TU Dresden

Fakultät Informatik

Institut für Softwaretechnologie II

Generische Metadatenmodelle

http://www-st.inf.tu-dresden.de/UMLToolkit

Hochschullehrer	**Prof. Dr. rer. nat. habil. H. Hußmann**
Betreuer	**Frau Dr. B. Demuth**
Referent	**Sven Obermaier**

Agenda

1 Thema (Großer Beleg)

2 XML (Geschichte, Bedeutung heute)

3 UML und XMI (Einordnung)

4 DTD2J (Ergebnis meiner Arbeit)

5 Generierte Metadatenmodelle (Leistung)

6 Ausblick (Zukunft von DTD2J)

7 Diskussion

1 Thema

Aufgabenstellung

1 Thema

Untersuchen Sie die Möglichkeit, die Dokumententypdefinitionen (DTD) der XML- Spezifikation so zu nutzen, um daraus eine interne Java-Repräsentation zu generieren.

Erproben Sie Ihre Untersuchungen anhand der OMG UML DTD.

2 XML

(eXtensible Markup Language)

2.1 Geschichtlicher Hintergrund

2.2 Bedeutung von XML heute

2.3 Aufbau von XML

2.1 Geschichtlicher Hintergrund

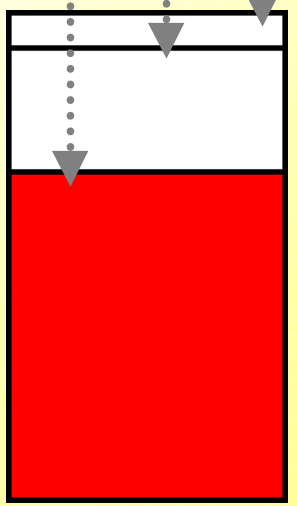
- 1945 Vannevar Bush: MEMEX
- 1965 Ted Nelson: XANADU (xanadu.net)
- 1969 Charles Goldfarb: GML
- 1980 IBM, Charles Goldfarb: SGML
- 1986 Oktober, ANSI: SGML = ISO 8879
- 1989 Idee des WWW
- 1991 HTTP und HTML
- 1997 Dezember, W3C: XML 1.0 als Standard

2.2 Bedeutung von XML heute

- Standardisiertes Datenaustauschformat
- Plattformunabhänig
- Reine (nichtlineare) Wissensspeicherung
- Wiederverwendbarkeit von Informationen
- Unterstützt die Suche über Daten (z.B.XQL)
- Wirtschaftliches Interesse steigt
- Wird HTML als Websprache ablösen
- XML basiert auf DOM (W3C-Standard)

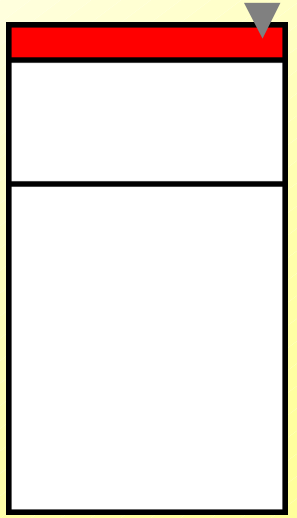
2.3 Aufbau von XML

- Header
- Prolog (DTD)
- Dokumenteninstanz



2.3.1 Header

- Identifikation einer XML-Datei
- `<?xml version=„1.0“?>`
- Fest vorgegeben
- Erweiterung möglich u.a. um Zeichensatz, z.B. UTF-8, ISO-10646-UCS-2 und EUC-JP

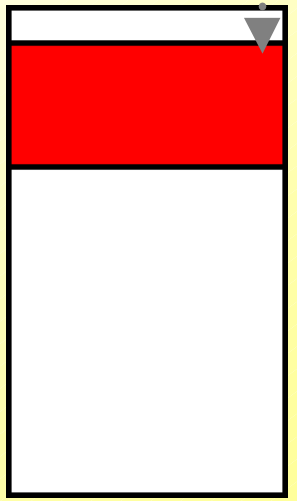


2.3.2 Prolog (DTD)

- Dokumententypdefinition
- Beschreibt die Formatierung der folgenden Dokumenteninstanz durch Elemente

```
<!ELEMENT email (#PCDATA)>
<!ATTLIST email
                address IDREF #REQUIRED
                subject CDATA #IMPLIED
                cc IDREFS #IMPLIED>
```

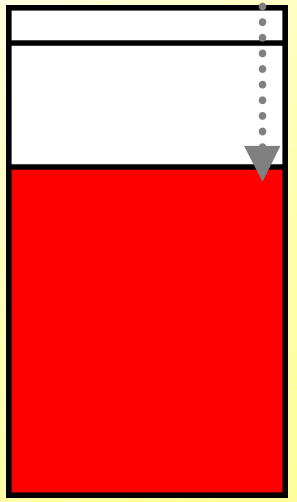
- DTD‘s sind wiederverwendbar



2.3.3 Dokumenteninstanz

- Speicherung von Informationen in der durch die DTD vorgegebenen Struktur

```
<email address=„sven“ subject=„Kuchen“>
Hallo Sven, der Kuchen ist alle,
geh‘ mal welchen kaufen!
</email>
```



3 UML und XMI

Begriffsklärung

Einordnung

3 UML und XMI

- Unified Modeling Language - bekannt
- XML Metadata Interchange
  - Metadaten-Austausch zwischen Modellierungs-tools basierend auf XML
  - Integriert XML, UML und MOF (Meta Object Facility - metamodeling and metadata repository standard)

4 DTD2J

4.1 Zweck von DTD2J

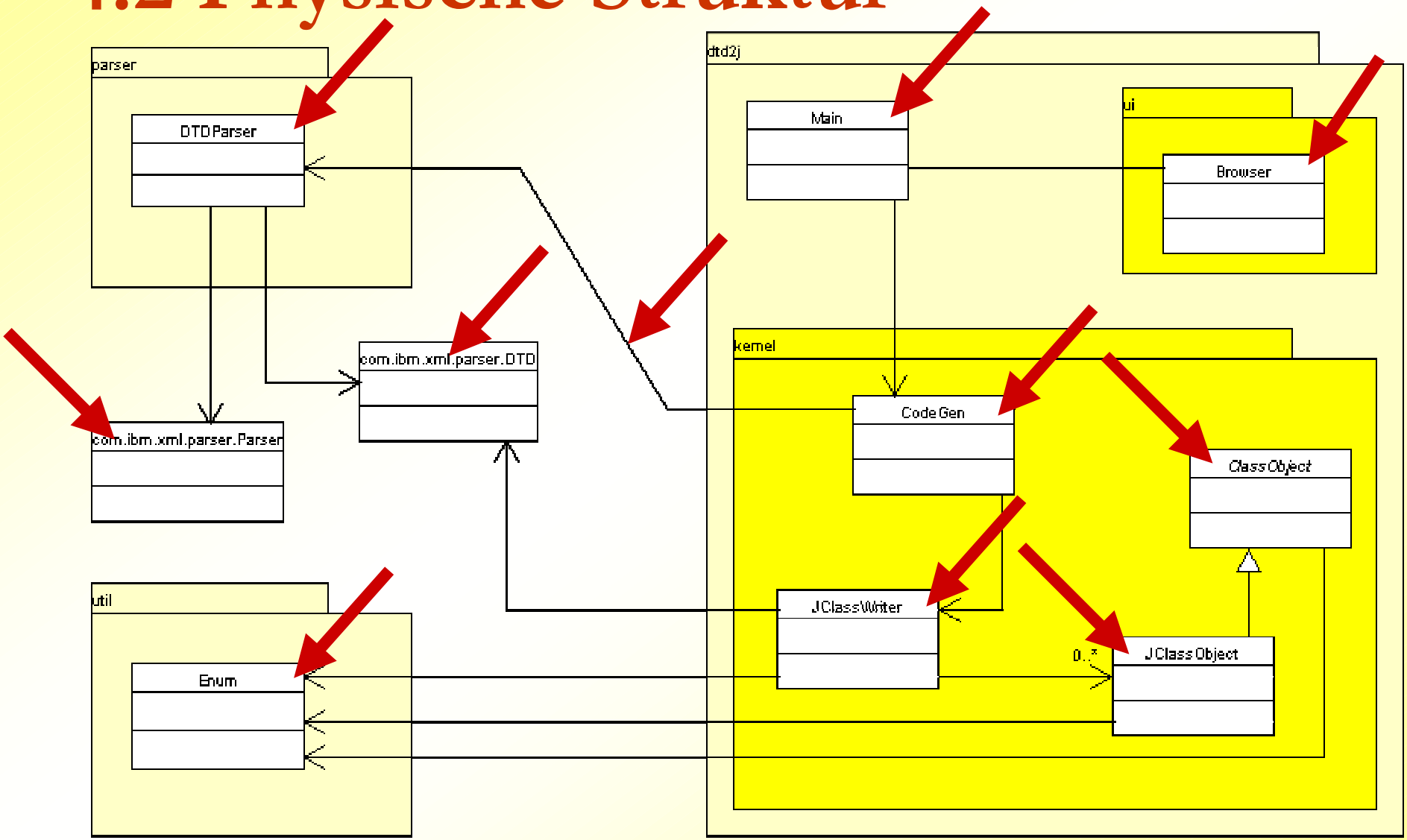
4.2 Physische Struktur

4.3 Funktionsweise

4.1 Zweck von DTD2J

- Generierung eines Metadatenmodells basierend auf der UML DTD der OMG
- Import und Export von XMI Dateien
- Verallgemeinerung für alle XML-DTD‘s
- Keine „Handarbeit“ im generierten Code
- DTD-Update ohne große Applikations-anpassungen
- Bsp.: UML.DTD Versionssprung 1.1 → 1.3
- Codeintegration in Argo/UML (kalifornisches Open Source Projekt)

4.2 Physische Struktur

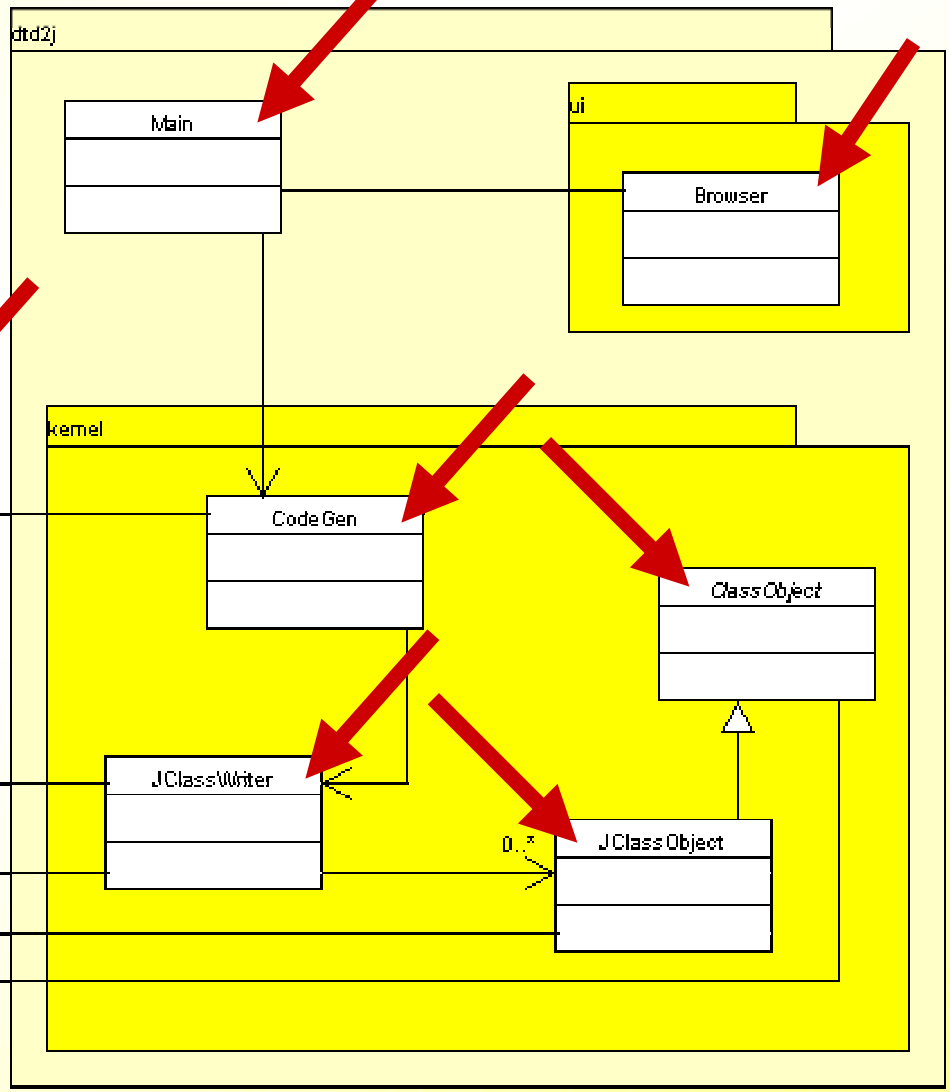


4.2 Physische Struktur

Main

public static boolean showProgress
public static boolean verboseMode
public static boolean comments
public static boolean reduce
public static String packageName
public static String sourceFilename
public static String destinationPath

public static void main (String[] argv)
public static boolean nowGenerate()
private static void showHelpMessages()

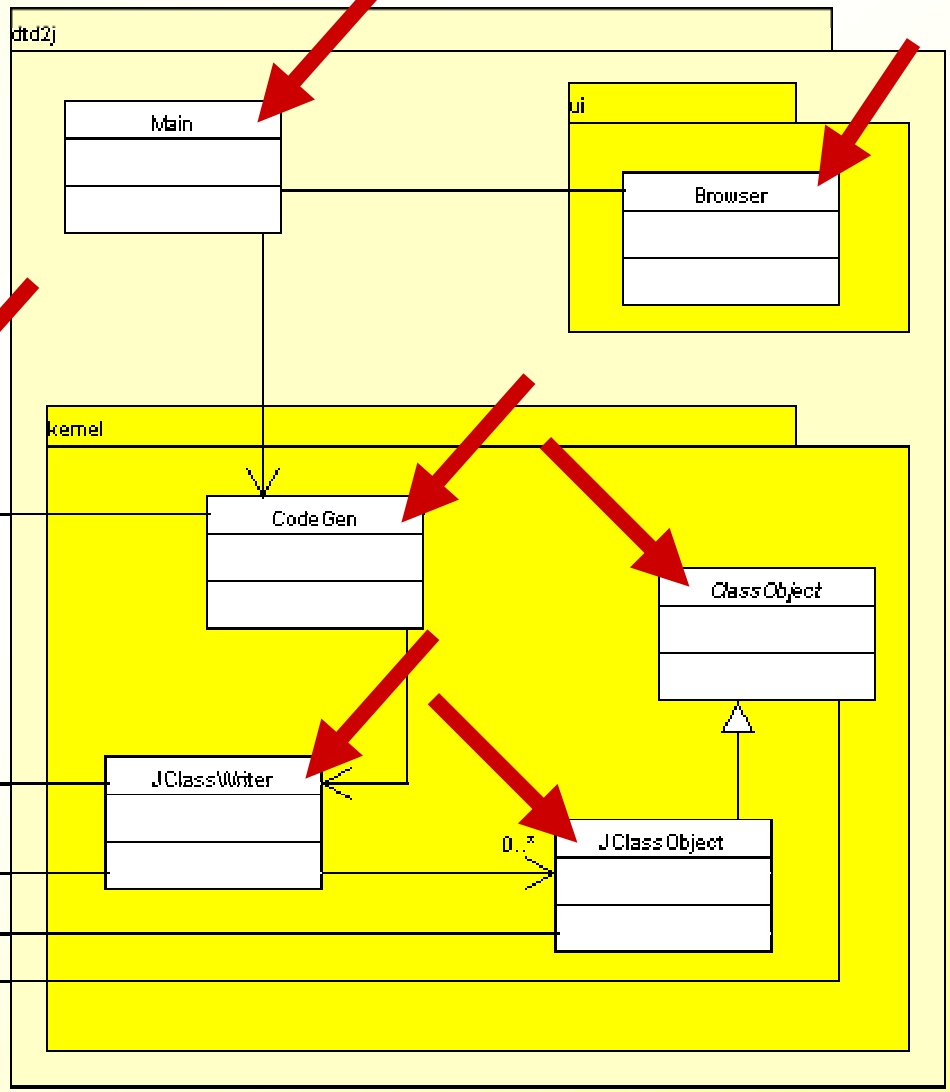


4.2 Physische Struktur

CodeGen

public boolean showProgress
public boolean verboseMode
public boolean comments
public boolean reduce

Public boolean work
(String DTDfilename,
String destinationPath,
String packageName)

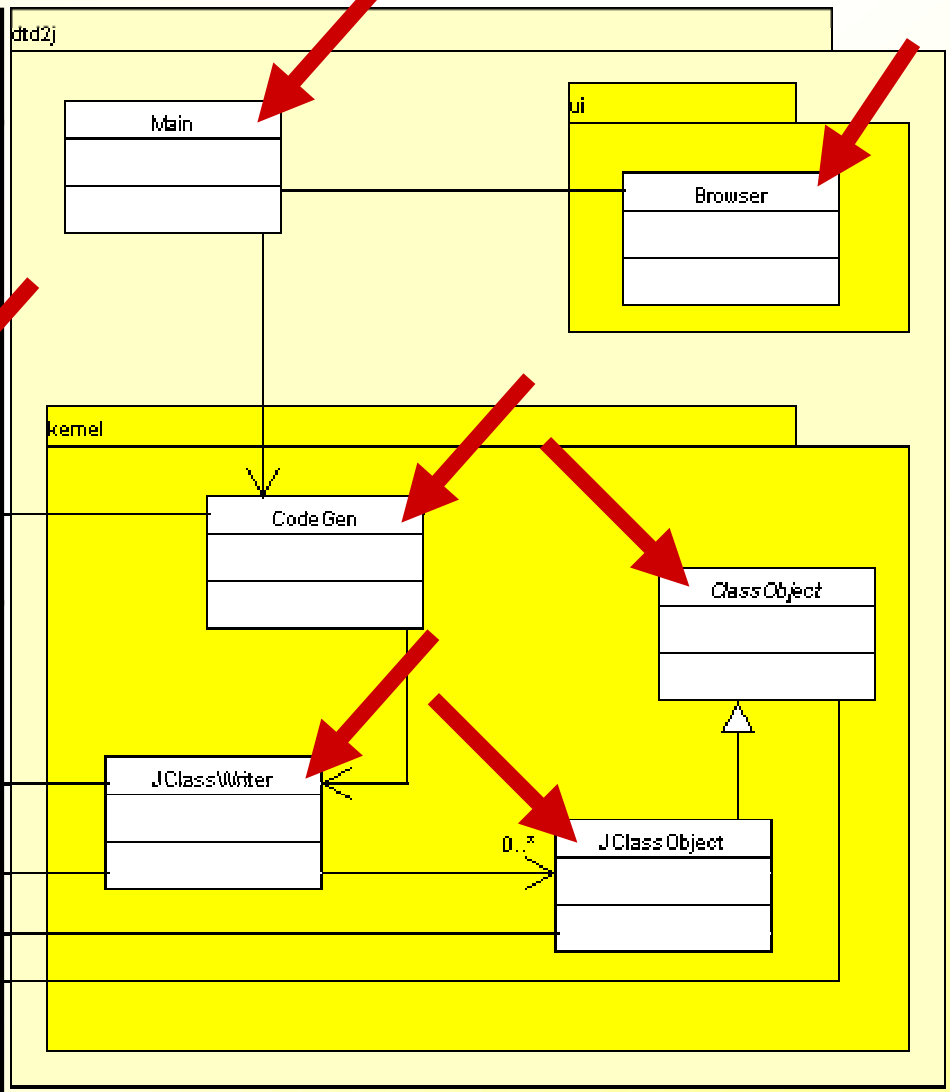


4.2 Physische Struktur

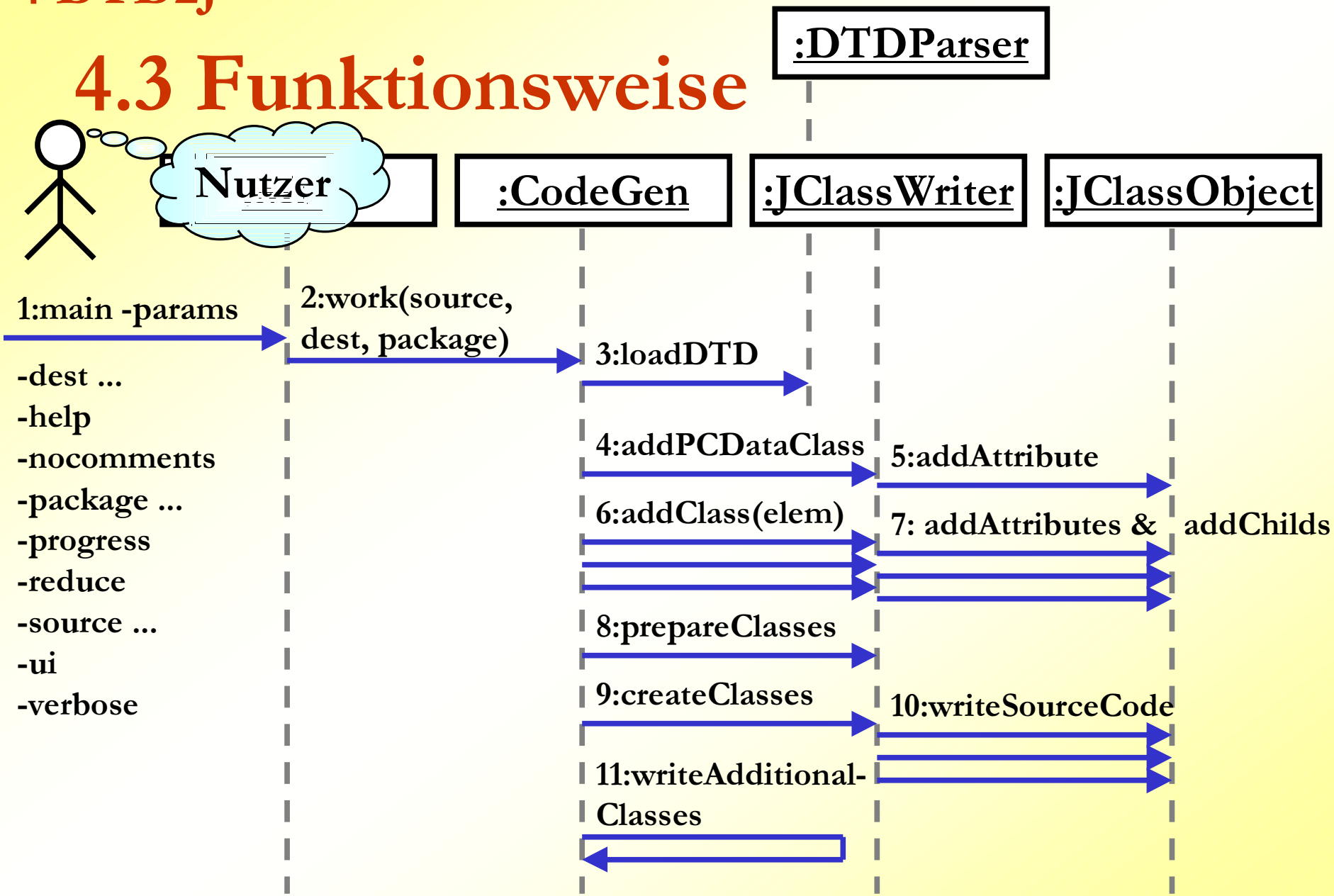
(J)ClassWriter

private DTD dtd
private static String destPath
private static String packName
private Hashtable allClasses

public boolean addPCDATAClass()
public boolean addClass
(ElementDecl element)
public int prepareClasses()
public int createClasses()



4.3 Funktionsweise



5 Generierte Metadatenmodelle

5.1 Beispiel einer Klasse

5.2 Heutige Leistungsfähigkeit

5.3 Schwierigkeiten

5.1 Beispiel einer Klasse (1-3)

```
package tudresden.test;
import tudresden.test.util.ElemObject;
import tudresden.test.util.DTDElement;
import tudresden.test.Personnel_car_type;
import tudresden.test.PCDATAContent;
// Klasse personal.car
public class Personnel_car extends
    tudresden.test.util.DTDElement {
    ... // nächste Folie
}
```

5.1 Beispiel einer Klasse (2-3)

```
// Konstruktor
public Personnel_car(ElemObject
    containerElement) {
        super(containerElement);
        allChilds = new Hashtable();
        privA_default_color = new
            String("red");
}
public final String originalElementDTD =
    "(#PCDATA|personnel.car.type)*";
// weiter auf nächsten Folie
```

5.1 Beispiel einer Klasse (3-3)

```
// Propertyzugriffe auf Attribute
public boolean default_color(String
    newA_default_color) { ... }
public String default_color() { ... }
// usw.
// Kind(untergeordneter Knoten) hinzufügen
public boolean putElement(ElemObject
    elemValue, boolean makeToChild)
{ ... }
// u.a.
```

5.2 Heutige Leistungsfähigkeit

- Vollständige Implementation der Klassen
- Navigation im Baum (ähnlich DOM)
- Direktzugriff auf Elemente über UID
- Direktzugriff auf Attribute möglich
- Umsetzung von ANY, EMPTY, #REQUIRED, #FIXED, #IMPLIED
- Vorbereitung für weitere Datentypen (int, ...)
- XML Import

5.3 Schwierigkeiten

- Code-Generierung dauert zu lang
- Zielcode zu groß! UML1.1 komplett >8MBytes, >115‘000 Zeilen Quellcode
- Keine DTD-bezogene Package-Struktur
- Umsetzung der Elementbeziehungen, Bsp.:
  `<!ELEMENT a (#PCDATA|(b*,(c?|d)))+>`
- Fehlende Individualität (applikationsspezif. Ergänzungen DTD-unabhängig)
- Fehlende Flexibilität (autom. DTD-Update)

6 Ausblick

Zukunft von DTD2J

6 Ausblick (1-2)

- User Interface (Dummy vorhanden)
- Erweiterung für andere Sprachen
  - C++ (CObjectClass)
  - Visual Basic (VBObjectClass)
- Schnittstellen für Import und Export
  - XML (z.T. schon realisiert)
  - Relationales SQL (Create Table, Select, ...)
  - HTML (XSL) (nur Generierung)
- Undo/Redo-Funktionalität

6 Ausblick (2-2)

- Zukünftig Javaprojekte des Instituts Softwaretechnologie II durch DTD2J schneller realisieren
- Einbindung des UML-Metadatenmodells in Argo/UML
- Kopplung mit Frank Fingers „OCL Class Library“
- Und mehr...

7 Diskussion

http://www-st.inf.tu-dresden.de/UMLToolkit