Handbuch
Version 2.0.1

# nsd2ltx
# Struktogramme mit XML

Jürgen A. Lamers
jaloma@dokutransdata.de

13. November 2002

DokuTransData

**Beschreibung der Anwendung nsd2ltx**

*Für Karen*

**Zusammenfassung**

Beschreibung der XML-Dokumentde£nitionen `struktogramm.dtd` und `struktex.dtd` in Verbindung mit der Anwendung `nsd2ltx`.

`nsd2ltx` konvertiert XML-Struktogramme in LaTeX-Struktogramme. Die XML-Dateien müssen der Dokument Typ De£nition `struktex.dtd` genügen. Die erzeugten LaTeX-Dateien benutzen den Befehlssatz aus dem Paket `struktex`.

# Inhaltsverzeichnis


**1 Die Idee** **1**
- 1.1 Einleitung . . . 1
- 1.2 Generelle Voraussetzungen . . . 2
- 1.3 Ausgabemodi . . . 3

**2 PHP-Version** **5**
- 2.1 Voraussetzungen . . . 5
- 2.2 Installation . . . 5
- 2.3 Benutzung . . . 6

**3 C++ Version** **7**
- 3.1 Das Kommandozeilen-Programm . . . 7
  - 3.1.1 Voraussetzungen . . . 7
  - 3.1.2 Installation . . . 7
  - 3.1.3 Benutzung . . . 7
- 3.2 Der Wizard . . . 8
  - 3.2.1 Voraussetzungen . . . 8
  - 3.2.2 Installation . . . 8
  - 3.2.3 Benutzung . . . 8

**4 Ergänzende Hilfsmittel** **15**
- 4.1 Emacs-Mode nsd-helper . . . 15
- 4.2 Style strukto . . . 15
- 4.3 epstopng . . . 15

**A Die Dokumenttypdefinitionen** **17**
- A.1 Erläuterung der Struktogrammdefinition . . . 17
- A.2 Erläuterung der Kon£gurationsdefinition . . . 25

**B Ein bisschen Theorie** **29**

**C Informationen** **33**

**Abbildungsverzeichnis** **37**

**Struktogrammverzeichnis** **39**

**Stichwortverzeichnis** **41**

# 1 Die Idee

## 1.1 Einleitung

Ich habe für Linux bis jetzt noch keinen Wysiwyg-Editor für Struktogramme[1] gefunden (DIA kann zwar Flow-Charts und UML-Diagramme erzeugen, aber eben keine Nassi-Shneiderman Diagramme.). Zum Anderen gibt es die grundsätzliche Vorgabe, dass die Ausgabe mit LaTeX und dem Style StrukTeX erzeugt wird, d. h. ich muss die graphisch erzeugten Struktogramme weiter verarbeiten können[2]. Deswegen musste ich mir etwas einfallen lassen, um eine ansprechende, einfache und sichere Benutzerführung bei der Erzeugung von Struktogrammen zu entwickeln.

Nach kurzem Besinnen bin ich auf die Idee gekommen, dass das Basisformat XML sein sollte und meine *graphische* Eingabe mit dem Emacs durchgeführt wird. Ich habe mich für den Emacs entschieden, weil der XML-Mode `psgml` eine einfache und **sichere** Schnittstelle zur Eingabe von XML-Dokumenten darstellt[3].

Zwar braucht man/frau im Grunde bei der Entwicklung von Struktogrammen keine Layoutkontrolle, doch wie ich die meisten Anwender kenne, möchten sie sehen, wie schön ihre Struktogramme aussehen. Zudem musste ich sowieso eine Konvertierung der XML-Dateien nach LaTeX programmieren, darum habe ich einen kleinen Emacs-Mode zur Anbindung der PHP-Anwendung `php4nsd2ltx` bzw. C++Anwendung `nsd2ltx` geschrieben, der On-The-Fly DVI- oder PDF-Dokumente erzeugt und die farbliche Gestaltung der XML-Dateien übernimmt.

#### Vorteile

- ☞ Emacs, (PHP) und LaTeX ist bald überall verfügbar
- ☞ Einfache und validierte Eingabe der Struktogramme

#### Nachteile

- ☞ Auch wenn On-the-Fly DVI-Dateien erzeugt werden können, den richtigen Überblick bei komplexen Struktogrammen bekommt man nicht (BTW: Wer komplexe Struktogramme hat, macht IMHO sowieso etwas falsch. . . )
- ☞ Komplexe Installation mit den Komponenten Emacs, (PHP,) (PDF)LaTeX und StrukTeX

[1] Wer keine Ahnung von Struktogrammen hat, soll sich sofort zu Abschnitt B begeben ;-)

[2] Bei G.E.S.y II (© Michael Denzlein) ist dies möglich, da die Struktogramme als ASCII-Dateien gespeichert werden und einigermassen logisch aufgebaut sind.

[3] Ich habe verschiedene GUI-Editoren für XML ausprobiert, die meisten validieren nicht zur DTD und wenn, dann kosten diese Geld. . .

## 1.2 Generelle Voraussetzungen

☞ LaTeX-Installation mit den Styles StruckTeX und `Float`. Je nach Ausgabeart müssen noch

- pdfLaTeX
- psLaTeX
- dvips
- ghostscript
- dvipdfm
- epstopng[4]
- epstopdf

installiert sein.

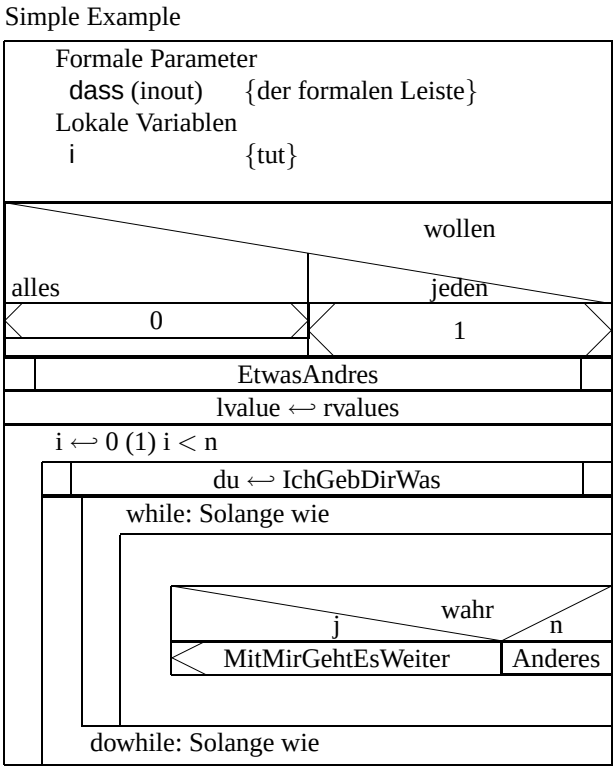
☞ Es sollte der Emacs-Mode `psgml` installiert sein

## Beispiele

In dem Verzeichnis `docs/examples` finden sich zwei Beispiele, deren Ausgabe findet sich unter `docs/examples/output`. Eine Beschreibung der XML-Syntax finden Sie in Abschnitt A.1.

```
<struktogramm name="Simple Example" height="119">
  <paramdecl>
    <pvardecl name="dass" reftype="inout"
              description="der formalen Leiste"/>
  </paramdecl>
  <localdecl>
    <lvardecl name="i" description="tut"/>
  </localdecl>
  <case condition="wollen">
    <switch choice="alles">
      <return value="0" />
    </switch>
    <switch choice="jeden" position="c">
      <return value="1" />
    </switch>
  </case>
  <call subroutine="EtwasAndres"/>
  <assignment lvalue="lvalue" rvalue="rvalues"/>
  <for countervar="i" start="0" end="i &lt; n" increment="1">
  <call subroutine="IchGebDirWas" lvalue="du"/>
    <dowhile condition="dowhile: Solange wie">
      <while condition="while: Solange wie">
        <loop>
          <if condition="wahr">
            <then>
              <exit value="MitMirGehtEsWeiter"/>
            </then>
            <else>
              <statement>Anderes</statement>
            </else>
          </if>
        </loop>
      </while>
    </dowhile>
  </for>
</struktogramm>
```

Abbildung 1.1: Einfaches Beispiel




Struktogramm 1: Simple Example

[4] In dieser Distribution enthalten

## 1.3 Ausgabemodi

Es gibt die nachfolgenden Möglichkeiten zur Ausgabe der Struktogramme, die mit (*) gekennzeichneten Modi sind in der PHP-Anwendung nicht implementiert.

**tex** Es werden nur die Dateien der Struktogramme erzeugt.

**dvi** Es wird zum Modul eine Hauptdatei angelegt, die einzelnen Struktogramme werden als separate TEX-Dateien angelegt. Anschliessend wird LATEX und der DVI-Viewer `xdvi` aufgerufen.

**pdf** Ähnlicher Ablauf wie bei **dvi**, nur das eine PDF-Datei erzeugt und `acroread` aufgerufen wird.

**Voraussetzung** Installiertes pdfLATEX.

**dvi2pdf** Es wird zuerst eine DVI-Datei erzeugt, anschliessend mit `dvipdfm` die PDF-Datei generiert.

**Voraussetzung** Installiertes `dvipdfm`.

**ps4dvi (*)** Es wird eine DVI-Datei mit psLATEX erzeugt.

**Voraussetzung** Installiertes psLATEX.

**pdf4ps (*)** Es wird eine DVI-Datei mit psLATEX erzeugt, anschliessend mit `dvips -Ppdf` die PS-Datei, daraufhin mit `epstopdf` die PDF-Datei.

**Voraussetzung** Installiertes `dvips`, `epstopdf`.

**eps** Die einzelnen Struktogramme werden als EPS-Dateien exportiert.

**Voraussetzung** Installiertes `dvips`.

**png** Die einzelnen Struktogramme werden als PNG-Dateien exportiert (Au¤ösung 100dpi).

**Voraussetzung** Installiertes GhostScript, Perl und das Script `epstopng`.

**png150** Die einzelnen Struktogramme werden als PNG-Dateien exportiert (Au¤ösung 150dpi).

**png300** Die einzelnen Struktogramme werden als PNG-Dateien exportiert (Au¤ösung 300dpi).

**png600** Die einzelnen Struktogramme werden als PNG-Dateien exportiert (Au¤ösung 600dpi).

**pdfp** Die einzelnen Struktogramme werden als PDF-Dateien exportiert.

**Voraussetzung** Installiertes pdfLATEX.

**dvi2pdf_pic (*)** Die einzelnen Struktogramme werden als PDF-Dateien exportiert.

**Voraussetzung** Installiertes dvipdfm.

**print** Es wird eine DVI-Datei erzeugt und zum Drucker geschickt.

**Voraussetzung** Installiertes `dvips`

Mit den Ausgabeoptionen sollten alle erdenklichen Alternativen für einen Import in *Textverarbeitungsprogramme* ala StarOffice abgedeckt sein[5].

[5] Weitere Erweiterungen sind natürlich denkbar.

# 2 PHP-Version

Im ersten Anlauf habe ich eine PHP-Version geschrieben, da ich dort schnellen Zugriff zu einem XML-Parser habe.

## 2.1 Voraussetzungen

- ☞ PHP 4.0 (**nur** mit PHP 4.0.6 getestet!)
- ☞ Es muss die XML-DOM Bibliothek zu PHP eingebunden sein.

## 2.2 Installation

### Makefile

Ist GNU-Make auf dem System installiert, muss nur die Datei `CONFIG` bearbeitet werden und die einzelnen Pfade angepasst werden:

`BIN_DIR` Verzeichnis für die ausführbaren Programme

`TETEX_DIR` Basisverzeichnis der TeTEX-Distribution

`XML_DIR` Systemverzeichnis für Document Type De£nitions

`EMACS_DIR` Basisverzeichnis zum Editor `emacs`

`PHP_DIR` Basisverzeichnis für PHP-Includes (PEAR)

`PHPBIN_DIR` Verzeichnis in dem `php.exe` zu £nden ist.

Anschliessend kann mit `make install-php` die Installation angestossen werden, eine Deinstallation mit `make uninstall-php`.

### Manuelle Installation

Die PHP-Anwendung `nsd2ltx` mit dem Verzeichnis `dtd` in den Includepfad von PHP kopieren, das Tool `php4nsd2ltx.php` in den allgemeinen Suchpfad kopieren, evtl. muss hier noch der Pfad zu PHP angepasst werden. Der Emacs-Mode `nsd-helper` in den Lisp-Suchpfad des Emacs kopieren und die `.emacs` entsprechend dem Beispiel aus `elisp/dot_emacs` anpassen. Die Dokumenttypen aus dem Verzeichnis `xml/DokuTransData/nsd/dtd/` müssen in den Systempfad des XML-Parsers gelegt werden oder lokal zu den erstellten NSD-Dateien liegen. Die Style-Datei `strukto.sty` aus dem Verzeichnis `styles/` in den Suchpfad `TEXINPUTS` kopieren.

## 2.3 Benutzung

Aufruf:

```
php4nsd2ltx <nsd-Datei> <AusgabeModus>
```

Das zweite Argument gibt die Art der Ausgabe an (s. Abschnitt 1.3).

Als drittes Argument kann folgender Wert übergeben werden:

**struktex** Ausgabe der Struktogramme für den Style StruktTEX (Default)

**nassi** Ausgabe der Struktogramme für den Style `nassi`

**flow** Ausgabe der Struktogramme für den Style `flow`

**Hinweis**
Für jedes Modul wird eine Hauptdatei[1] für LaTeX mit dem Namen des Moduls und der Endung `.ltx` gebildet. Für die Struktogramme innerhalb eines Moduls werden über die Struktogramm-Namen einzelne Dateien mit der Endung `.tex` erzeugt. Es sollte also darauf geachtet werden, das kein Struktogramm den Namen des Moduls trägt.

[1] Diese enthält die Dokumentenklasse und alle benötigten Styles, sowie die Umgebung `document`

# 3 C++ Version

## 3.1 Das Kommandozeilen-Programm

Mit dem Programm `nsd2ltx` und dem Emacs Helper-Mode `nsd-helper` lassen sich XML-Struktogramme nach LaTeX konvertieren.

### 3.1.1 Voraussetzungen

☞ Installierte `libxml2`

### 3.1.2 Installation

Mit `./configure` ist das Make£le zu erzeugen. Anschliessend ist mit `make install` das Programm zu installieren.

### 3.1.3 Benutzung

Aufruf:

```
  nsd2ltx [Options] file1 file2 ...

Options:
    -s    Call the external viewer after compilation
    -a    Don't call the external viewer asynchron!
    -b    Don't display the description before a structo
    -v    Print nearly all message to stdout
    -h    You are reading this help at the moment

Options with arguments:
    -m <mode>
          Which output mode should i use:
          Document Modes:
              tex, dvi, pdf, dvi2pdf, ps4dvi, pdf4ps, print
          Picture Modes:
              eps, png, png100, png150, png300, png600, pdfp, dvi2pdf_pic
    -d <directory>
        Generate the output in this *existing* directory
    -r <resfile>
        Parse this additional config file
```

Das Argument für den Ausgabemodus wurde in Abschnitt 1.3 erläutert. Das genaue Format zur Kon£gurationsdatei wird in Abschnitt A.2 beschrieben.

## 3.2 Der Wizard

Als kleine *Designstudie* habe ich ein Desktop-Programm entwickelt mit dem sich die Kon£gurationsoptionen bearbeiten lassen, sowie der Compilerlauf für Struktogrammdateien anstossen lässt.

### 3.2.1 Voraussetzungen

Es gibt folgende Versionen:

- ☞ QT2 mit KDE2 (Make£le: `Makefile.in.qt2`)
- ☞ QT3 mit KDE3 (Make£le: `Makefile.in.qt3`)

### 3.2.2 Installation

**QT2** Ist nur QT2 mit dem Designer 1.1 installiert, muss Make£le.in.qt2 nach Make£le.in kopiert werden.

**QT3** Ist nur QT3 mit dem Desigern 2.0 installiert, muss Make£le.in.qt3 nach Make£le.in kopiert werden.

Sorry für diese Mühe, aber besser habe ich es noch nicht hinbekommen mit autoconf etall.

Anschliessend kann mit `./configure --enable-kde(2|3)` das Make£le erzeugt werden und mit `make install` das Programm installiert werden.

### 3.2.3 Benutzung

Im nachfolgenden werden die einzelnen Karteikarten beschrieben.

**nsd2tx**

Auf der Karteikarte **nsd2ltx** (s. Abb. 3.1) lassen sich die Parameter zur Programmsteuerung manipulieren.

**Output** Ausgabeformat

**DVI** Normale Konvertierung als LaTeX-Dokument mit `latex` und anschliessendem Aufruf des DVI-Viewers.

**TEX** Normale Konvertierung als LaTeX-Dokument mit `latex`, es werden nur die LaTeX-Dateien erzeugt.

**PDF** Normale Konvertierung als LaTeX-Dokument mit `pdflatex`. Vorsicht: `pdflatex` kann nicht alle Erweiterungen der Pakete `eepic` bzw. `epic`.

**DVI2PDF** Konvertierung der DVI-Datei mit `dvipdfm` nach PDF.

**PSDVI** Normale Konvertierung als LaTeX-Dokument mit `pslatex`.

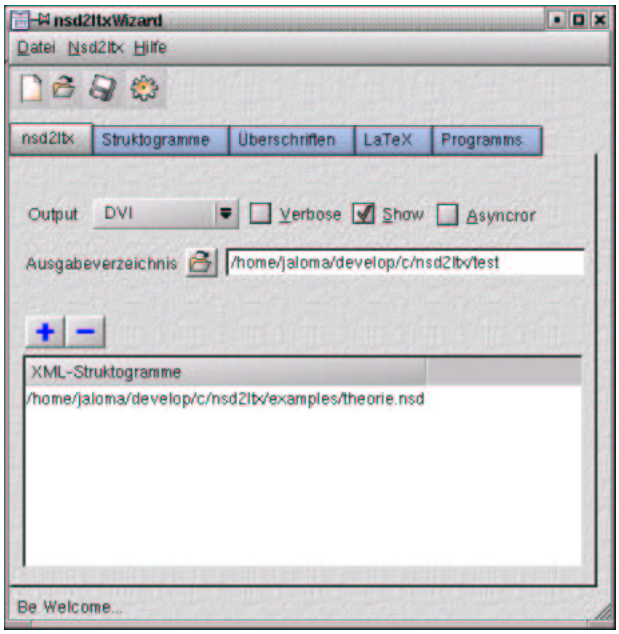



Abbildung 3.1: Optionen zur Steuerung von nsd2ltx

**PDF4PS** Normale Konvertierung als LaTeX-Dokument mit `pslatex` und konvertierung nach PDF.

**EPS** Die einzelnen Struktogramme werden als EPS-Dateien exportiert.

**PNG, PNG (100dpi, 150dpi, 300dpi, 600dpi)** Die einzelnen Struktogramme werden als PNG-Dateien in der entsprechenden Au¤ösung exportiert.

**PDF (pics)** Die einzelnen Struktogramme werden als PDF-Dateien exportiert. Da hier der Weg über EPS gegangen wird, sind alle Unstimmigkeiten der `pdflatex`-Konvertierung nicht gegeben.

**DVI2PDF (pics)** Die einzelnen Struktogramme werden als PDF-Dateien über `dvipdfm` exportiert.

**Verbose** Schalter ob die Ausgabe der Kompilierung unterdrückt werden soll oder nicht.

**Show** Schalter ob nach Abschluss der Kompilierung der externe Viewer aufgerufen werden soll oder nicht.

**Asyncron** Schalter ob nach Abschluss der Kompilierung der externe Viewer asyncron aufgerufen werden (z.B. acroread &) soll oder nicht. Diese Option muss für den Emacs-Mode `nsd-helper` abgestellt sein!

**Ausgabeverzeichnis** Die Ausgabe wird in diesem Verzeichnis abgelegt.

**XML-Struktogramme** Liste der XML-Struktogramme die verarbeitet werden sollen.

- **+** Dateien hinzufügen
- **−** Angewählte Datei aus der Liste entfernen

### Struktogramme

Auf der Karteikarte **Struktogramme** (s. Abb. 3.2) lassen sich die Parameter zu den Struktogrammen manipulieren.

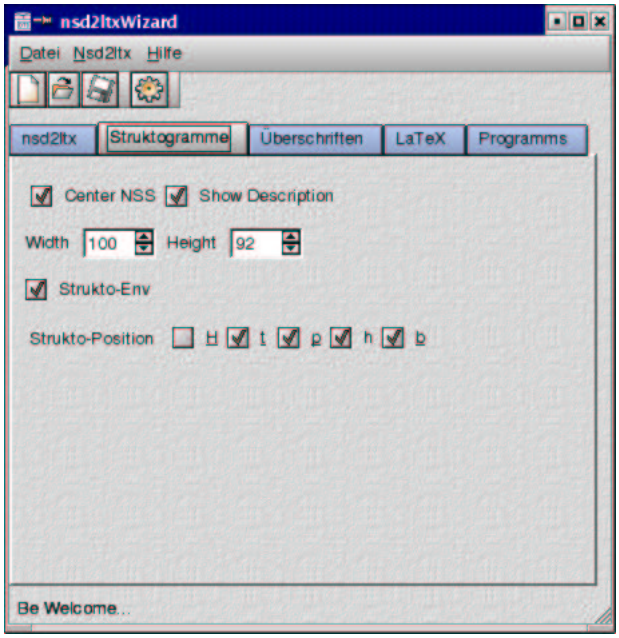



Abbildung 3.2: Optionen zur Steuerung des Verhalten der Struktogramme

**Strukto-Env** Schalter ob die einzelnen Struktogramme in eine Fliessumgebung eingebunden werden sollen oder nicht. Dazu wird zusätzlich der Style `strukto` benötigt.

**Strukto-Position** Mit diesen Schaltern kann der Positionsparamter der Fliessumgebung gesteuert werden. Bei der Option `H` wird der Style `float` benötigt.

**Center NSS** Schalter ob die einzelnen Struktogramme in eine `centernss`-Umgebung eingebunden werden sollen oder nicht.

**Show Description** Schalter ob die Ausgabe des Elements <description> unterdrückt werden soll oder nicht.

**Width** Standardbreite für ein Struktogramm, sofern das Attribut `width` bei dem `struktogramm`-Element nicht gesetzt ist.

**Height** Standardhöhe für ein Struktogramm, sofern das Attribut `height` bei dem `struktogramm`-Element nicht gesetzt ist.

### Überschriften

Auf der Karteikarte **Überschriften** (s. Abb. 3.3) lassen sich die Parameter zu den Überschriften in den Struktogrammen manipulieren.

**Funktionsparameter** Überschrift zur Beschreibung der Parameter einer Funktion.

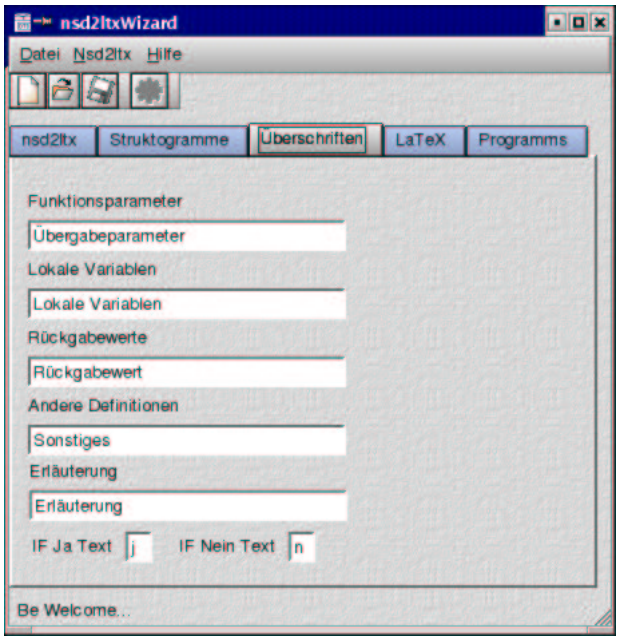



Abbildung 3.3: Eingabe der Überschriften

**Lokale Variablen** Überschrift zur Beschreibung der lokalen Variablen einer Funktion.

**Rückgabewerte** Überschrift zur Beschreibung der Rückgabewerte einer Funktion.

**Andere De£nitionen** Überschrift zur Beschreibung der Variablen in der Hauptfunktion.

**Erläuterung** Überschrift zur Beschreibung der Funktion.

**IF Ja Text** Text für den `true`-Zweig einer IF-Abfrage.

**IF Nein Text** Text für den `false`-Zweig einer IF-Abfrage.

### LaTeX

Auf der Karteikarte **LaTeX** (s. Abb. 3.4) lassen sich die Dokumentklasse und weitere Pakete verändern. Vorsicht: Ein `\begin{document}` oder gar ein `\end{document}` darf in diesen Texten nicht auftauchen.

**Document** LaTeX-Dateikopf für ein normales Dokument.

**Picture** LaTeX-Dateikopf für den Export der Struktogramme als EPS- oder PNG-Datei.

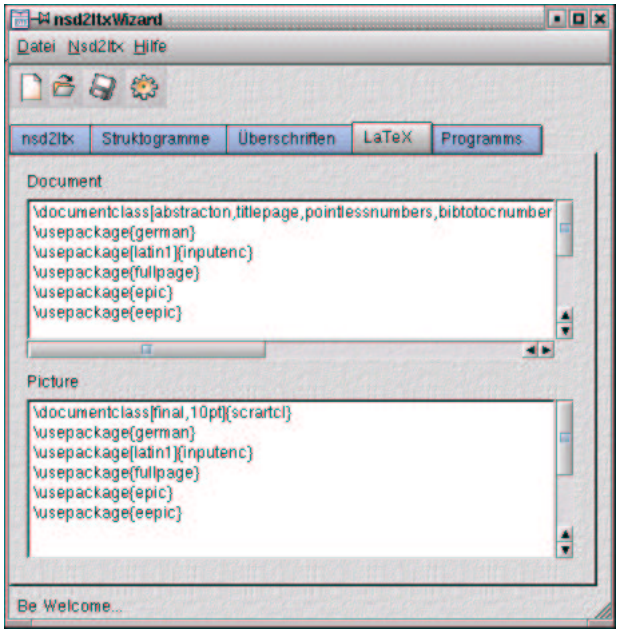



Abbildung 3.4: Eingabe der Dokumentklassen-Optionen

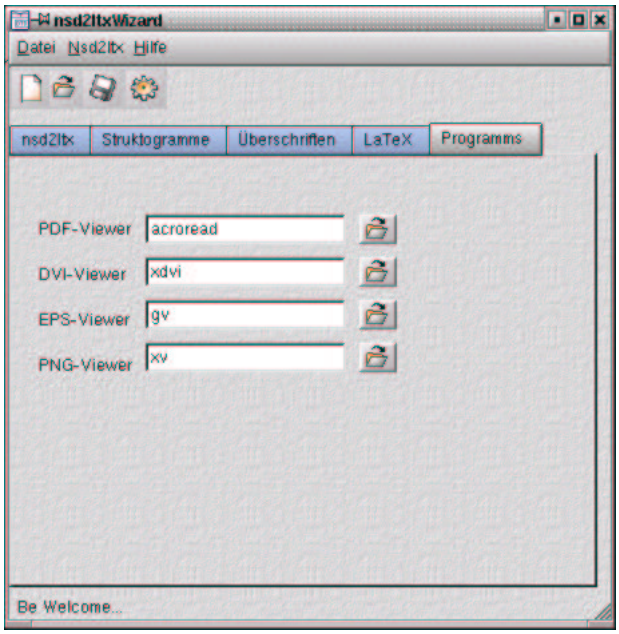



Abbildung 3.5: Eingabe der externen Programme

### Programms

Auf der Karteikarte **Programms** (s. Abb. 3.5) lassen sich die externen Viewer einstellen.

**PDF-Viewer** Externer PDF-Viewer (Default: xpdf).

**DVI-Viewer** Externer DVI-Viewer (Default: xdvi).

**EPS-Viewer** Externer EPS-Viewer (Default: gv).

**PNG-Viewer** Externer PNG-Viewer (Default: xv).

# 4 Ergänzende Hilfsmittel

## 4.1 Emacs-Mode nsd-helper

Im folgenden werden die einfachen Tastenbefehle zur Erstellung von Nassi-Shneiderman Diagrammen mit dem Emacs-Mode `nsd-helper` erläutert:

**CTRL-c CTRL-e** Einfügen eines neuen Elements. Ist die DTD vorhanden und konnte vom Emacs-Mode `psgml` eingeladen werden, so werden nur zugelassene Element zur Auswahl angezeigt.

**CTRL-c CTRL-a** Zum aktuellen Element können jetzt die Attribute bearbeitet werden, insbesondere die optionalen Attribute.

**CTRL-c +** Zum aktuellen Element kann jetzt im Minipuffer (Mini-Buffer) ein Attribut angewählt und bearbeitet werden.

**CTRL-c CTRL-v** Der externe Parser (nsgmls) wird aufgerufen und die XML-Datei wird geprüft.

**CTRL-c v** Eigentliche Funktionalität von `nsd-helper`: Es wird die externe PHP-Anwendung `php4nsd2ltx` mit der aktuellen Datei aufgerufen.

**CTRL-c CTRL-x** Eigentliche Funktionalität von `nsd-helper`: Es wird die externe Anwendung `nsd2ltx` mit der aktuellen Datei aufgerufen.

## 4.2 Style strukto

Der Style `strukto` beinhaltet eine De£nition zu einer Fliessumgebung ala `figure` und die Möglichkeit mit `\listofstruktos` ein entsprechendes Verzeichnis ausdrucken zulassen. Die Überschrift zum Verzeichnis lässt sich mit `\def\liststruktoname{}` manipulieren und mit der Option `lortotoc` im Inhaltsverzeichnis aufführen.

## 4.3 epstopng

Das Skript `epstopng` ist im Grunde `epstopdf` mit dem Ausgabemodus `png` und der zusätzlichen Option `resolution`.

```
EPSTOPDF 2.5, 1999/05/06 - Copyright 1998,1999 by Sebastian Rahtz et al.
Syntax:  epstopdf [options] <eps file>
Options:
```

```
  --help:           print usage
  --outfile=<file>: write result to <file>
  --(no)filter:     read standard input   (default: false)
  --(no)gs:         run ghostscript       (default: true)
  --(no)compress:   use compression       (default: true)
  --(no)hires:      scan HiresBoundingBox (default: false)
  --(no)exact:      scan ExactBoundingBox (default: false)
  --(no)debug:      debug informations    (default: false)
  --resolution:     output resolution     (default: 600x600)
Examples for producing 'test.pdf':
  * epstopdf test.eps
  * produce postscript | epstopdf --filter >test.pdf
  * produce postscript | epstopdf -f -d -o=test.pdf
Example: look for HiresBoundingBox and produce corrected PostScript:
  * epstopdf -d --nogs -hires test.ps>testcorr.ps
```

# A Die Dokumenttypdefinitionen

Im nachfolgenden möchte ich die Struktur der Dokument Typ Definition (DTD) erläutern. Ich habe bewusst keine externen Programme (livedtd et all) zur Dokumentation benutzt, so kann ich die Erläuterungen etwas flexibler gestalten. Ich habe dafür folgendes Schema entwickelt.

<**elementname**> Name und Beschreibung des Elements

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| Alle Attribute zum Element | | | *Contentmodel* des Elements mit einem kleinen Beispiel. | |

Dabei gilt folgendes:

#### Erläuterungen der Attribute

**Modus** Pflicht → Das Attribut muss mit einem Wert belegt werden.
Optional → Das Attribut kann mit einem Wert belegt werden.

**Definiert in** Das Attribut wird in dieser DTD definiert.

#### Erläuterung der Elemente

In der rechten Spalte werden die Elemente aufgeführt, die innerhalb des aktuell beschriebenen Elementes auftreten können. Dabei haben die folgenden Einträge die Bedeutung:

**EMPTY** Das Element hat keine weiteren Kinderelemente

**PCDATA** Einfacher Text

***Elementname*** Es existiert eine Elementdefinition, die entsprechend *Wiederholung* in dem aktuell beschriebenen Element eingetragen werden kann. Das Element *%block* nimmt einen besonderen Status ein: Es umschreibt alle Elemente die *rekursiv* in einem Element auftreten können.

## A.1 Erläuterung der Struktogrammdefinition

#### Hinweis

Das wichtigste Attribut bei der anschliessenden Bearbeitung der Struktogramme, ist `height` bei dem Element `struktogramm`. Der Standardwert liegt bei 180 mm und veranschlagt damit eine komplette DIN A4 Seite. Weitere Attribute zur Änderung der Größe eines Struktogrammes sind `leftangle` und `rightangle` bei dem IF-Element und `angle` und `height` bei dem CASE-Element.

**modul** Hauptelement zu den Nassi-Shneiderman Diagrammen

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| name | Pflicht | struktogramm.dtd | struktogramm | Null oder viele |

Name des Moduls.
Das Attribut wird beim Export als Dateiname für die Hauptdatei verwendet

```
<modul name='TestProg'
description='Ein Beispiel'>
</modul>
```

| | | |
|---|---|---|
| description | Optional | struktogramm.dtd |

Weitere Beschreibung des Moduls
(ohne Verwendung z. Z.)

| | | |
|---|---|---|
| author | Optional | struktex.dtd |

Name des Autors
(ohne Verwendung z. Z.)

| | | |
|---|---|---|
| mailto | Optional | struktex.dtd |

EMail des Autors
(ohne Verwendung z. Z.)

| | | |
|---|---|---|
| date | Optional | struktex.dtd |

Datum der Erstellung des Moduls
(ohne Verwendung z. Z.)

**struktogramm** Hauptelement zu einem Struktogramm

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| name | Pflicht | struktogramm.dtd | description | Null oder einmal |
| | | | paramdecl | Null oder einmal |
| | | | localdecl | Null oder einmal |
| | | | declaration | Null oder viele |
| | | | %block | Null oder viele |

Name der Funktion.
Das Attribut wird beim Export als Dateiname verwendet sowie als Label für eine spätere Referenz

```
<struktogramm name='TestFunc'
height='80'> </struktogramm>
```

| | | |
|---|---|---|
| width | Optional | struktex.dtd |

StruktTEX-Attribut für die Breite

| | | |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

**description** Element zum beschreibenden Text zu einem Struktogramm

PCDATA

```
<description> Dieses Struktogramm
beschreibt...</description>
```

**paramdecl** Block in dem formale Variablen beschrieben werden.

| *Elementname* | *Wiederholung* |
|---|---|
| pvardecl | Einmal oder viele |

```
<paramdecl> </paramdecl>
```

**pvardecl** Eine einzelne Erläuterung einer formalen Variablen.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| name | Pflicht | struktogramm.dtd |

Name der Variablen

| | | |
|---|---|---|
| reftype | Pflicht | struktogramm.dtd |

Art der Referenzierung: in, inout, out

| | | |
|---|---|---|
| description | Pflicht | struktogramm.dtd |

Beschreibung der Verwendung

| | | |
|---|---|---|
| type | Optional | struktogramm.dtd |

Variablentyp, derzeit ohne weitere Auswertung.

EMPTY

```
<pvardecl name='eineVar' reftype='in'
description='Ich bin zu was ' />
```

**localdecl** Block in dem lokale Variablen beschrieben werden.

| *Elementname* | *Wiederholung* |
|---|---|
| lvardecl | Einmal oder viele |

```
<localdecl> </localdecl>
```

**lvardecl** Eine einzelne Erläuterung einer lokalen Variablen.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| name | Pflicht | struktogramm.dtd |

Name der Variablen

| | | |
|---|---|---|
| description | Pflicht | struktogramm.dtd |

Beschreibung der Verwendung

| | | |
|---|---|---|
| type | Optional | struktogramm.dtd |

Variablentyp, derzeit ohne weitere Auswertung.

EMPTY

```
<lvardecl name='eineVar'
description='Ich bin zu was ' />
```

**returndecl** Eine Erläuterung des Rückgabewertes.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| description | Pflicht | struktogramm.dtd |

Beschreibung des Rückgabewertes

| | | |
|---|---|---|
| type | Optional | struktogramm.dtd |

Variablentyp, derzeit ohne weitere Auswertung.

EMPTY

```
<returndecl description='Ich gebe was '
/>
```

**declaration** Block in dem Variablen (lokale, formale) beschrieben werden. Es sollte grundsätzlich `paramdecl` und `localdecl` zur Erläuterung der Variablen benutzt werden! Dieses Element ist eine Altlast aus der ersten Version.

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| title | Optional | struktogramm.dtd | vardecl | Einmal oder viele |

Titel für die nachfolgenden Deklarationen.

```
<declaration title='Formale Parameter'>
</declaration>
```

**vardecl** Eine einzelne Erläuterung einer Variablen.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| name | Pflicht | struktogramm.dtd |

Name der Variablen

| | | |
|---|---|---|
| description | Pflicht | struktogramm.dtd |

Beschreibung der Verwendung

| | | |
|---|---|---|
| type | Optional | struktogramm.dtd |

Variablentyp, derzeit ohne weitere Auswertung.

EMPTY

```
<vardecl name='eineVar' description='Ich
bin zu was ' />
```

**statement** Eine einfache Anweisung.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

PCDATA

```
<statement> Wir tun was </statement>
```

**assignment** Eine einfache Zuweisung.

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| lvalue | Pflicht | struktogramm.dtd |

EMPTY

```
<assignment lvalue='ichBekommeWas'
rvalue='ichBerechneWas' />
```

Variable die den berechneten Wert erhält

| | | |
|---|---|---|
| rvalue | Pflicht | struktogramm.dtd |

Berechnung

| | | |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

**if** Binäre Auswahlanweisung (wahr/falsch).

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| condition | Pflicht | struktogramm.dtd |

| *Elementname* | *Wiederholung* |
|---|---|
| then | Genau einmal |
| else | Einmal oder garnicht |

Bedingung der IF-Anweisung

```
<if condition='IstWahr'> <then>
</then> </if>
```

| | | |
|---|---|---|
| leftangle | Optional | struktex.dtd |

StruktTEX-Attribut für den Winkel der linken Linie

| | | |
|---|---|---|
| rightangle | Optional | struktex.dtd |

StruktTEX-Attribut für den Winkel der rechten Linie

| | | |
|---|---|---|
| yestext | Optional | struktex.dtd |

StruktTEX-Attribut für den Text `wahr`

| | | |
|---|---|---|
| notext | Optional | struktex.dtd |

StruktTEX-Attribut für den Text `falsch`

**then** Block für den *wahren* Fall in einer IF-Anweisung.

| *Elementname* | *Wiederholung* |
|---|---|
| %block | Null oder viele |

**else** Block für den *falschen* Fall in einer IF-Anweisung.

| *Elementname* | *Wiederholung* |
|---|---|
| %block | Null oder viele |

**case** Mehrfachauswahl

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| condition | Pflicht | struktogramm.dtd | switch | Genau einmal oder viele |

| | |
|---|---|
| Bedingung für die SWITCH-Auswahl | `<case condition='WarumWas'> </case>` |

| | | | |
|---|---|---|---|
| angle | Optional | struktex.dtd | |

StrukTEX-Attribut für den Winkel der Linie

| | | | |
|---|---|---|---|
| height | Optional | struktex.dtd | |

StrukTEX-Attribut für die Höhe

**switch** Auswahlschalter zur Mehrfachauswahl

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| choice | Pflicht | struktogramm.dtd | %block | Null oder viele |

| | |
|---|---|
| Wert der Auswahl | `<switch choice='IstKlasse'> </switch>` |

| | | | |
|---|---|---|---|
| position | Optional | struktex.dtd | |

StrukTEX-Attribut für die Position des Auswahltextes

**while** Kopfgesteuerte Schleife, intern wird die Bedingung für den Abbruch generiert.

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| condition | Pflicht | struktogramm.dtd | %block | Null oder viele |

| | |
|---|---|
| Bedingung, solange dies gilt wird die Schleife durchlaufen | `<while condition='SoLangeWahrIst'>`<br>`</while>` |

| | | | |
|---|---|---|---|
| width | Optional | struktex.dtd | |

StrukTEX-Attribut für die Breite

**dowhile** Fussgesteuerte Schleife, intern wird die Bedingung für den Abbruch generiert.

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| condition | Pflicht | struktogramm.dtd | %block | Null oder viele |

| | |
|---|---|
| Bedingung, solange dies gilt wird die Schleife durchlaufen | `<dowhile condition='SoLangeWahrIst'>`<br>`</dowhile>` |

| | | | |
|---|---|---|---|
| width | Optional | struktex.dtd | |

StrukTEX-Attribut für die Breite

**repeatuntil** Fussgesteuerte Schleife, intern wird die Bedingung für den Abbruch generiert.

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| condition | Pflicht | struktogramm.dtd | %block | Null oder viele |

| | |
|---|---|
| Bedingung, solange **bis** diese gilt wird die Schleife durchlaufen | `<repeatuntil condition='BISWahrIst'> </repeatuntil>` |
| width Optional struktex.dtd | |
| StrukTEX-Attribut für die Breite | |

**for** Kopfgesteuerte Schleife, die Bedingung für den Abbruch wird automatisch durch den Schleifenzähler generiert.

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| countervar | Pflicht | struktogramm.dtd | %block | Null oder viele |

| | |
|---|---|
| Zählervariable | `<for countervar='Jünger' start='Urknall' end='JüngsterTag' increment='Gebet'> </for>` |
| start Pflicht struktogramm.dtd | |
| Beginn der Schleife | |
| end Pflicht struktogramm.dtd | |
| Ende der Schleife | |
| increment Pflicht struktogramm.dtd | |
| Der Schleifenzähler wird um diesen Wert erhöht | |
| width Optional struktex.dtd | |
| StrukTEX-Attribut für die Breite | |

**loop** Endlosschleife. Abbruch durch return

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| width | Optional | struktex.dtd | %block | Null oder viele |

| | |
|---|---|
| StrukTEX-Attribut für die Breite | `<loop> </loop>` |

**exit** Aussprunganweisung aus einer Loop

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| value | Optional | struktogramm.dtd |

Rückgabewert

| | | |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

EMPTY

```
<exit value='EsKannWeiterGaehen' />
```

**call** Aufruf einer Unterroutine

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| subroutine | Pflicht | struktogramm.dtd |

Name der Subroutine die aufgerufenen wird

| | | |
|---|---|---|
| lvalue | Optional | struktogramm.dtd |

Hat die Subroutine einen Rückgabewert, kann hiermit der Speicherplatz angegeben werden

| | | |
|---|---|---|
| arguments | Optional | struktogramm.dtd |

Sollen an die Subroutine Argumente übergeben werden ist dieses Attribut zu füllen.

| | | |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

EMPTY

```
<call subroutine='WoAndersGehtEsWeiter'
/>
```

**return** Rücksprunganweisung aus dem Modul

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| value | Optional | struktogramm.dtd |

Rückgabewert

| | | |
|---|---|---|
| height | Optional | struktex.dtd |

StruktTEX-Attribut für die Höhe

EMPTY

```
<return value='IschBinFetisch' />
```

**%block** Elemente die geschachtelt auftreten können.

| *Elementname* | |
|---|---|
| statement | oder |
| assignment | oder |
| if | oder |
| case | oder |
| while | oder |
| for | oder |
| dowhile | oder |
| repeatuntil | oder |
| loop | oder |
| call | oder |
| return bzw. exit | |

## A.2 Erläuterung der Kon£gurationsdefinition

Zu dem Programm existiert eine Kon£gurationsdatei im XML-Format. Die Datei wird in der Suchreihenfolge `/etc/nsd2ltxrc`, `$HOME/.nsd2ltxrc` und `./.nsd2ltxrc` verarbeitet.

**nsd2ltx** Hauptelement der Kon£gurationsdatei

| *Elementname* | *Wiederholung* |
|---|---|
| application | Null oder einmal |
| modul | Null oder einmal |
| struktogramme | Null oder einmal |

**application** Element zur Steuerung der Applikation

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| verbose | Optional | application.dtd |

Ausgabe der Compilerläufe nicht unterdrücken.

| | | |
|---|---|---|
| show | Optional | application.dtd |

Nach Abschluss der Kompilation, die erzeugten Dateien mit den externen Programmen anzeigen.

| | | |
|---|---|---|
| asyncron | Optional | application.dtd |

Die externen Programme werden asyncron aufrufen.

| | | |
|---|---|---|
| mode | Optional | application.dtd |

Ausgabemodus

| *Elementname* | *Wiederholung* |
|---|---|
| outputdir | Null oder einmal |
| pdfviewer | Null oder einmal |
| dviviewer | Null oder einmal |
| epsviewer | Null oder einmal |
| pngviewer | Null oder einmal |

```
<application verbose='Off'
show='On'>
</application>
```

**outputdir** Element für den Pfad zum Ausgabeverzeichnis

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<outputdir>
/home/dubistes/in/deinem/verzeichnis
</outputdir>
```

**pdfviewer** Element für das externe PDF-Anzeigeprogramm

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<pdfviewer>
acroread </pdfviewer>
```

**dviviewer** Element für das externe DVI-Anzeigeprogramm

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<dviviewer>
xdvi </dviviewer>
```

**epsviewer** Element für das externe EPS-Anzeigeprogramm

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<epsviewer>
gv </epsviewer>
```

**pngviewer** Element für das externe PNG-Anzeigeprogramm

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<pngviewer>
xv </pngviewer>
```

**modul** Element zur Steuerung der LaTeX-De£nitionen

| *Attributname* | *Modus* | *De£niert in* | *Elementname* | *Wiederholung* |
|---|---|---|---|---|
| | | | texheader | Null oder einmal |
| | | | picheader | Null oder einmal |

```
<modul
>
</modul>
```

**texheader** Element für den Standardtext zu den LaTeX-Dokumentdateien

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<texheader>
\documentclass{book} </texheader>
```

**picheader** Element für den Standardtext zu den LaTeX-Picture Dateien

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<picheader>
\documentclass{article} </picheader>
```

**struktogramme** Element zur Steuerung der Struktogramme

| *Attributname* | *Modus* | *De£niert in* |
|---|---|---|
| withFloat | Optional | application.dtd |

Soll das Struktogramm in die Float-Umgebung `strukto` eingebunden werden.

| | | |
|---|---|---|
| ¤oatPosition | Optional | application.dtd |

Welchen Positionsparamater hat die Float-Umgebung `strukto` (H,htbp).

| | | |
|---|---|---|
| withCenterNSS | Optional | application.dtd |

Soll das Struktogramm in die Umgebung `centernss` eingebunden werden.

| | | |
|---|---|---|
| width | Optional | application.dtd |

Standardbreite für Struktogramme sofern kein Attribut `width` angegeben ist.

| | | |
|---|---|---|
| height | Optional | application.dtd |

Standardhöhe für Struktogramme sofern kein Attribut `height` angegeben ist.

| *Elementname* | *Wiederholung* |
|---|---|
| description | Null oder einmal |
| paramdecl | Null oder einmal |
| localdecl | Null oder einmal |
| returndecl | Null oder einmal |
| ifyestext | Null oder einmal |
| ifnotext | Null oder einmal |

```
<struktogramme withFloat='On'
withCenterNSS='Off'>
</struktogramme>
```

**description** Element für die Titelzeile zur Erläuterung eines Struktogrammes

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<description>
Beschreibung </description>
```

**paramdecl** Element für die Titelzeile zur Erläuterung der Funktionsparameter

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<paramdecl>
Funktionsparameter </paramdecl>
```

**localdecl** Element für die Titelzeile zur Erläuterung der lokalen Variablen

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<localdecl>
Lokale Variablen </localdecl>
```

**returndecl** Element für die Titelzeile zur Erläuterung der Rückgabewerte

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<returndecl>
Rückgabewert </returndecl>
```

**declaration** Element für die Titelzeile zu der Erläuterung von Variablen in einem Hauptprogramm

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<declacration>
Variablen </declacration>
```

**ifyestext** Element für den Standardtext des TRUE-Zweigs einer IF-Abfrage

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<ifyestext>
Ja </ifyestext>
```

**ifnotext** Element für den Standardtext des FALSE-Zweigs einer IF-Abfrage

| *Elementname* | *Wiederholung* |
|---|---|
| PCDATA | Null oder einmal |

```
<ifnotext>
Nein </ifnotext>
```

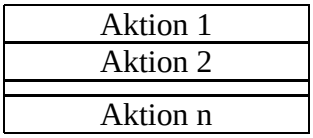
# B Ein bisschen Theorie

## Was sind Struktogramme

**Struktogramme**, auch *Nassi - Shneiderman - Diagramme* genannt, sind eine gra£sche Darstellungsmethode von Programmen im Sinne der strukturierten Programmierung.
Jeder Arbeitsschritt (hier *Aktion* genannt) eines Programmes bildet einen Strukturblock. Algorithmen stellen Kombinationen von Strukturblöcken dar. Die möglichen Kombinationen in Struktogrammen entsprechen den Steuerstrukturen der strukturierten Programmierung.

## Erläuterung der Steuerstrukturen

**Sequenz.** Sie stellt eine Aneinanderreihung von Aktionen dar. Dies entspricht einer Ausführung der

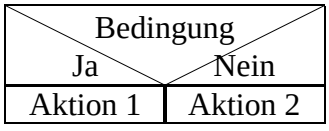



Struktogramm 2: Sequenz

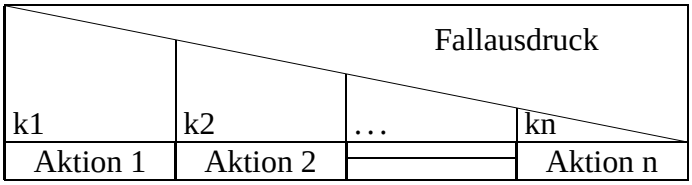
Aktionen in der angegebenen Reihenfolge (↪ Struktogramm 2).

**Alternative.** Die Alternative erfordert das Abprüfen einer Bedingung, von deren Erfülltsein das Auswählen von Aktionen abhängt. Bei der *vollständigen Alternative* (binäre Auswahl) wird in Abhängigkeit vom Erfülltsein einer Bedingung eine von zwei Aktionen ausgeführt (↪ Struktogramm 3).

**Fallauswahl.** Eine Erweiterung stellt die Fallauswahl dar (↪ Struktogramm 4). Jede der möglichen auswählbaren Aktionen wird durch einen Kennwert charakterisiert. Bei Programmausführung wird über den Wert des *Fallausdrucks* die Aktion ausgewählt und ausgeführt, deren Kennwert mit dem Wert des Fallausdrucks übereinstimmt. Wird keine Übereinstimmung erreicht, wird die zu sonst gehörende Aktion ausgeführt[1].




Struktogramm 3: Vollständige Alternative



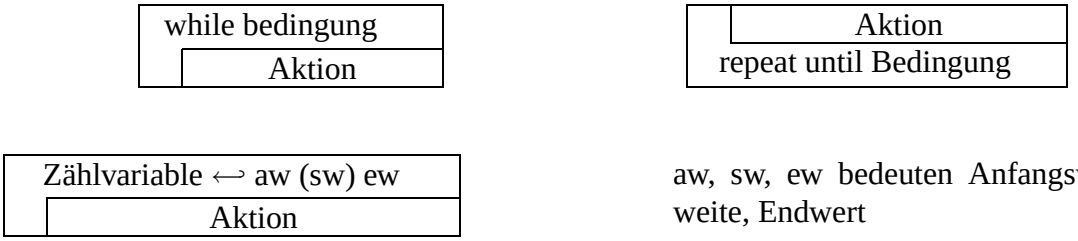

Struktogramm 4: Fallauswahl

[1] StruktTEX bildet momentan den `default`(sonst)-Zweig nicht ab.

**Schleifen.** Sie ermöglichen ein wiederholtes Ausführen von Aktionen unter Auswertung von Schleifenbedingungen. In Abhängigkeit davon, wann und wie die Schleifenbedingung formuliert ist, unterscheidet man verschiedene Schleifen (↪ Struktogramm 5). Bei *Abweisschleifen* (kopfgesteuerte Schleifen) wird vor *jedem Schleifendurchlauf* geprüft, ob die *Ausführbedingung* erfüllt ist. Der *Extremfall* ist das *nullmalige* Ausführen einer Schleife.

Bei *Wiederholschleifen* (fussgesteuerte Schleifen) wird *nach jedem Schleifendurchlauf* geprüft, ob die Abbruchbedingung erreicht ist oder die Schleife wiederholt zu durchlaufen ist. Die Schleife wird mindestens einmal durchlaufen.

Ein Beispiel zeigt Struktogramm 6.




Struktogramm 5: Schleifen


Struktogramm 6: Berechnung der Summe von $n$ Zahlen

## Strukturierte Programmierung

**Strukturierte Programmierung.** Sie stellt eine systematische Entwicklungsmethode für Einzelprogramme dar und wird auch als *systematisches Programmieren* bezeichnet.

Die strukturierte Programmierung beinhaltet das *Top-down-Vorgehen* beim Erarbeiten eines Algorithmus unter Verwendung der Steuerstrukturen und anschließendes Formulieren in einer Programmiersprache, die möglichst solche Steuerstrukturen enthält. Das trifft beispielsweise auf Pascal zu. Ansonsten sollten die Steuerstrukturen mit anderen Anweisungen nachgebildet werden.

Diese Methode wurde etwa 1970 entwickelt. Sie war der Beginn des ingenieurmäßigen Vorgehens bei der Erstellung von Programmen.

**Top-Down-Entwicklung.** Top-down (von oben nach unten) bedeutet vom Groben zum Feineren bzw. vom Allgemeinen zum Detaillierten und wird auch als schrittweise Verfeinerung bezeichnet.

Für die Lösung eines Problems bedeutet die Top-down-Entwicklung das wiederholte Zerlegen des Problemes in Teilprobleme und Beziehungen zwischen diesen so lange, bis die Teilprobleme lösbar

sind, d. h. Algorithmen aufstellbar oder vorhandene nutzbar sind. Die Zerlegung ist problemabhängig. Die Menge der Einzellösungen bildet, durch Steuerstrukturen kombiniert, die Lösung des Problems.

Das Top-down-Vorgehen führt zu gut *gegliederten Programmen*. Es fördert das Bereitstellen und Nutzen von allgemeingültigen Teilleistungen, möglichst sofort in einer Programmiersprache.

**Bottom-Up-Entwicklung.** Das Gegenstück zur *Top-down*-Vorgehensweise stellt die *Bottom-up*-Entwicklung dar. Hier wird versucht, überschaubare Teilprobleme zu lösen bzw. vorhandene Lösungen für solche zu nutzen und durch Zusammensetzen schrittweise das Gesamtproblem zu lösen.

In der Praxis ist ein Kompromiß zwischen beiden Arbeitsprinzipien oft zweckmäßig, der in einem wiederholten Wechsel zwischen den Prinzipien besteht und als *Jo-Jo*-Verfahren bezeichnet wird.

# C Informationen

## Danksagung

Mein Dank geht an Karen, Hanno und Markus für Ihre Erfahrungsberichte über die Benutzung der ersten Version, sowie für Ihre Hinweise zu der Dokumentation dieser Anwendung.

Ein besonderer Dank geht noch an Hanno, der mir zur Kompilation der C-Anwendung unter anderen Linux-Systemen wichtige Hinweise geben konnte.

## Lizenz



### Gewährleistung

# Autor

Über Anregungen bzw. Fehlermeldungen würde ich mich freuen.
Jürgen A. Lamers (jaloma@dokutransdata.de)

## Homepage

http://www.dokutransdata.de/nsd2ltx/

# Entwicklungsgeschichte

### 2.0.1 (2002-10-28)

- Für die Fliessumgebung `strukto` ist der Positionsparameter über die Kon£gurationsdatei und im Wizard beein¤ussbar.
- Weitere Ausgabemodi hinzugefügt: psLaTeX, dvipdfm.

### 2.0 (2002-07-25)

- Es existiert jetzt ein C++ Anwendung inkl. Wizard. Diese Anwendung ist wesentlich schneller und kon£gurierbarer.

### 1.2.3 (2002-06-01)

- Bei Funktionen sollte in der Beschreibung zur Funktion der Rückgabewert besonders ausgezeichnet werden, deswegen wurde ein neues Element `returndecl` zur Dokument Typ De£nition ergänzt.
- Wenn die Struktogramme in eine laufende Dokumentation ohne weitere Erklärung eingebaut werden, sind diese Struktogramme nicht unbedingt selbstredend, deswegen wurde die Dokument Typ De£nition um ein Element `description` für Struktogramme ergänzt.

### 1.2.2 (2002-03-12)

- Fix: Element `Exit` war als PHP-Klasse vorhanden, aber nicht in der Dokument Typ De£nition aufgenommen.

### 1.2.1 (2002-02-25)

- Bug£x: Im sogenannten Picture-Mode sind die Unterschriften der Fliessumgebung `strukto` über¤üssig.
- Das Element `for` hat jetzt zwingend ein Attribut `countervar`, zur Verdeutlichung dieser *besonderen* Schleife.
- Element `assignment` zur logischen Auszeichnung von Zuweisungen ergänzt.
- Das Element `call` hat jetzt ein Attribut `lvalue` um den Rückgabewert auszuzeichnen.
- Im Emacs-Mode kann mit `nsd-submit-bug-report` eine EMail an den Entwickler gesendet werden.

### 1.2 (2002-02-24)

- Einzelne Struktogramme können jetzt als EPS, PNG, PDF exportiert werden.
- Ausdruck der DVI-Datei ermöglicht.
- Attribute `author`, `mailto` und `date` für das Element `modul` ergänzt.
- Einfache Install-Routine zum Make£le ergänzt.

**1.1 (2002-02-21)**

- Das Element `declaration` wurde aufgeteilt in `paramdecl` und `localdecl`.
- Die StruktTEX-Ausgabe für Deklarationen wurde ausgebessert.
- Der Style `nassi.sty` wird unterstützt.

**1.0 (2002-01-31)**

- Erste Veröffentlichung

# Abbildungsverzeichnis

1.1 Einfaches Beispiel . . . . . 2

3.1 Optionen zur Steuerung von nsd2ltx . . . . . 9
3.2 Optionen zur Steuerung des Verhalten der Struktogramme . . . . . 10
3.3 Eingabe der Überschriften . . . . . 11
3.4 Eingabe der Dokumentklassen-Optionen . . . . . 12
3.5 Eingabe der externen Programme . . . . . 12

# Struktogrammverzeichnis


1 Simple Example . . . . . 2
2 Sequenz . . . . . 29
3 Vollständige Alternative . . . . . 29
4 Fallauswahl . . . . . 29
5 Schleifen . . . . . 30
6 Berechnung der Summe von $n$ Zahlen . . . . . 30

# Stichwortverzeichnis


## A
Abweisschleife 30
Aktion 29
Alternative 29
  – vollständige 29
assignment 20
Ausführbedingung 30

## B
block 25
Bottom-Up-Entwicklung 31

## C
call 24
case 21

## D
declaration 20
description 18
Document Type De£nition 17
Dokument Typ De£nition 17
dowhile 22
DTD 17

## E
else 21
exit 24

## F
Fallausdruck 29
Fallauswahl 29
for 23
fussgesteuerte Schleife 30

## G
G.E.S.y 1
Gewährleistung 33

## I
if 21
Installation 5
  – Manuelle 5

## J
Jo-Jo-Verfahren 31

## K
kopfgesteuerte Schleife 30

## L
Lizenz 33
localdecl 19
loop 23
lvardecl 19

## M
Make£le 5
modul 18

## N
Nassi-Shneiderman-Diagramm 1, 29

## P
paramdecl 19
Programmierung
  – Strukturierte 30
  – Systematische 30
pvardecl 19

## R
repeatuntil 23
return 24
returndecl 19

## S
Schleife 30
  – , fussgesteuerte 30
  – , kopfgesteuerte 30
  – Abweis- 30
  – Wiederhol- 30
Schleifenbedingung 30

Sequenz 29
statement 20
Steuerstruktur 30
StrukTeX 1
struktogramm 18
Struktogramm 29
Strukturblock 29
Strukturierte Programmierung 30
switch 22
Systematische Programmierung 30

## T

then 21
Top-Down-Entwicklung 30

## V

vardecl 20
Verfeinerung,
- schrittweise 30

## W

while 22
Wiederholschleife 30