelistet.

Der in Abschnitt 3.1.2 beschriebene Qualitätsstandard ISO 9126 wurde als Grundlage für die Einschätzung der qualitativen Merkmale der XML-DTD und somit auch für die XML-Dokumente genommen. Bei der Bewertung der DTD sind natürlich nicht alle genannten Qualitätsmerkmale relevant. Die Bewertung der DTD wurde hier auf die Merkmale Änderbarkeit und Benutzbarkeit sowie deren Untermerkmale beschränkt, da diese qualitativen Kriterien ausschlaggebend bei der Verwendung der XML-DTDs und der XML-Dokumente sind. Die Benutzbarkeit spielt insofern eine Rolle, da ein Anwender die DTD kennen muß, wenn er ein gültiges XML-Dokument schreiben will. Das gleiche gilt auch für denjenigen, welcher Änderungen an den DTDs vornimmt, wobei hier zusätzlich noch die Auswirkungen auf die gültigen Dokumente berücksichtigt werden müssen. In beiden Fällen sind für die genannten Qualitätsmerkmale die Größe und die strukturellen Eigenschaften die bestimmenden Einflußfaktoren. Diese werden quantitativ durch die definierten Metriken bewertet.

Die Dokument-orientierte Ausrichtung der DTD ist der Grund der hauptsächlich strukturellen Bewertung. Bei dem Vergleich zwischen der XML-DTD und dem XML-Schema in Abschnitt 2.3 wurde bereits erwähnt, daß dies nicht als Nachteil zu sehen ist. Bis auf wenige Ausnahmen sind die strukturellen Eigenschaften der DTD und des XML-Schema gleich, so daß die hier definierten Metriken sich auch auf XML-Schema anwenden lassen. Die Ausnahmen sind in Abschnitt 2.3 beschrieben worden und wurden bei der Entwicklung der Metriken bereits berücksichtigt.

Die DTD-Metriken können auch auf XML-Dokumente bzw. -kollektionen angewendet werden. Dabei muß berücksichtigt werden, daß zwischen der DTD und den XML-Dokumenten ein prinzipieller Unterschied besteht. Bei der modellbezogenen Bewertung wird die DTD durch einen gerichteten Graphen repräsentiert, in dem die Elementwiederholungen und -rekursionen durch Schlingen und Kreise im Graphen dargestellt sind. Damit ist dieser Graph kein Baum. Dagegen werden die XML-Dokumente als Dokumentbaum repräsentiert ([W3C01c]). Dies hat Auswirkungen auf die strukturellen Bewertungen der XML-Dokumente, welche im Zusammenhang mit den Metriken beschrieben sind.

Tabelle 4.3: Zusammenfassung der DTD-Bewertung

Kürzel	**Beschreibung**	**Charakteristik**
E	Anzahl der Elemente	Grundeinheit
A	Anzahl der Attribute	Grundeinheit
EN	Anzahl der Entities	Grundeinheit
EN_A	Anzahl der allgemeinen Entities	Grundeinheit
EN_P	Anzahl der Parameter-Entities	Grundeinheit
N	Anzahl der Notationen	Grundeinheit
U_{DTD}	Umfang der DTD	Codemetrik
U_D	Umfang des XML-Dokuments	Codemetrik
SK	Strukturkomplexität der DTD	Komplexitätsmetrik (DTD)
ST_{DTD}	Strukturtiefe der DTD	Komplexitätsmetrik (DTD)
ST_E	Strukturtiefe des Elements	Komplexitätsmetrik (Element)
$FAN\text{-}IN$	Strukturbreite des Elements	Komplexitätsmetrik (Element)
$FAN\text{-}OUT$	Wiederverwendung des Elements	Komplexitätsmetrik (Element)

Gerade durch den vielfältigen Einsatz von XML (siehe Abschnitt 2.1.3) besteht das Interesse, die unterschiedlichsten Eigenschaften von XML bzw. der XML-DTDs zu bewerten. Hierin liegt aber auch die Schwierigkeit, denn beim Einsatz von XML in den unterschiedlichen Gebieten werden jeweils andere Prioritäten gesetzt. Das spiegelt sich auf der einen Seite in der Datenmodellierung wider. DTDs für den Datenaustausch werden sich stark von denen für die Textpräsentation unterscheiden. Auf der anderen Seite werden durch die anwendungsspezifische Sicht jeweils andere Eigenschaften als wesentlich betrachtet, die zu bewerten sind. Die XML-Empfehlung [W3C00b] als Basis für die Beschreibung der Daten ist hier die einzige Gemeinsamkeit bei der Verwendung von XML in den unterschiedlichen Gebieten und wurde deshalb als Grundlage für die Bewertung verwendet.

Zu der genannten Problematik kommt noch erschwerend hinzu, daß es nicht wie beim Datenbank- oder Softwareentwurf eine allgemeingültige Vorgehensweise bei der Datenmodellierung bzw. dem DTD-Entwurf gibt. Bekannt ist diese Diskussion[3] unter dem Thema "*Elemente versus Attribute*". Darauf wurde im Abschnitt 2.2.2 bereits eingegangen und aus diesem Grund beschränken sich die zu bewertenden wesentlichen Eigenschaften auf allgemeine, die in allen DTDs vorkommen und nicht anwendungsspezifisch sind. Es ist an dieser Stelle nicht sinnvoll, wie z.B. im Abschnitt 3.2.3 bei den Datenbankmetriken, die Elemente und Attribute in Beziehung zu setzen, um daraus eine bestimmte Eigenschaft abzuleiten und diese allgemein zu interpretieren. Dies kann nur einer Betrachtung aus dem Blickwinkel einer spezifischen Anwendungsdomäne vorbehalten sein. Dies wird auch im Abschnitt 4.2 im Zusammenhang mit den Grundeinheiten beschrieben. So lassen sich beispielsweise verschiedene Grundeinheiten aus den Attributen ableiten, die in der Bewertung entweder nur bei den Grundeinheiten mit der einfachen Auftrittshäufigkeit aufgeführt werden oder auch in den Anforderungen entsprechende spezielle Metriken mit eingehen können. Eine anwendungsspezifische Sicht auf die DTD bzw. XML-Dokumente ist hier aber nicht Gegenstand der Betrachtung. Die vollständige Zusammenfassung der Grundeinheiten ist in Tabelle 4.1 auf Seite 50 zu finden.

Auf eine Einschätzung, was als groß und komplex betrachtet wird, ist bis auf zwei Ausnahmen im Zusammenhang mit den Metriken verzichtet worden. Der Hintergrund ist, daß diese Einschätzung ein subjektives Ergebnis der jeweiligen Betrachtung ist, auch wenn hier eine ausreichend repräsentative Menge von DTDs und XML-Dokumenten getestet worden wäre. Beispielsweise wird bei dieser Einschätzung schon ein großer Unterschied zwischen DTD bzw. Dokumenten bestehen, welche manuell oder automatisch (z.B. für den Datenaustausch) erstellt sind. Im ersten Fall ist die Einschätzung weiterhin stark vom Wissensstand des Benutzers abhängig und im zweiten Fall beeinflussen andere Kriterien diese Einschätzung.

Aus unterschiedlichen Gründen sind die einzelnen Maße nicht in einer Gesamtbewertung bzw. einer Gesamtmetrik für die DTD zusammengefaßt worden, wie dies beispielsweise bei der Bewertung des ER-Modells in [GJP00] (siehe auch Abschnitt 3.2.3) mit einer Regressionsanalyse durchgeführt wird. In diesem Fall wurde das ER-Modell bewertet, wobei man hier davon ausgehen kann, daß wenn vielleicht auch prinzipielle Unterschiede bei der Gewichtung der Einzelmaße möglich sind, doch grundsätzlich in jedem ER-Modell aufgrund der gleichartigen Verwendung des Modells dieselben Eigenschaften als wesentlich angesehen werden können. Gerade durch die unterschiedliche Verwendung der DTD und auch XML-Dokumente, hier also nicht beschränkt auf RDBS, wird die Gesamtbetrachtung aller allgemeinen Maße als nicht sinnvoll betrachtet. Wenn diese Bewertung sinnvoll angewendet werden soll, muß zunächst die Menge der DTDs entsprechend ihrer Verwendung klassifiziert werden. Für diese Teilmengen können dann aufgrund der gleichartigen Verwendung die wesentlichen Eigenschaften bestimmt werden, die beispielsweise in eine Gesamtbewertung bzw. auch ein Gesamtmaß eingehen sollen.

Einige weitere Gründe, die hier ausschlaggebend dafür sind, daß die einzelnen Komplexitätswerte

[3] http://www.oasis-open.org/cover/elementsAndAttrs.html

nicht zu einer Gesamtmetrik zusammengefaßt werden, sollen an dieser Stelle auch nicht unerwähnt bleiben. Aus den unterschiedlichen indirekten Metriken auf der DTD- und der Element-Ebene lassen sich einige Eigenschaften bzw. besondere strukturelle Eigenschaften ablesen, die in einer Gesamtmetrik nicht mehr erkennbar sind. Würde man beispielsweise bei der Bewertung der FAN-Werte nur die Summe darstellen, so wären die speziellen Eigenschaften des Wurzelelements und der Blattelemente, die gerade an den unterschiedlichen FAN-Werten auf Elementebene erkennbar sind, nicht mehr zu erkennen. Außerdem ist die Bewertung einer einzelnen Eigenschaft durch eine Metrik eine Forderung der Maßtheorie, was im Abschnitt 3.2.1.3 im Zusammenhang mit der Henry-Kafura-Metrik bereits erwähnt wurde. An einer Gesamtmetrik ist desweiteren nicht erkennbar, welche der Grundeigenschaften die Größe der Gesamtkomplexität maßgeblich beeinflussen. Zusätzlich werden innerhalb der Gesamtmetrik die Einzelmetriken unterschiedlich gewichtet, da nicht jede Metrik für die Gesamtkomplexität gleichbedeutend ist. Dafür müßten an dieser Stelle zunächst umfangreiche Untersuchungen durchgeführt werden, was aber hier nicht das eigentliche Problem darstellt. Gerade die Gewichtung, durch welches Verfahren der Datenanalyse auch immer, stellt an dieser Stelle einen subjektiven Eingriff in die Bewertung dar und ist bei den Grundmetriken auf keinen Fall erwünscht.

Im Zusammenhang mit der Beschreibung der Metriken wurde auf die Verwendung des Maßes bei modularisierten DTDs und XML-Dokumenten eingegangen. Im Fall einer modularisierten DTD ist es meistens am einfachsten, ähnlich wie bei der Dokumentverarbeitung, die gesamte DTD zu betrachten. Bei der Anwendung der Metriken auf XML-Dokumente müssen einige Einschränkungen beachtet werden, um vergleichbare Werte zu erhalten.

Wie im Anhang B in der Zusammenfassung in Abschnitt B.5 zu sehen, ist es sinnvoll, die ermittelten DTD-Besonderheiten mit anzugeben. Dazu zählen beispielsweise die Anzahl der Elementrekursionen und auch die Anzahl der Wurzelelemente. Diese Werte weisen konkret auf einige Besonderheiten bei der Deklaration innerhalb der DTD hin, was in der Tabelle B.1 auf Seite 99 an den Beispielen 2 und 4 zu sehen ist. Auch angegeben ist die Anzahl der Elemente mit einem FAN-IN von Null, wobei diese Eigenschaft kennzeichnend für einen Blattknoten im DTD-Graph ist. Die Besonderheiten wurden im Zusammenhang mit der Definition der Metriken beschrieben.

Das Ziel der Arbeit war das Aufstellen von allgemeinen qualitativen Kriterien für die DTDs, die geeignet quantitativ bewertet werden sollten. Die qualitativen Eigenschaften basieren auf den allgemeinen Software-Qualitätsmerkmalen, wie sie in Abschnitt 3.1.2 beim ISO 9126 Standard beschrieben wurden. Dabei wurden die Qualitätsmerkmale Änderbarkeit und Benutzbarkeit als wesentlich angesehen und lassen sich durch die in der Tabelle 4.3 zusammengefaßten Grundeinheiten und Metriken quantitativ bewerten.

Kapitel 5

Schlußbetrachtung

In diesen Kapitel werden die wichtigsten Aspekte der Arbeit zusammengefaßt und ein Ausblick für die entwickelten Metriken gegeben. Der Ausblick umfaßt eine kurze Darstellung unterschiedlicher Verwendungsmöglichkeiten der Metriken sowie zukünftiger Arbeiten.

5.1 Zusammenfassung

In dieser Arbeit wurden Metriken für die Bewertung von XML-Dokumentkollektionen entwickelt. Im Mittelpunkt der Bewertung stand hierbei eine allgemeine Sicht auf die Dokumente, also unabhängig von speziellen Anwendungsdomänen.

Ausgangspunkt der Bewertung war die aktuelle XML-Empfehlung [W3C00b]. Es wurden zunächst die Eigenschaften der Auszeichnungssprache untersucht, die als Grundlage einer allgemeinen Bewertung genutzt werden können. Hierbei hat sich herausgestellt, daß es wenig Sinn macht, die einzelnen Dokumente zu bewerten, sondern vielmehr ihre Grundstruktur, welche in den DTDs definiert ist. Aus diesem Grund basiert die Bewertung der XML-Dokumentkollektionen auf der Bewertung der DTDs.

Desweiteren wurden bekannte Metriken der Softwaremetrie untersucht, um festzustellen, inwieweit diese für die Bewertung der DTDs genutzt werden können. Die Vermutung, daß sich diese Metriken nicht direkt für die Bewertung der DTD nutzen lassen, bestätigte sich. Es zeigte sich aber, daß die Grundideen einiger Metriken für die DTD-Metriken genutzt werden können, da teilweise ähnliche strukturelle Eigenschaften, die sich durch Graphen darstellen lassen, bewertet werden.

Die Bewertung der DTDs läßt sich nach der Art der Messung in direkt und indirekt meßbare Eigenschaften unterteilen. Zu den direkt meßbaren Eigenschaften zählen zunächst die Deklarationen innerhalb der DTD, die in der Arbeit als Grundeinheiten bezeichnet wurden. Basierend auf den Grundeinheiten wurde das Umfangsmaß definiert. Weiterhin lassen sich die Grundeinheiten zur Definition von weiteren Metriken für anwendungsspezifische Bewertungen nutzen, die hier aber nicht Gegenstand der Betrachtung waren. Die durch die DTD definierte Dokumentstruktur ist neben der Größe der DTD eine weitere wesentliche Eigenschaft, die bewertet wurde und zu den indirekt meßbaren Eigenschaften gezählt wird. Die Dokumentstruktur läßt sich durch einen Baum-ähnlichen Graphen darstellen und wird durch den entwickelten DTD-Graph repräsentiert. An diesem Graph ist zu erkennen, daß die Dokumentstruktur durch unterschiedliche strukturelle Eigenschaften geprägt ist. Diese Eigenschaften lassen sich nach dem Umfang der Bewertung der Graphenebene oder der Knotenebene zuordnen. Zu den Eigenschaften auf der Graphenebene zählen die Strukturkomplexität und die Strukturtiefe und auf der Knotenebene die Strukturtiefe, der Ausgangsgrad (Strukturbreite – FAN-IN) und der Eingangsgrad (Elementwiederverwendung – FAN-OUT). Diese Eigenschaften werden neben dem DTD-Umfang als die wesentlichen Eigenschaften in Bezug auf die Änderbarkeit und Benutzbarkeit der DTD für XML-Dokumentkollektionen angesehen und werden durch die entwickelten Struktur- bzw. Komplexitätsmetriken bewertet.

Die Metriken lassen sich teilweise auf die XML-Dokumente anwenden. Der Hintergrund dieser Bewertung ist der Vergleich, inwieweit die in der DTD deklarierten Bestandteile in den Dokumenten auftreten. Bei diesem Vergleich sind einige Einschränkungen bzw. Besonderheiten zu beachten. Beispielsweise müssen die in der DTD deklarierten Elemente in den Dokumenten nicht unbedingt auftreten, können aber auch wiederholt vorkommen. Weiterhin unterscheidet sich der Dokumentgraph (Baum) vom DTD-Graph. Dies sind auch die Gründe, weswegen die Metriken nicht für eine direkte und ausschließliche Dokumentbewertung verwendet werden können. Diese Bewertung unterscheidet sich aber auch insofern von der DTD-Bewertung, da die DTD-Bewertung eine Struktur-orientierte Bewertung darstellt, während die XML-Dokumentbewertung mehr auf eine Inhalts-orientierte Bewertung ausgerichtet ist.

5.2 Ausblick

Mit den in dieser Arbeit entwickelten Metriken lassen sich aufgrund der Bewertung der DTD Merkmale der XML-Dokumentkollektionen bestimmen. Beispielsweise ist die Strukturkomplexität ein ausschlaggebendes Maß für die mögliche Unterschiedlichkeit in Größe und Art der Dokumente einer Kollektion. Alle Maße können als Kriterium genutzt werden, um die Verständlichkeit der DTD zu bestimmen, die Voraussetzung für Änderungen der DTD und die Erstellung gültiger Dokumente ist.

Aufgrund der Bewertung von allgemeinen Eigenschaften, wie die Größe und die Struktur, lassen sich die Metriken unabhängig von einer speziellen Anwendungsdomäne verwenden. Dies ist auch als eine Voraussetzung für die Vergleichbarkeit der Bewertungen von DTDs gleicher sowie unterschiedlicher Anwendungsdomänen zu sehen. Wie in der Arbeit mehrfach erwähnt, existiert keine allgemeingültige Vorgehensweise für den DTD-Entwurf. Dies ist auch ein Grund dafür, daß sich die DTD-Entwürfe gleicher Anwendungsdomänen sehr stark voneinander unterscheiden können. Die Metriken können hier beispielsweise verwendet werden, um ähnliche DTDs nach ihren Entwurfseigenschaften hin zu analysieren.

Die XML-Dokumentkollektionen werden im Laufe der Zeit geändert, so daß durch Korrekturen oder auch Erweiterungen sogenannte Versionen entstehen, beispielsweise wie die Versionen der HTML-DTDs. Hier ist es möglich, die Versionen mittels der Metriken zu analysieren, um im nachhinein Unterschiede zwischen den einzelnen Versionen zu ermitteln.

Weiterhin können einige Metriken genutzt werden, um den Grad der Übereinstimmung zwischen den deklarierten Bestandteilen in der DTD und der Verwendung bzw. dem Auftreten in den XML-Dokumenten zu bestimmen. Hierdurch läßt sich feststellen, ob und mit welcher Häufigkeit die deklarierten Bestandteile der DTD in den Dokumenten auftreten.

Auch wenn die Metriken anhand einiger Beispiel-DTDs evaluiert wurden, ist dies natürlich noch nicht Beweis genug für die Zuverlässigkeit bzw. Gültigkeit dieser Metriken. Vorteil ist hier, daß die Metriken auf bekannten Metriken und Modellen basieren und diese bereits validiert wurden. Das läßt die Vermutung zu, daß sich die entwickelten Metriken ähnlich verhalten, was aber noch an umfangreichen Test zu zeigen ist.

Neben der XML-DTD wird in Zukunft XML-Schema als Schemasprache von XML mehr an Bedeutung zunehmen, so daß die Dokumentstrukturen basierend auf XML-Schema auch bewertet werden sollen. Dies ist bei der Entwicklung der Metriken schon berücksichtigt worden. Da die strukturellen Eigenschaften ähnlich sind, werden die vorhandenen DTD-Metriken keine grundlegenden Anpassungen erfordern. Es ist eher von einer Erweiterung der Metriken auszugehen, da XML-Schema ausdrucksstärker ist als die DTD.

Für die Bewertung der Struktureigenschaften von XML-Dokumentkollektionen bleibt in Zukunft zu hoffen, daß im Zusammenhang mit der Entwicklung der Schemasprachen für XML auch ein allgemeingültiger Konsens für den Entwurf von Schemabeschreibungen gefunden wird. Zur Zeit ist es aufgrund unterschiedlicher Auffassungen beim DTD-Entwurf nicht möglich, für alle Schemabeschreibungen konkret zu bestimmen, welche Deklarationen die Dokumentstruktur und die Dokumentinformation definieren. Sollte dies vielleicht mit der Entwicklung von XML einmal geklärt werden, könnten die Metriken die Schemabeschreibungen und somit auch die Dokumentkollektionen noch besser bewerten.

Anhang A

XML-Regeln

Dieser Anhang ist ein Auszug der EBNF-Regeln aus der XML 1.0 (Second Edition) [W3C00b]. Entsprechend der aktuellen Fehlerliste (XML 1.0 Second Edition Specification Errata) zu [W3C00b] sind die Regeln [6], [8] und [28] korrigiert und die Regel [28b] hinzugefügt worden. Damit die Regeln im Originaltext schneller wiedergefunden werden können, sind die notwendigen Inhaltsangaben und Kommentare nicht entfernt worden. Die durch "`/* */`" gekenzeichneten Abschnitte enthalten einfache Kommentare. Abschnitte mit `[WFC: ... ]` – Wohlgeformtheitsbeschränkung – oder `[VC: ... ]` – Gültigkeitsbeschränkung – identifizieren durch einen Namen eine mit einer Produktion verknüpfte Beschränkung eines wohlgeformten oder gültigen Dokuments. Diese sind im Originaltext [W3C00b] nachzulesen. Abschnitte aus dem Originaltext, die keine Regeln enthalten, wurden weggelassen, was durch Punkte gekennzeichnet ist.

...

2 Documents

2.1 Well-Formed XML Documents

Document

```
[1] document ::= prolog element Misc*
```

2.2 Characters

Character Range

```
[2] Char ::= #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
    /* any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. */
```

2.3 Common Syntactic Constructs

White Space

```
[3] S ::= (#x20 | #x9 | #xD | #xA)+
```

Names and Tokens

```
[4] NameChar ::= Letter | Digit | ’.’ | ’-’ | ’_’ | ’:’ | CombiningChar | Extender
[5] Name ::= (Letter | ’_’ | ’:’) (NameChar)*
```

Errata as of 2001-05-24 – E20 Substantive

Change productions [6] Names and [8] Nmtokens to use #x20 (a single space character) instead of S:

```
[6] Names ::= Name (#x20 Name)*
[7] Nmtoken ::= (NameChar)+
[8] Nmtokens ::= Nmtoken (#x20 Nmtoken)*
```

Literals

```
[9] EntityValue ::= ’"’ ([^%&"] | PEReference | Reference)* ’"’
    | "’" ([^%&’] | PEReference | Reference)* "’"
[10] AttValue ::= ’"’ ([^<&"] | Reference)* ’"’ | "’" ([^<&’] | Reference)* "’"
[11] SystemLiteral ::= (’"’ [^"]* ’"’) | ("’" [^’]* "’")
[12] PubidLiteral ::= ’"’ PubidChar* ’"’ | "’" (PubidChar - "’")* "’"
[13] PubidChar ::= #x20 | #xD | #xA | [a-zA-Z0-9] | [-’()+,./:=?;!*#@$_%]
```

2.4 Character Data and Markup

Character Data

```
[14] CharData ::= [^<&]* - ([^<&]* ']]>' [^<&]*)
```

2.5 Comments

Comments

```
[15] Comment ::= '<!-' ((Char - '-') | ('-' (Char - '-')))* '->'
```

2.6 Processing Instructions

Processing Instructions

```
[16] PI ::= '<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'
[17] PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
```

2.7 CDATA Sections

CDATA Sections

```
[18] CDSect ::= CDStart CData CDEnd
[19] CDStart ::= '<![CDATA['
[20] CData ::= (Char* - (Char* ']]>' Char*))
[21] CDEnd ::= ']]>'
```

2.8 Prolog and Document Type Declaration

Prolog

```
[22] prolog ::= XMLDecl? Misc* (doctypedecl Misc*)?
[23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
[24] VersionInfo ::= S 'version' Eq ("'" VersionNum "'" | '"' VersionNum '"')
[25] Eq ::= S? '=' S?
[26] VersionNum ::= ([a-zA-Z0-9_.:] | '-')+
[27] Misc ::= Comment | PI | S
```

Document Type Definition

Errata as of 2001-06-13 – E21 Substantive

Add a new production [28b] and modify production [28] to refer to it :

```
[28] doctypedecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('[' intSubset ']' S?)? '>'
     [VC: Root Element Type][WFC: External Subset]
[28a] DeclSep ::= PEReference | S
      [WFC: PE Between Declarations]
[28b] intSubset ::= (markupdecl | DeclSep)*
[29] markupdecl ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment
     [VC: Proper Declaration/PE Nesting][WFC: PEs in Internal Subset]
```

External Subset

```
[30] extSubset ::= TextDecl? extSubsetDecl
[31] extSubsetDecl ::= ( markupdecl | conditionalSect | DeclSep)*
```

Standalone Document Declaration

```
[32] SDDecl ::= S 'standalone' Eq (("'" ('yes' | 'no') "'") | ('"' ('yes' | 'no') '"'))
     [VC: Standalone Document Declaration]
```

...

2.12 Language Identification

(Productions 33 through 38 have been removed.)

3 Logical Structures

Element

```
[39] element ::= EmptyElemTag | STag content ETag
     [WFC: Element Type Match][VC: Element Valid]
```

3.1 Start-Tags, End-Tags, and Empty-Element Tags

Start-tag

```
[40] STag ::= '<' Name (S Attribute)* S? '>'
     [WFC: Unique Att Spec]
```

```
[41] Attribute ::= Name Eq AttValue
     [VC: Attribute Value Type][WFC: No External Entity References]
     [WFC: No < in Attribute Values]
```

End-tag

```
[42] ETag ::= '</' Name S? '>'
```

Content of Elements

```
[43] content ::= CharData? ((element | Reference | CDSect | PI | Comment) CharData?)*
```

Tags for Empty Elements

```
[44] EmptyElemTag ::= '<' Name (S Attribute)* S? '/>'
     [WFC: Unique Att Spec]
```

3.2 Element Type Declarations

Element Type Declaration

```
[45] elementdecl ::= '<!ELEMENT' S Name S contentspec S? '>'
     [VC: Unique Element Type Declaration]
[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed | children
```

3.2.1 Element Content

Element-content Models

```
[47] children ::= (choice | seq) ('?' | '*' | '+')?
[48] cp ::= (Name | choice | seq) ('?' | '*' | '+')?
[49] choice ::= '(' S? cp ( S? '|' S? cp )+ S? ')'
     [VC: Proper Group/PE Nesting]
[50] seq ::= '(' S? cp ( S? ',' S? cp )* S? ')'
     [VC: Proper Group/PE Nesting]
```

3.2.2 Mixed Content

Mixed-content Declaration

```
[51] Mixed ::= '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*' | '(' S? '#PCDATA' S? ')'
     [VC: Proper Group/PE Nesting] [VC: No Duplicate Types]
```

3.3 Attribute-List Declarations

Attribute-list Declaration

```
[52] AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'
[53] AttDef ::= S Name S AttType S DefaultDecl
```

3.3.1 Attribute Types

Attribute Types

```
[54] AttType ::= StringType | TokenizedType | EnumeratedType
[55] StringType ::= 'CDATA'
[56] TokenizedType ::= 'ID'[VC: ID][VC: One ID per Element Type] [VC: ID Attribute Default]
     | 'IDREF'[VC: IDREF]
     | 'IDREFS'[VC: IDREF]
     | 'ENTITY'[VC: Entity Name]
     | 'ENTITIES'[VC: Entity Name]
     | 'NMTOKEN'[VC: Name Token]
     | 'NMTOKENS'[VC: Name Token]
[57] EnumeratedType ::= NotationType | Enumeration
[58] NotationType ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
     [VC: Notation Attributes][VC: One Notation Per Element Type]
     [VC: No Notation on Empty Element]
[59] Enumeration ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'
     [VC: Enumeration]
```

3.3.2 Attribute Defaults

Attribute Defaults

```
[60] DefaultDecl ::= '#REQUIRED' | '#IMPLIED' | (('#FIXED' S)? AttValue)
     [VC: Required Attribute][VC: Attribute Default Legal]
     [WFC: No < in Attribute Values][VC: Fixed Attribute Default]
```

...

3.4 Conditional Sections

Conditional Section

```
[61] conditionalSect ::= includeSect | ignoreSect
[62] includeSect ::= '<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
     [VC: Proper Conditional Section/PE Nesting]
[63] ignoreSect ::= '<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>'
     [VC: Proper Conditional Section/PE Nesting]
[64] ignoreSectContents ::= Ignore ('<![' ignoreSectContents ']]>' Ignore)*
[65] Ignore ::= Char* - (Char* ('<![' | ']]>') Char*)
```

4 Physical Structures

4.1 Character and Entity References

Character Reference

```
[66] CharRef ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
     [WFC: Legal Character]
```

Entity Reference

```
[67] Reference ::= EntityRef | CharRef
[68] EntityRef ::= '&' Name ';'
     [WFC: Entity Declared][VC: Entity Declared][WFC: Parsed Entity][WFC: No Recursion]
[69] PEReference ::= '%' Name ';'
     [VC: Entity Declared][WFC: No Recursion][WFC: In DTD]
```

4.2 Entity Declarations

Entity Declaration

```
[70] EntityDecl ::= GEDecl | PEDecl
[71] GEDecl ::= '<!ENTITY' S Name S EntityDef S? '>'
[72] PEDecl ::= '<!ENTITY' S '%' S Name S PEDef S? '>'
[73] EntityDef ::= EntityValue | (ExternalID NDataDecl?)
[74] PEDef ::= EntityValue | ExternalID
```

...

4.2.2 External Entities

External Entity Declaration

```
[75] ExternalID ::= 'SYSTEM' S SystemLiteral | 'PUBLIC' S PubidLiteral S SystemLiteral
[76] NDataDecl ::= S 'NDATA' S Name
     [VC: Notation Declared]
```

4.3 Parsed Entities

4.3.1 The Text Declaration

Text Declaration

```
[77] TextDecl ::= '<?xml' VersionInfo? EncodingDecl S? '?>'
```

4.3.2 Well-Formed Parsed Entities

Well-Formed External Parsed Entity

```
[78] extParsedEnt ::= TextDecl? content
```

4.3.3 Character Encoding in Entities

Encoding Declaration

```
[80] EncodingDecl ::= S 'encoding' Eq ('"' EncName '"' | "'" EncName "'")
```

```
[81] EncName ::= [A-Za-z] ([A-Za-z0-9._] | '-')*
         /* Encoding name contains only Latin characters */
```

...

4.7 Notation Declarations

Notation Declarations

```
[82] NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S? '>'
         [VC: Unique Notation Name]
[83] PublicID ::= 'PUBLIC' S PubidLiteral
```

...

B Character Classes

Characters

```
[84] Letter ::= BaseChar | Ideographic
[85] BaseChar ::= /* siehe [W3C00b] */
[86] Ideographic ::= /* siehe [W3C00b] */
[87] CombiningChar ::= /* siehe [W3C00b] */
[88] Digit ::= /* siehe [W3C00b] */
[89] Extender ::= /* siehe [W3C00b] */
```

...

Anhang B

XML-DTD-Beispiele

Die in Kapitel 4 entwickelten Metriken werden in diesem Anhang auf vier Beispiel-DTDs angewendet. Bei der Auswahl der DTDs wurde darauf geachtet, daß sie die durch die Metriken bewerteten wesentlichen Eigenschaften repräsentieren und in ihnen die erwähnten unterschiedlichen Besonderheiten bezüglich der strukturellen Aspekte auftreten.

Zunächst werden in Anhang B.1 - B.4 die DTDs aufgelistet. Jeder Zeile ist eine Zahl vorangestellt, die als Zeilenzahl für die Zuordnung der DTD-Deklarationen bei der Bewertung verwendet wird. Entsprechend des Auftretens der Elementdeklarationen werden diese der Reihenfolge nach durchnummeriert. Die Zuordnung der Zeilenzahl zur Element-ID ist jeweils im 3. Unterabschnitt der Tabelle zu entnehmen.

Für jede DTD wird im 2. Unterabschnitt der DTD-Graph dargestellt. Die Entwicklung eines DTD-Graphen aus den einzelnen Elementdeklarationen wurde in Abschnitt 4.3.2 beschrieben. An diesem lassen sich alle strukturellen Eigenschaften am besten erkennen und durch eine manuelle Bewertung nachvollziehen, was anhand der Werte in der Tabelle B.1 im Anhang B.5 überprüft werden kann.

Im jeweils 3. Unterabschnitt werden die indirekten Bewertungen der DTD auf Elementebene dargestellt. Hierzu zählen die Strukturtiefe, der FAN-IN und der FAN-OUT der Elemente. Die statischen Werte sind in einer Tabelle zusammengefaßt und werden weiterhin durch Diagramme graphisch präsentiert. Die Interpretation dieser Bewertungen ist bereits bei der Metrikdefinition in den entsprechenden Abschnitten (ST 4.3.3, FAN-IN 4.3.4, FAN-OUT 4.3.5) beschrieben worden und kann dort nachgelesen werden.

Im Abschnitt B.5 in der Tabelle B.1 sind die direkten und indirekten Bewertungen der DTDs zusammengefaßt. In der Tabelle sind zunächst die direkt meßbaren Grundeinheiten der einzelnen DTDs aufgeführt, aus denen sich der Umfang der DTD ermitteln läßt. Danach folgt die Bewertung der indirekten Metriken, wobei im jeweiligen Abschnitt im Zusammenhang mit der Definition der Metrik die Metrik und deren Interpretation bereits erläutert wurde. Hier sollen nur noch einmal kurz einige Bemerkungen zu den Metriken und Diagrammen hinzugefügt werden.

Obwohl der *Umfang (U)* der DTDs fast gleich ist, unterscheiden sie sich sehr stark in ihrer *Strukturkomplexität (SK)*. Das liegt zum einen an der unterschiedlichen Anzahl der Wiederholungsquantoren (siehe DTD-Graphen), aber auch an der Anzahl der Elementwiederholungen von Elementen mit Quantoren. Die Elementwiederholung ist zum einen gekennzeichnet durch die Summe der *FAN*-Werte, aber auch die Wiederholung der Elemente durch andere Elemente (Teilgraphwiederholung). Diese Wiederholung ist nicht gesondert bewertet worden.

Die *Strukturtiefe* (ST_{max}) auf der DTD-Ebene entspricht der maximalen Strukturtiefe im DTD-Graph vom Wurzelknoten ausgehend zu einem der Blattknoten. Die *Strukturtiefe* (ST_E) auf Elementebene wurde bereits in den einzelnen Abschnitten der Beispiel-DTDs aufgeführt. In der Tabelle B.1 wird durch $\overline{ST_E}$ die mittlere Strukturtiefe der Elemente angegeben. Die Strukturtiefen-Verteilungen der Elemente aller Beispiel-DTDs wurden in der Abbildung B.19 in einem Diagramm zusammengefaßt dargestellt.

Die *Strukturbreite* ($FAN\text{-}IN$) und die *Elementwiederverwendung* ($FAN\text{-}OUT$) wurden bereits in den einzelnen Abschnitten der Beispiel-DTDs aufgeführt. In der Tabelle B.1 sind noch einmal die maximalen *FAN*-Werte durch $FAN\text{-}IN_{max}$ und $FAN\text{-}OUT_{max}$ aufgeführt. Die Summe von $FAN\text{-}IN$ und $FAN\text{-}OUT$ in der DTD ist gleich (siehe Abschnitt 4.3.5.1) und wird durch den Wert $\sum FAN_{DTD}$ in der Tabelle repräsentiert. Da die Summe von $FAN\text{-}IN$ und $FAN\text{-}OUT$ gleich ist, sind auch die mittleren Werte der FAN-Werte gleich. Diese werden durch $\overline{FAN_{DTD}}$ in der Tabelle repräsentiert. Weiterhin sind in den beiden Diagrammen der Abbildung B.18 die FAN-Kurven der Beispiel-DTDs zusammengefaßt dargestellt.

In der Tabelle wurden weiterhin einige Besonderheiten der DTDs aufgeführt. Alle Beispiel-DTDs haben ein Wurzelelement. In den Beispiel-DTDs 1 bis 3 ist dieses durch den FAN-OUT von Null bei genau einem Element direkt erkennbar. In Beispiel 4 dagegen ist das Wurzelelement innerhalb einer Rekursion enthalten und nicht direkt erkennbar. Elemente mit einem FAN-IN von Null sind als Blattelemente bzw. -knoten direkt erkennbar. Diese Anzahl entspricht der Anzahl unterschiedlicher einfacher Blattknoten im DTD-Graph. Rekursive Elemente, wie in den Beispielen 2 und 4, sind in dieser Anzahl nicht berücksichtigt, auch wenn sie für die Ermittlung der Strukturtiefe ein Blattelement darstellen. Die Anzahl der Rekursionen ist dabei eine Größe, welche auch in die Strukturkomplexität eingegangen ist. Die Rekursion wird nach direkter und indirekter unterschieden, was in der Tabelle gezeigt ist. Die jeweilige Anzahl der Rekursionen ist hier die Auftrittshäufigkeit innerhalb des DTD-Graphen, wobei jede Wiederholung mitgezählt wurde, da genau diese Anzahl in die Summe Strukturkomplexität eingeht.

B.1 Beispiel 1: "Periodensystem der Elemente" (aus [HE00b])

B.1.1 DTD-Listing

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT PERIODIC_TABLE (ATOM)*>
    <!ELEMENT ATOM (NAME, ATOMIC_WEIGHT, ATOMIC_NUMBER, OXIDATION_STATES?, BOILING_POINT?,
        MELTING_POINT?, SYMBOL, DENSITY?, ELECTRON_CONFIGURATION,COVALENT_RADIUS?,
        ELECTRONEGATIVITY?, ATOMIC_RADIUS?, HEAT_OF_VAPORIZATION?, ATOMIC_VOLUME?,
        HEAT_OF_FUSION?, IONIZATION_POTENTIAL?, SPECIFIC_HEAT_CAPACITY?,
        THERMAL_CONDUCTIVITY?)>
        <!ATTLIST ATOM STATE (GAS | SOLID) #IMPLIED>
        <!ELEMENT NAME (#PCDATA)>
        <!ELEMENT ATOMIC_WEIGHT (#PCDATA)>
        <!ELEMENT ATOMIC_NUMBER (#PCDATA)>
        <!ELEMENT OXIDATION_STATES (#PCDATA)>
        <!ELEMENT BOILING_POINT (#PCDATA)>
            <!ATTLIST BOILING_POINT UNITS CDATA #FIXED "Kelvin">
        <!ELEMENT MELTING_POINT (#PCDATA)>
            <!ATTLIST MELTING_POINT UNITS CDATA #FIXED "Kelvin">
        <!ELEMENT SYMBOL (#PCDATA)>
        <!ELEMENT DENSITY (#PCDATA)>
            <!ATTLIST DENSITY UNITS CDATA #FIXED "grams/cubic centimeter">
        <!ELEMENT ELECTRON_CONFIGURATION (#PCDATA)>
        <!ELEMENT COVALENT_RADIUS (#PCDATA)>
            <!ATTLIST COVALENT_RADIUS UNITS CDATA #FIXED "Angstroms">
        <!ELEMENT ELECTRONEGATIVITY (#PCDATA)>
        <!ELEMENT ATOMIC_RADIUS (#PCDATA)>
            <!ATTLIST ATOMIC_RADIUS UNITS CDATA #FIXED "Angstroms">
        <!ELEMENT HEAT_OF_VAPORIZATION (#PCDATA)>
            <!ATTLIST HEAT_OF_VAPORIZATION UNITS CDATA #FIXED "kilojoules/mole">
        <!ELEMENT ATOMIC_VOLUME (#PCDATA)>
            <!ATTLIST ATOMIC_VOLUME UNITS CDATA #FIXED "cubic centimeters/mole">
        <!ELEMENT HEAT_OF_FUSION (#PCDATA)>
            <!ATTLIST HEAT_OF_FUSION UNITS CDATA #FIXED "kilojoules/mole">
        <!ELEMENT IONIZATION_POTENTIAL (#PCDATA)>
        <!ELEMENT SPECIFIC_HEAT_CAPACITY (#PCDATA)>
            <!ATTLIST SPECIFIC_HEAT_CAPACITY UNITS CDATA #FIXED "Joules/gram/degree Kelvin">
        <!ELEMENT THERMAL_CONDUCTIVITY (#PCDATA)>
            <!ATTLIST THERMAL_CONDUCTIVITY UNITS CDATA #FIXED "Watts/meter/degree Kelvin">
```

B.1.2 DTD-Graph

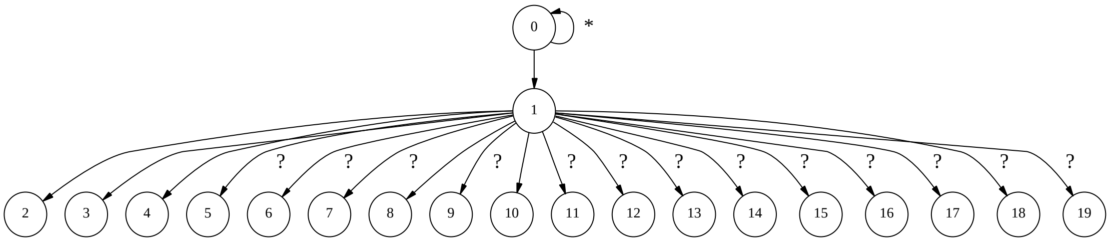


Abbildung B.1: DTD-Graph Beispiel 1

B.1.3 DTD-Bewertung (FAN-IN, FAN-OUT und ST)

Zeilen-ID	3	2	9	10	11	12	13	15	17	18	20	21	23	24	26	28	30	32	33	35
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
FAN-IN	1	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FAN-OUT	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ST	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

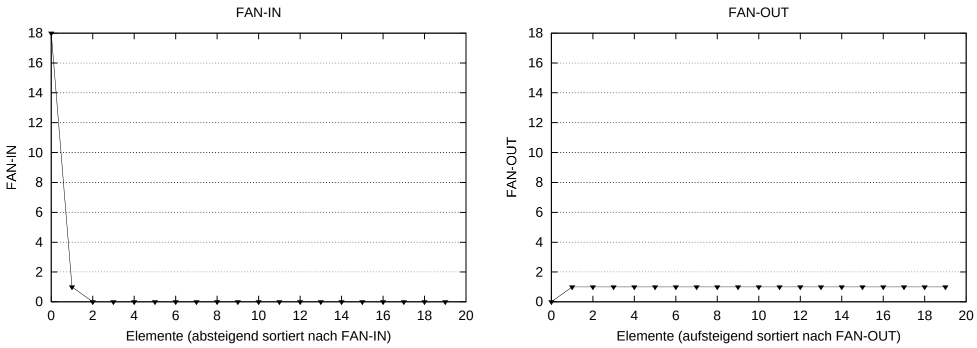


Abbildung B.2: FAN-IN und FAN-OUT Beispiel 1

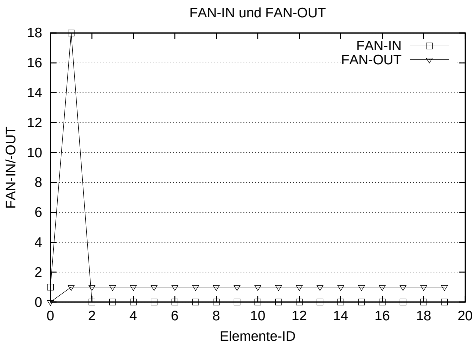


Abbildung B.3: FAN-IN versus FAN-OUT Beispiel 1

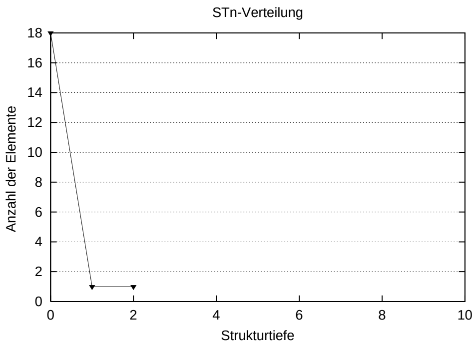


Abbildung B.4: STn-Verteilung Beispiel 1

B.2 Beispiel 2: ”DTD for Testaments” (aus [HE00b])

B.2.1 DTD-Listing

```
<!-- DTD for testaments    J. Bosak -->
<!-- Early versions 1992-1998 -->
<!-- Major revision Copyright (c) Jon Bosak September 1998 -->

<!ENTITY % plaintext "#PCDATA|i">

<!ELEMENT tstmt (coverpg?, titlepg?, preface?, (bookcoll | suracoll)+)>
    <!ELEMENT coverpg ((title | title2)+, (subtitle | p)*)>
    <!ELEMENT titlepg ((title | title2)+, (subtitle | p)*)>
    <!ELEMENT title (%plaintext;)*>
    <!ELEMENT title2 (%plaintext;)*>
    <!ELEMENT subtitle (p)+>
    <!ELEMENT preface ((ptitle | ptitle0)+, p+, witlist?)+>
    <!ELEMENT witlist (witness)+>
    <!ELEMENT ptitle (%plaintext;)*>
    <!ELEMENT ptitle0 (%plaintext;)*>
    <!ELEMENT witness (%plaintext;)*>
    <!ELEMENT bookcoll (book | sura)+>
    <!ELEMENT book (bktlong, bktshort, epigraph?, bksum?, chapter+)>
    <!ELEMENT suracoll (sura+)>
    <!ELEMENT sura (bktlong, bktshort, epigraph?, bksum?, v+)>
    <!ELEMENT bktlong (%plaintext;)*>
    <!ELEMENT bktshort (%plaintext;)*>
    <!ELEMENT bksum (p)+>
    <!ELEMENT chapter (chtitle, chstitle?, epigraph?, chsum?, (div+ | v+))>
    <!ELEMENT chtitle (%plaintext;)*>
    <!ELEMENT chstitle (%plaintext;)*>
    <!ELEMENT div (divtitle, v+)>
    <!ELEMENT divtitle (%plaintext;)*>
    <!ELEMENT chsum (p)+>
    <!ELEMENT epigraph (%plaintext;)*>
    <!ELEMENT p (%plaintext;)*>
    <!ELEMENT v (%plaintext;)*>
    <!ELEMENT i (%plaintext;)*>
```

B.2.2 DTD-Graph

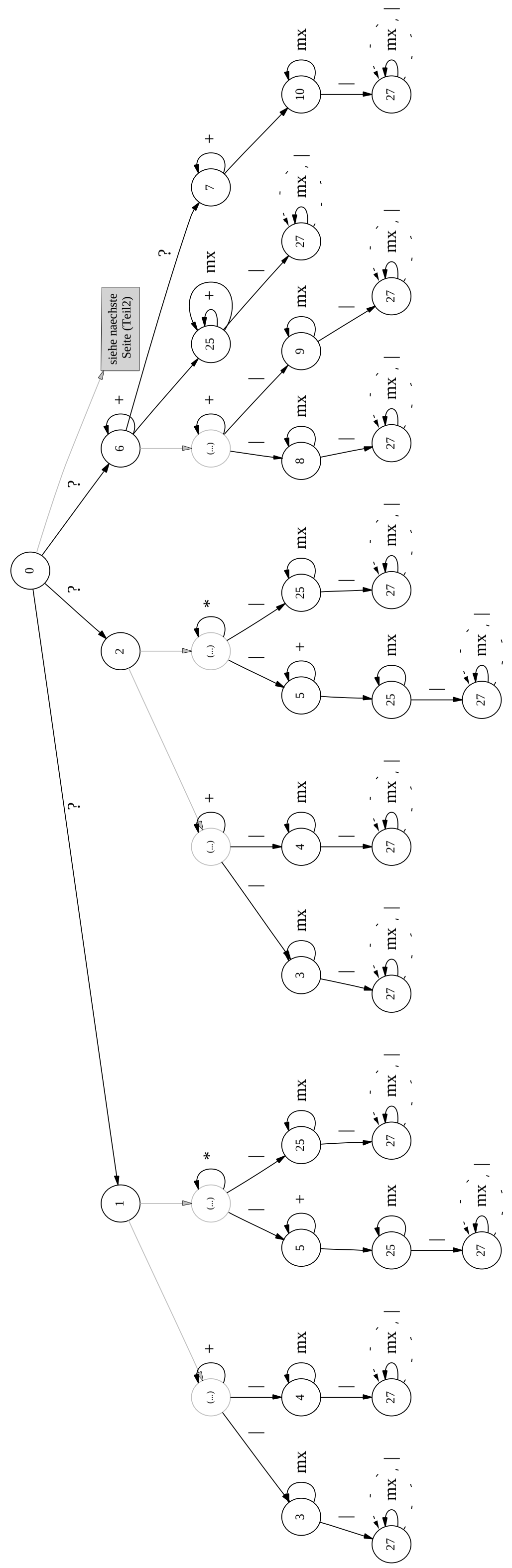


Abbildung B.5: DTD-Graph Beispiel 2 (Teil 1)

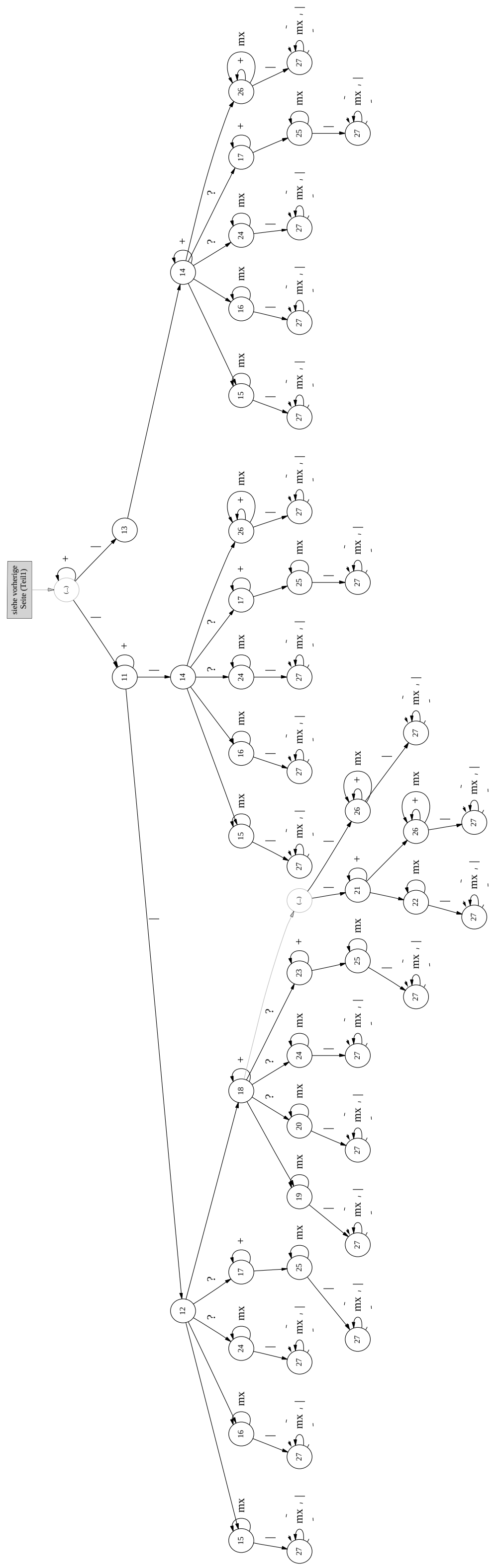


Abbildung B.6: DTD-Graph Beispiel 2 (Teil 2)

B.2.3 DTD-Bewertung (FAN-IN, FAN-OUT und ST)

Zeilen-ID	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
FAN-IN	5	4	4	1	1	1	4	1	1	1	1	2	5	1	5	1	1	1	6	1	1	2	1	1	1	1	1	1
FAN-OUT	0	1	1	2	2	2	1	1	1	1	1	1	1	2	1	2	2	2	1	1	1	1	1	1	3	6	3	14
ST	6	3	3	1	1	2	3	2	1	1	1	5	4	4	3	1	1	2	3	1	1	2	1	2	1	1	1	0

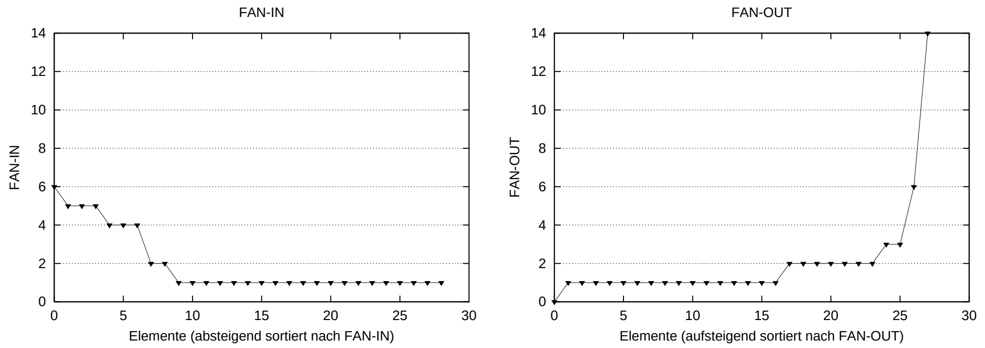


Abbildung B.7: FAN-IN und FAN-OUT Beispiel 2

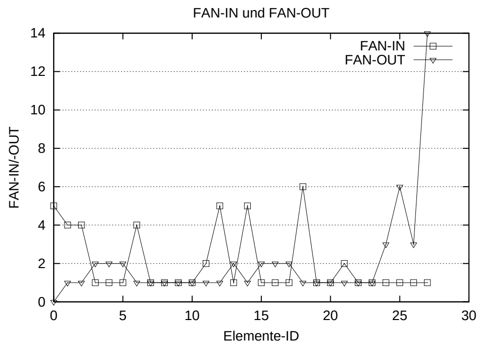


Abbildung B.8: FAN-IN versus FAN-OUT Beispiel 2

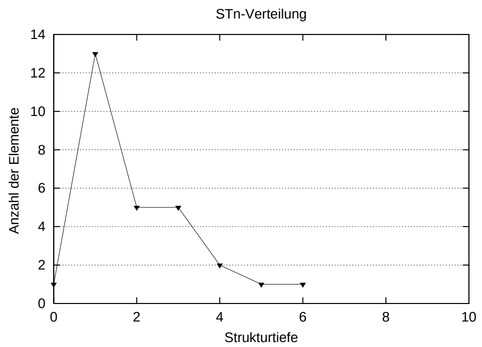


Abbildung B.9: STn-Verteilung Beispiel 2

B.3 Beispiel 3: "DTD for Shakespeare" (aus [HE00b])

B.3.1 DTD-Listing

```
<!-- DTD for Shakespeare    J. Bosak    1994.03.01, 1997.01.02 -->
<!-- Revised for case sensitivity 1997.09.10 -->
<!-- Revised for XML 1.0 conformity 1998.01.27 (thanks to Eve Maler) -->

<!ENTITY amp "&#38;#38;">

<!ELEMENT PLAY (TITLE, FM, PERSONAE, SCNDESCR, PLAYSUBT, INDUCT?, PROLOGUE?, ACT+, EPILOGUE?)>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT FM (P+)>
    <!ELEMENT P (#PCDATA)>
    <!ELEMENT PERSONAE (TITLE, (PERSONA | PGROUP)+)>
    <!ELEMENT PGROUP (PERSONA+, GRPDESCR)>
    <!ELEMENT PERSONA (#PCDATA)>
    <!ELEMENT GRPDESCR (#PCDATA)>
    <!ELEMENT SCNDESCR (#PCDATA)>
    <!ELEMENT PLAYSUBT (#PCDATA)>
    <!ELEMENT INDUCT (TITLE, SUBTITLE*, (SCENE+ | (SPEECH | STAGEDIR | SUBHEAD)+))>
    <!ELEMENT ACT (TITLE, SUBTITLE*, PROLOGUE?, SCENE+, EPILOGUE?)>
    <!ELEMENT SCENE (TITLE, SUBTITLE*, (SPEECH | STAGEDIR | SUBHEAD)+)>
    <!ELEMENT PROLOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
    <!ELEMENT EPILOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
    <!ELEMENT SPEECH (SPEAKER+, (LINE | STAGEDIR | SUBHEAD)+)>
    <!ELEMENT SPEAKER (#PCDATA)>
    <!ELEMENT LINE (#PCDATA | STAGEDIR)*>
    <!ELEMENT STAGEDIR (#PCDATA)>
    <!ELEMENT SUBTITLE (#PCDATA)>
    <!ELEMENT SUBHEAD (#PCDATA)>
```

B.3.2 DTD-Graph

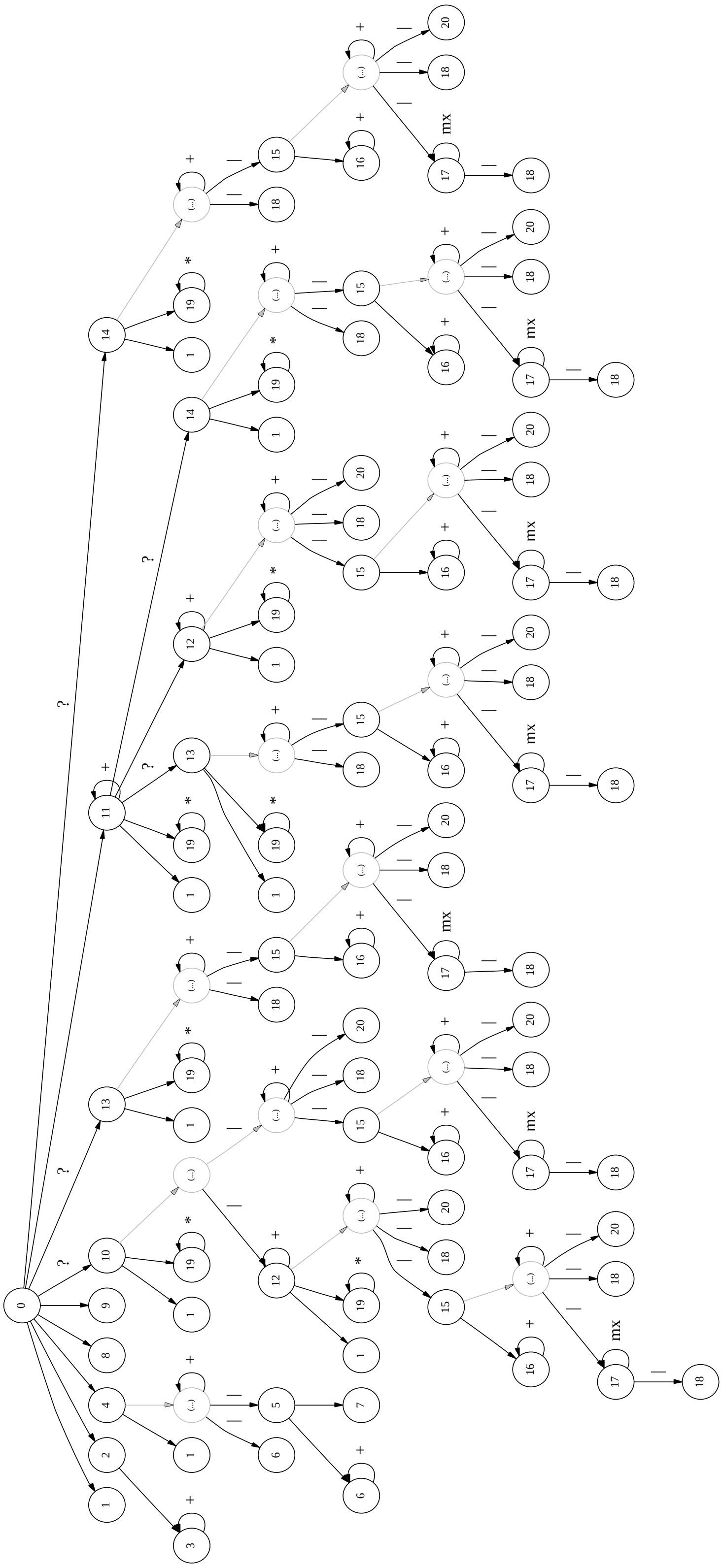


Abbildung B.10: DTD-Graph Beispiel 3

B.3.3 DTD-Bewertung (FAN-IN, FAN-OUT und ST)

Zeilen-ID	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FAN-IN	9	0	1	0	3	2	0	0	0	0	6	5	5	4	4	4	0	1	0	0	0
FAN-OUT	0	7	1	1	1	1	2	1	1	1	1	1	2	2	2	4	1	1	6	5	3
ST	5	0	1	0	2	1	0	0	0	0	4	4	3	3	3	2	0	1	0	0	0

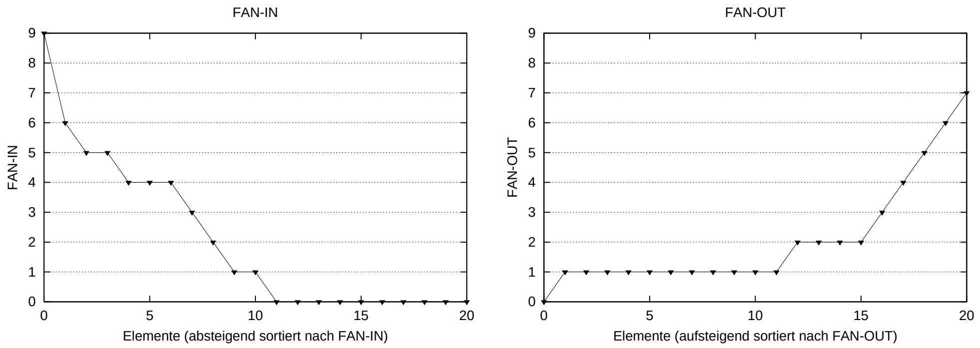


Abbildung B.11: FAN-IN und FAN-OUT Beispiel 3

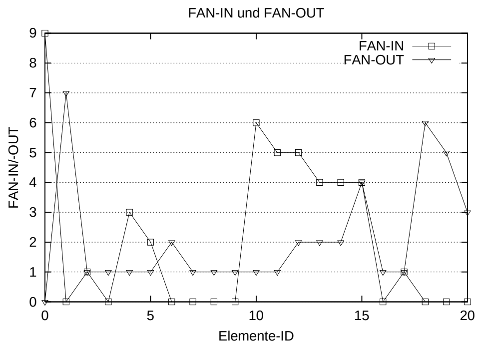


Abbildung B.12: FAN-IN versus FAN-OUT Beispiel 3

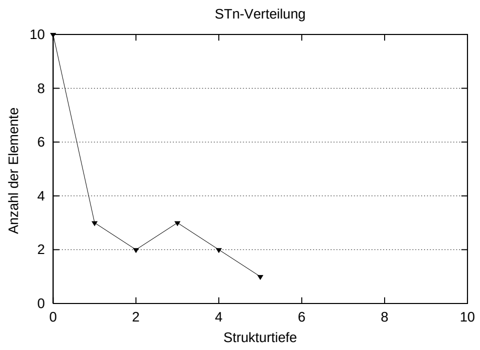


Abbildung B.13: STn-Verteilung Beispiel 3

B.4 Beispiel 4: "Publikationen" (aus [KM00])

B.4.1 DTD-Listing

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT publications (book | article | conference)*>
<!-- book -->
    <!ELEMENT book (front, body, references)>
    <!ELEMENT front (title, author+, edition, publisher)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (first, second, email?)>
    <!ELEMENT first (#PCDATA)>
    <!ELEMENT second (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT edition (#PCDATA)>
    <!ELEMENT publisher (#PCDATA)>
    <!ELEMENT body (part+ | chapter+)>
    <!ELEMENT part (ptitle, chapter+)>
        <!ATTLIST part id ID #REQUIRED>
    <!ELEMENT ptitle (#PCDATA)>
    <!ELEMENT chapter (ctitle, section+)>
        <!ATTLIST chapter id ID #REQUIRED>
    <!ELEMENT ctitle (#PCDATA)>
    <!ELEMENT section (stitle, paragraph+)>
        <!ATTLIST section id ID #REQUIRED>
    <!ELEMENT stitle (#PCDATA)>
    <!ELEMENT paragraph (#PCDATA)>
    <!ELEMENT references (publications*)>
        <!ATTLIST references reftype (book | article | conferences | wwwaddress) "article">
<!-- article -->
    <!ELEMENT article (meta, body, references)>
    <!ELEMENT meta (author+, title, conference)>
<!-- conference -->
    <!ELEMENT conference (editor+, conftitle, city, year)>
    <!ELEMENT editor (first, second, email?)>
    <!ELEMENT conftitle (#PCDATA)>
    <!ELEMENT city (#PCDATA)>
    <!ELEMENT year (#PCDATA)>
```

B.4.2 DTD-Graph

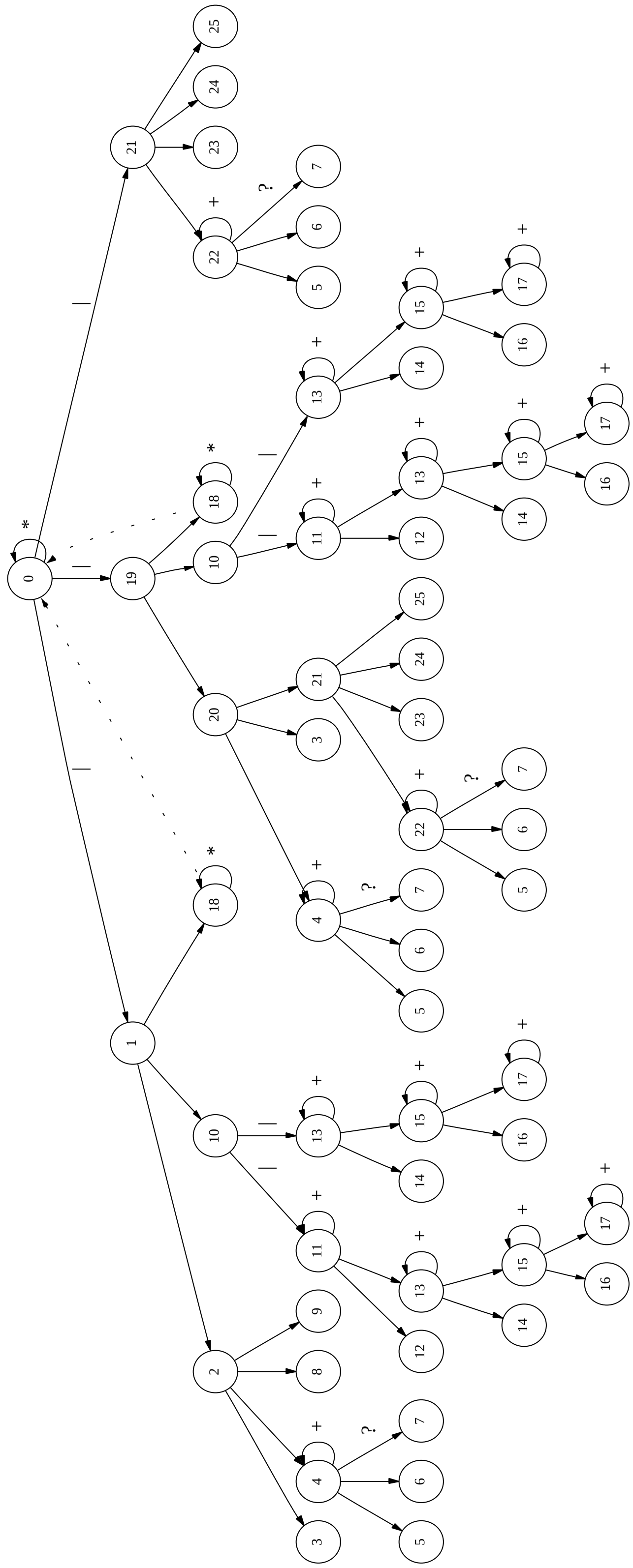


Abbildung B.14: DTD-Graph Beispiel 4

B.4.3 DTD-Bewertung (FAN-IN, FAN-OUT und ST)

Zeilen-ID	2	4	5	6	7	8	9	10	11	12	13	14	16	17	19	20	22	23	24	27	28	30	31	32	33	34
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
FAN-IN	3	3	4	0	3	0	0	0	0	0	2	2	0	2	0	2	0	0	1	3	3	4	3	0	0	0
FAN-OUT	1	1	1	2	2	2	2	2	1	1	2	1	1	2	1	1	1	1	2	1	1	2	1	1	1	1
ST	6	5	2	0	1	0	0	0	0	0	4	3	0	2	0	1	0	0	0	5	3	2	1	0	0	0

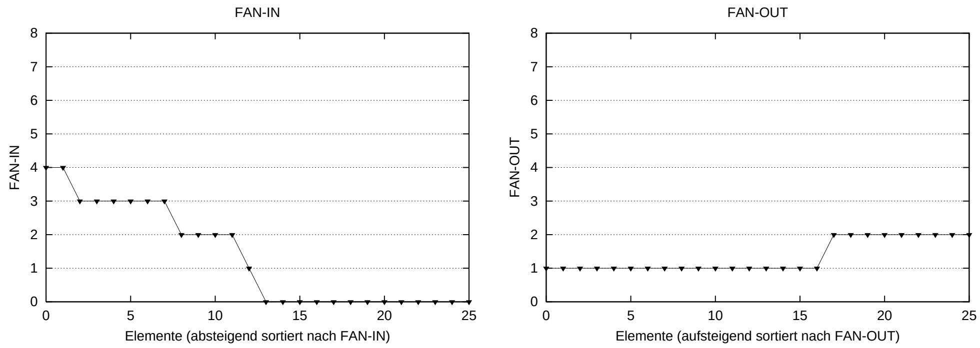


Abbildung B.15: FAN-IN und FAN-OUT Beispiel 4

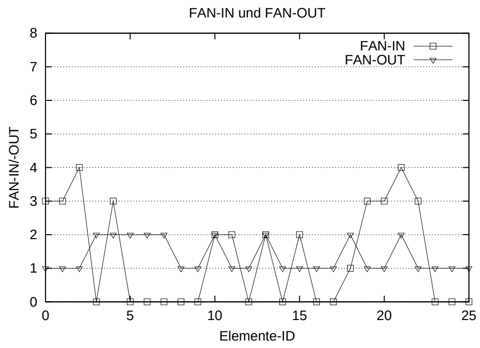


Abbildung B.16: FAN-IN versus FAN-OUT Beispiel 4

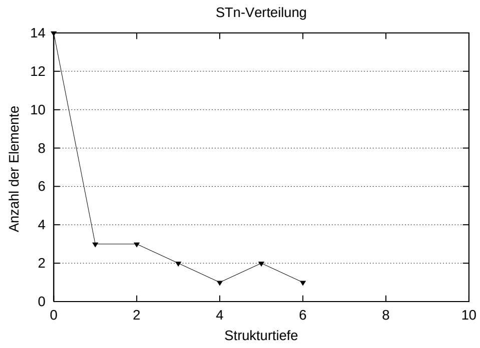


Abbildung B.17: STn-Verteilung Beispiel 4

B.5 Zusammenfassung der Auswertung der DTD-Beispiele

Tabelle B.1: Bewertung der DTD-Beispiele

DTD-Konstrukte / DTD-Metrik	**Beipiel 1**	**Beipiel 2**	**Beipiel 3**	**Beipiel 4**
Grundeinheiten				
Elemente (E)	20	28	21	26
Attribute (A)	11	0	0	4
Entities (EN)	0	1	1	0
Notationen (N)	0	0	0	0
direkte DTD-Metrik				
Umfang der DTD (U_{DTD})	31	29	22	30
indirekte DTD-Metriken				
Strukturkomplexität (SK)	2	123	43	24
Strukturtiefe der DTD (ST_{DTD})	2	6	5	6
Strukturtiefe der Elemente $\overline{ST_E}$[1]	0,15	2	1,38	1,35
$FAN\text{-}IN_{max}$	18	6	9	4
$FAN\text{-}OUT_{max}$	1	14	7	2
$\sum FAN_{DTD}$	19	56	44	35
$\overline{FAN_{DTD}}$[1]	0,95	2	2,1	1,35
Besonderheiten der DTDs				
Wurzelelemente	1	1	1	1
davon mit $FAN\text{-}OUT = 0$	1	1	1	0
Elemente mit $FAN\text{-}IN = 0$	18	0	10	13
Elementrekursion	0	33	0	2
davon direkte	0	33	0	0
davon indirekte	0	0	0	2

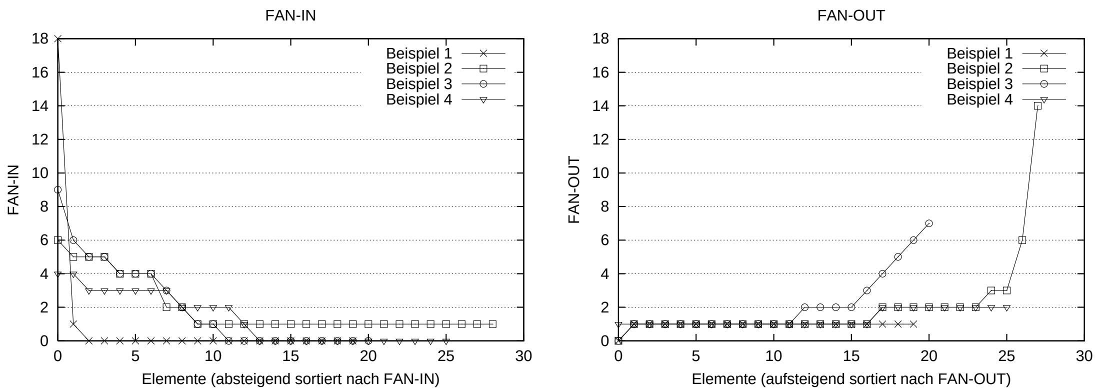


Abbildung B.18: FAN-IN und FAN-OUT der Beispiele im Vergleich

[1]arithmetisches Mittel

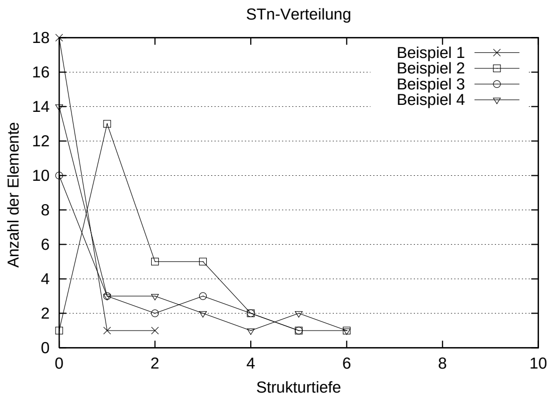


Abbildung B.19: STn-Verteilung der Beispiele im Vergleich

Anhang C

Verzeichnis der Abkürzungen

C

CDATA	**C**haracter **DATA**
CMM	**C**apability **M**aturity **M**odel
CSS	**C**ascading **S**tyle **S**heet

D

DBMS	**D**aten**B**ank-**M**anagement-**S**ystem
DBS	**D**aten**B**ank**S**ystem
DIN	**D**eutsches **I**nstitut für **N**ormung
DSSSL	**D**ocument **S**tyle **S**emantics and **S**pecification **L**anguage
DTD	**D**ocument **T**ype **D**efinition

E

EBNF	**E**rweiterte **B**ackus **N**aur **F**orm
ER	**E**ntity **R**elationship Modell

F

FCM	**F**actor **C**riteria **M**etric – Modell

G

GML	**G**eneralized **M**arkup **L**anguage
GQM	**G**oal **Q**uestion **M**etric – Modell

H

HTML	**H**yper **T**ext **M**arkup **L**anguage

I

IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
ISO	**I**nternational **S**tandardisation **O**rganization

L

LOC	**L**ines **O**f **C**ode

M

MTBF	**M**ean **T**ime **B**etween **F**ailure

O	
ODBS	**O**bjekt**D**aten**B**ank**S**ystem
OMT	**O**bject **M**odeling **T**echnique
OOA	**O**bjekt-**O**rientierte **A**nalyse
OOD	**O**bjekt-**O**rientiertes **D**esign
OODM	**O**bjekt-**O**rientiertes **D**atenbank**M**odell
P	
P3P	**P**latform for **P**rivacy **P**references **P**roject
PCDATA	**P**arsed **C**haracter **DATA**
PI	**P**rocessing **I**nstruction
PICS	**P**latform for **I**nternet **C**ontent **S**election
R	
RDBS	**R**elationales **D**aten**B**ank**S**ystem
RDF	**R**essource **D**escription **F**ramework
RDM	**R**elationales **D**atenbank**M**odell
RM	**R**elationalen-**M**odell
S	
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SK	**S**truktur**K**omplexität
ST	**S**truktur**T**iefe
U	
UML	**U**nified **M**odeling **L**anguage
URI	**U**niform **R**esource **I**dentifier
W	
W3C	**W**orld **W**ide **W**eb **C**onsortium
WWW	**W**orld **W**ide **W**eb
X	
XHTML	e**X**tensible **H**yper **T**ext **M**arkup **L**anguage
XLinks	**X**ML **Link**ing Language
XML	e**X**tensible **M**arkup **L**anguage
XML-DTD	siehe XML und DTD
XPath	**X**ML **Path** Language
XPointer	**X**ML **Pointer** Language
XSD	**X**ML **S**chema **D**escription Language
XSL	e**X**tensible **S**tylesheet **L**anguage
XSLT	**XSL** **T**ransformation

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Arn00] ARNAUD SAHUGUET: Everythink You Ever Wanted to Know About DTDs, But Were Afraid to Ask. In: *Third International Workshop WEBDB2000* Bd. 1997, Springer, May 2000, S. 171–183

[BBM95] BASILI, Victor R. ; BRIAND, Lionel ; MELO, Walcèlio L.: A Validation of Object-oriented Design Metric as Quality Indicators / University of Maryland. 1995. – Forschungsbericht

[BM98] BEHME, Henning ; MINTERT, Stefan: *XML in der Praxis*. Bonn : Addison Wesley Longman Verlag GmbH, 1998

[Boe80] BOEHM, Barry: *Characteristics of Software Quality*. 2. print. Amsterdam : North-Holland Publ. Co., 1980

[Cho00] CHOI, Byron: A Few Tips for Good XML Design / University of Pennsylvania. http://db.cis.upenn.edu/~kkchoi/DTDI2/, November 2000. – Manuskript

[CK94] CHIDAMBER, Shyam R. ; KEMERER, Chris F.: A Metric Suite for Object-oriented Design. In: *IEEE Transaction on Software Engineering* Bd. 20(6), 1994, S. 476–493

[CS00] CHOI, Byron ; SAHUGUET, Arnaud: DTD Mininig Labor Day Report / University of Pennsylvania. http://db.cis.upenn.edu/~kkchoi/DTDI2/, September 2000. – Manuskript

[Dum92] DUMKE, Reiner: *Softwareentwicklung nach Maß: Schätzen, Messen, Bewerten*. Braunschweig : Vieweg, 1992

[Dum01] DUMKE, Reiner: Softwaremetrie - Grundlagen und Ziele / Otto-von-Guericke-Universität Magdeburg. http://ivs.cs.uni-magdeburg.de/~dumke/Metrie/Smetrie.html, 2001. – Vorlesungsskript

[Fen91] FENTON, Norman E.: *Software Metrics – A Rigorous Approach*. London : Chapman & Hall, 1991

[GJP00] GENERO, Marcela ; JIMÉNEZ, Luis ; PIATTINI, Mario: Measuring the Quality of Entity Relationship Diagrams. In: *Conceptual Modeling - ER 2000* Bd. 1920, Springer, 2000

[HE00a] HAROLD, Eliotte R. ; ENGEL, REINHARD (ÜBERS.): *Die XML-Bibel*. 1. Auflage. Bonn : MITP-Verlag, 2000

[HE00b] HAROLD, Eliotte R. ; ENGEL, REINHARD (ÜBERS.): *Die XML-Bibel*. 1. Auflage. Bonn : MITP-Verlag, 2000. – CD-ROM

[HK81] HENRY, Sallie M. ; KAFURA, Dennis G.: Software Structure Metrics based on Information Flow. In: *IEEE Transactions on Software Engineering* Bd. SE7(5), 1981, S. 510–518

[Hog97] Hogan, Jer: An Analysis of OO Software Metrics / University of Warwick. 1997. – Forschungsbericht

[IEE93] Institute of Electrical and Electronics Engineers (IEEE): *Standard for a Software Quality Metrics Methodology.* 1993. – IEEE Std. 1061-1992

[ISO86] International Organisation for Standardization (ISO): *Information processing - Text and office systems - Standard Generalized Markup Language (SGML).* 1986. – ISO 8879:1986

[ISO91a] International Organisation for Standardization (ISO): *Information Technology - Software Evaluation, Quality Characteristics and Guidelines for their Use.* 1991. – ISO 9126

[ISO91b] International Organisation for Standardization (ISO): *Information technology - Text and office systems - Document Style Semantics and Specification Language (DSSSL).* 1991. – ISO/IEC 10197 DIS 1991

[Jec01] Jeckle, Mario: eXtensible Markup Language / Fachhochschule Augsburg. http://www.jeckle.de/vorlesung/xml/script.html, Juli 2001. – Vorlesungsskript

[KM00] Klettke, Meike ; Meyer, Holger: XML and Object-Relational Database Systems - Enhancing Structural Mappings Based on Statistics. In: *Third International Workshop WEBDB2000* Bd. 1997, Springer, May 2000

[McC76] McCabe, Thomas J.: A Complexity Measure. In: *IEEE Transactions on Software Engineering* Bd. SE2(4), 1976, S. 308–320

[Mil56] Miller, George A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. In: *The Psychological Review* Bd. 63, 1956, S. 81–97

[SH97] Saake, Gunter ; Heuer, Andreas: *Datenbanken: Konzepte und Sprachen.* 1. Auflage. Bonn : MITP–Verlag GmbH, 1997

[Tha00] Thaller, Georg E.: *Software-Metriken einsetzen, bewerten, messen.* 2. Auflage. Berlin : Verlag Technik, 2000

[Tur96] Turau, Volker: *Algorithmische Graphentheorie.* Bonn : Addison-Wesley, 1996

[W3C98] Bray, Tim; Paoli, Jean; Sperberg-McQueen, M. C.: Extensible Markup Language (XML) 1.0 / W3C. http://www.w3.org/TR/1998/REC-xml-19980210, 1998. – Recommendation

[W3C99a] Bray, Tim; Hollander, Dave; Layman, Andrew: Namespaces in XML / W3C. http://www.w3.org/TR/1999/REC-xml-names-19990114, 1999. – Recommendation

[W3C99b] Clark, James; DeRose, Steve: XML Path Language (XPath) Version 1.0 / W3C. http://www.w3.org/TR/1999/REC-xpath-19991116, 1999. – Recommendation

[W3C99c] Clark, James: XSL Transformations (XSLT) Version 1.0 / W3C. http://www.w3.org/TR/1999/REC-xslt-19991116, 1999. – Recommendation

[W3C99d] Lassila, Ora; Swick, Ralph R.: Resource Description Framework (RDF): Model and Syntax Specification / W3C. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222, 1999. – Recommendation

[W3C99e] Raggett, Dave; Le Hors, Arnaud; Jacobs, Ian: HTML 4.01 Specification / W3C. http://www.w3.org/TR/1999/REC-html401-19991224, 1999. – Recommendation

[W3C00a] Adler, Sharon [u. a.]: Extensible Stylesheet Language (XSL) Version 1.0 / W3C. http://www.w3.org/TR/2000/CR-xsl-20001121, 2000. – Candidate Recommendation

[W3C00b] Bray, Tim; Maler, Eve; Paoli, Jean; Sperberg-McQueen, M. C.: Extensible Markup Language (XML) 1.0 (Second Edition) / W3C. http://www.w3.org/TR/2000/REC-xml-20001006, 2000. – Recommendation

[W3C00c] Pemberton, Steven [u. a.]: XHTML™ 1.0: The Extensible HyperText Markup Language - A Reformulation of HTML 4 in XML 1.0 / W3C. http://www.w3.org/TR/2000/REC-xhtml1-20000126, 2000. – Recommendation

[W3C01a] Beech, David; Maloney, Murray; Mendelsohn, Noah; Thompson, Henry S.: XML Schema Part 1: Structures / W3C. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502, 2001. – Recommendation

[W3C01b] Biron, Paul V.; Malhotra, Ashok: XML Schema Part 2: Datatypes / W3C. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502, 2001. – Recommendation

[W3C01c] Cowan, John; Tobin, Richard: XML Information Set / W3C. http://www.w3.org/TR/2001/REC-xml-infoset-20011024, 2001. – Recommendation

[W3C01d] Daniel Jr., Ron; DeRose, Steve; Maler, Eve: XML Pointer Language (XPointer) Version 1.0 / W3C. http://www.w3.org/TR/2001/WD-xptr-20010108, 2001. – Working Draft

[W3C01e] DeRose, Steve; Maler, Eve; Orchard, David: XML Linking Language (XLink) Version 1.0 / W3C. http://www.w3.org/TR/2001/REC-xlink-20010627, 2001. – Recommendation

[W3C01f] Fallside, David C.: XML Schema Part 0: Primer / W3C. http://www.w3.org/TR/2001/REC-xmlschema-0-20010502, 2001. – Recommendation

[ZD01] Zuse, Horst ; Drabe, Karin: *ZD-MIS: Zuse/Drabe - Measurement Information System*. URL http://home.t-online.de/home/horst.zuse/zdmis.html, February 2001

[Zei01] Zeitz, Andre: *Evolution von XML-Dokumenten*, Universität Rostock, Fachbereich Informatik, Studienarbeit, 2001

[Zus91] Zuse, Horst: *Software Complexity : measures and methods*. Berlin : de Gruyter, 1991

[Zus98] Zuse, Horst: *A Framework of Software Measurement*. Berlin : de Gruyter, 1998