

Handbuch  
Version 2.0.1

**nsd2ltx**  
**Struktogramme mit XML**

Jürgen A. Lamers  
jaloma@dokutransdata.de

13. November 2002

DokuTransData

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht. Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Herausgeber und Autor können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Für Verbesserungsvorschläge und Hinweise auf Fehler sind Autor und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Fast alle Hardware- und Softwarebezeichnungen, die in diesem Handbuch erwähnt werden, sind gleichzeitig auch eingetragene Warenzeichen oder sollten als solche betrachtet werden.

## **Beschreibung der Anwendung nsd2ltx**

© 2002 DokuTransData, Altstr. 112, 52066 Aachen, Germany.

Alle Rechte vorbehalten

*Für Karen*



## Zusammenfassung

Beschreibung der XML-Dokumentdefinitionen `struktogramm.dtd` und `struktex.dtd` in Verbindung mit der Anwendung `nsd2ltx`.

`nsd2ltx` konvertiert XML-Struktogramme in  $\text{\LaTeX}$ -Struktogramme. Die XML-Dateien müssen der Dokument Typ Definition `struktex.dtd` genügen. Die erzeugten  $\text{\LaTeX}$ -Dateien benutzen den Befehlssatz aus dem Paket `struktex`.



# Inhaltsverzeichnis

<b>1 Die Idee</b>	<b>1</b>
1.1 Einleitung . . . . .	1
1.2 Generelle Voraussetzungen . . . . .	2
1.3 Ausgabemodi . . . . .	3
<b>2 PHP-Version</b>	<b>5</b>
2.1 Voraussetzungen . . . . .	5
2.2 Installation . . . . .	5
2.3 Benutzung . . . . .	6
<b>3 C++ Version</b>	<b>7</b>
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
<b>4 Ergänzende Hilfsmittel</b>	<b>15</b>
4.1 Emacs-Mode nsd-helper . . . . .	15
4.2 Style strukto . . . . .	15
4.3 epstopng . . . . .	15
<b>A Die Dokumenttypdefinitionen</b>	<b>17</b>
A.1 Erläuterung der Struktogrammdefinition . . . . .	17
A.2 Erläuterung der Konfigurationsdefinition . . . . .	25
<b>B Ein bisschen Theorie</b>	<b>29</b>
<b>C Informationen</b>	<b>33</b>
<b>Abbildungsverzeichnis</b>	<b>37</b>
<b>Struktogrammverzeichnis</b>	<b>39</b>
<b>Stichwortverzeichnis</b>	<b>41</b>



# 1 Die Idee

## 1.1 Einleitung

Ich habe für Linux bis jetzt noch keinen Wysiwyg-Editor für Struktogramme<sup>1</sup> 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  $\text{\LaTeX}$  und dem Style  $\text{\Stu\kernTeX}$  erzeugt wird, d. h. ich muss die graphisch erzeugten Struktogramme weiter verarbeiten können<sup>2</sup>. 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<sup>3</sup>.

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  $\text{\LaTeX}$  programmieren, darum habe ich einen kleinen Emacs-Mode zur Anbindung der PHP-Anwendung `php4nsd21tx` bzw. C++-Anwendung `nsd21tx` geschrieben, der On-The-Fly DVI- oder PDF-Dokumente erzeugt und die farbliche Gestaltung der XML-Dateien übernimmt.

### Vorteile

- ☞ Emacs, (PHP) und  $\text{\LaTeX}$  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) $\text{\LaTeX}$  und  $\text{\Stu\kernTeX}$

---

<sup>1</sup>Wer keine Ahnung von Struktogrammen hat, soll sich sofort zu Abschnitt B begeben ;-)

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

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

## 1.2 Generelle Voraussetzungen

☞ L<sup>A</sup>T<sub>E</sub>X-Installation mit den Styles St<sub>U</sub>K<sub>T</sub>E<sub>X</sub> und Float. Je nach Ausgabeart müssen noch

- pdfL<sup>A</sup>T<sub>E</sub>X
- psL<sup>A</sup>T<sub>E</sub>X
- dvips
- ghostscript
- dvi<sub>2</sub>pdfm
- epstopng<sup>4</sup>
- epstopdf

installiert sein.

☞ Es sollte der Emacs-Mode psgml installiert sein

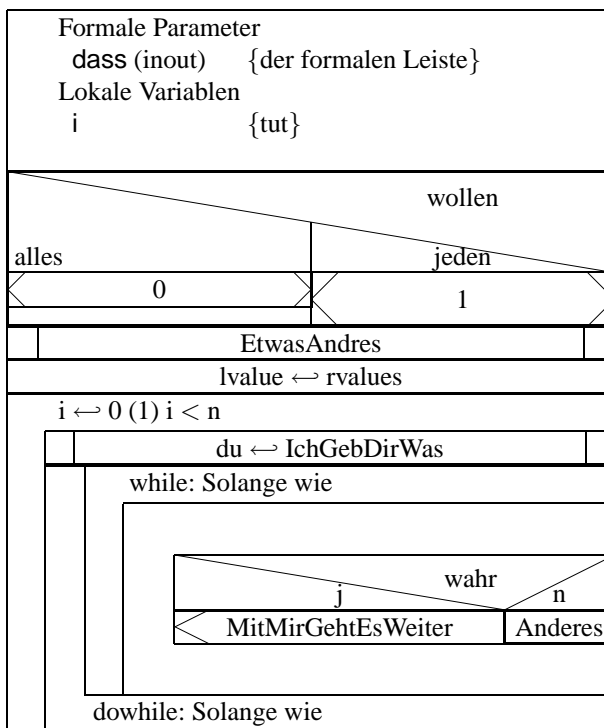
## Beispiele

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

Simple Example



Struktogramm 1: Simple Example

<sup>4</sup>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  $\text{\TeX}$ -Dateien angelegt. Anschliessend wird  $\text{\LaTeX}$  und der DVI-Viewer `xdvi` aufgerufen.

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

**Voraussetzung** Installiertes `pdf $\text{\LaTeX}$` .

**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 `ps $\text{\LaTeX}$`  erzeugt.

**Voraussetzung** Installiertes `ps $\text{\LaTeX}$` .

**pdf4ps (\*)** Es wird eine DVI-Datei mit `ps $\text{\LaTeX}$`  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 (Auflösung 100dpi).

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

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

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

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

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

**Voraussetzung** Installiertes `pdf $\text{\LaTeX}$` .

**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<sup>5</sup>.

<sup>5</sup>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 Te<sub>T</sub>E<sub>X</sub>-Distribution  
`XML_DIR` Systemverzeichnis für Document Type Definitions  
`EMACS_DIR` Basisverzeichnis zum Editor emacs  
`PHP_DIR` Basisverzeichnis für PHP-Includes (PEAR)  
`PHPBIN_DIR` Verzeichnis in dem `php.exe` zu finden 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  $\text{Strukt}_{\text{TeX}}$  (Default)

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

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

### Hinweis

Für jedes Modul wird eine Hauptdatei<sup>1</sup> für  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  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.

---

<sup>1</sup>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  $\text{\LaTeX}$  konvertieren.

### 3.1.1 Voraussetzungen

☞ Installierte `libxml2`

### 3.1.2 Installation

Mit `./configure` ist das Makefile 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 Konfigurationsdatei wird in Abschnitt A.2 beschrieben.

## 3.2 Der Wizard

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

### 3.2.1 Voraussetzungen

Es gibt folgende Versionen:

- ☞ QT2 mit KDE2 (Makefile: Makefile.in.qt2)
- ☞ QT3 mit KDE3 (Makefile: Makefile.in.qt3)

### 3.2.2 Installation

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

**QT3** Ist nur QT3 mit dem Designer 2.0 installiert, muss Makefile.in.qt3 nach Makefile.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 Makefile 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  $\text{\LaTeX}$ -Dokument mit `latex` und anschliessendem Aufruf des DVI-Viewers.

**TEX** Normale Konvertierung als  $\text{\LaTeX}$ -Dokument mit `latex`, es werden nur die  $\text{\LaTeX}$ -Dateien erzeugt.

**PDF** Normale Konvertierung als  $\text{\LaTeX}$ -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  $\text{\LaTeX}$ -Dokument mit `pslatex`.

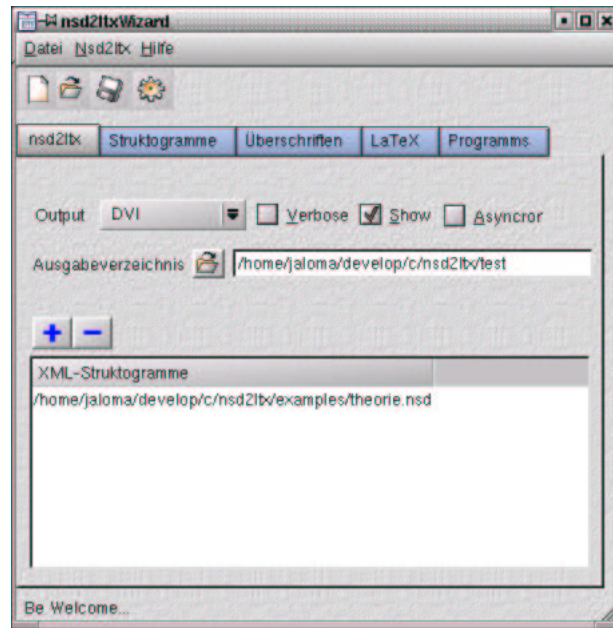


Abbildung 3.1: Optionen zur Steuerung von nsd2ltx

**PDF4PS** Normale Konvertierung als  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -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 Auflösung 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.

**Asynchron** Schalter ob nach Abschluss der Kompilierung der externe Viewer asynchron 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.

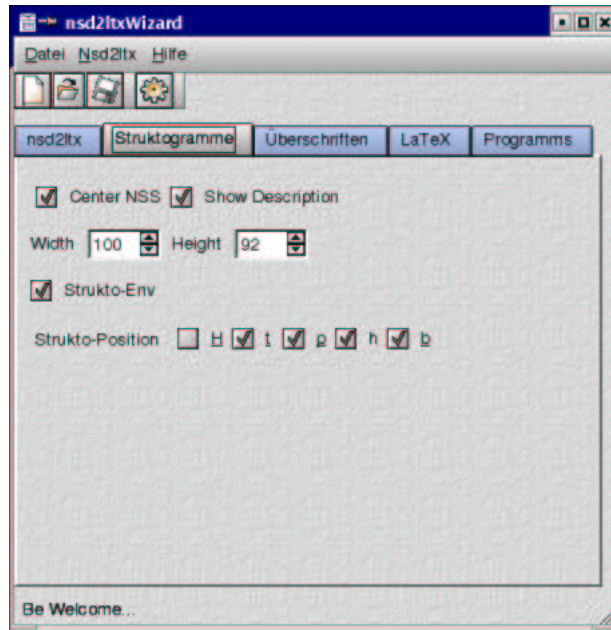


Abbildung 3.2: Optionen zur Steuerung des Verhalten der Struktogramme

**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 Positionsparameter 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 `strukto`-Element nicht gesetzt ist.

**Height** Standardhöhe für ein Struktogramm, sofern das Attribut `height` bei dem `strukto`-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.

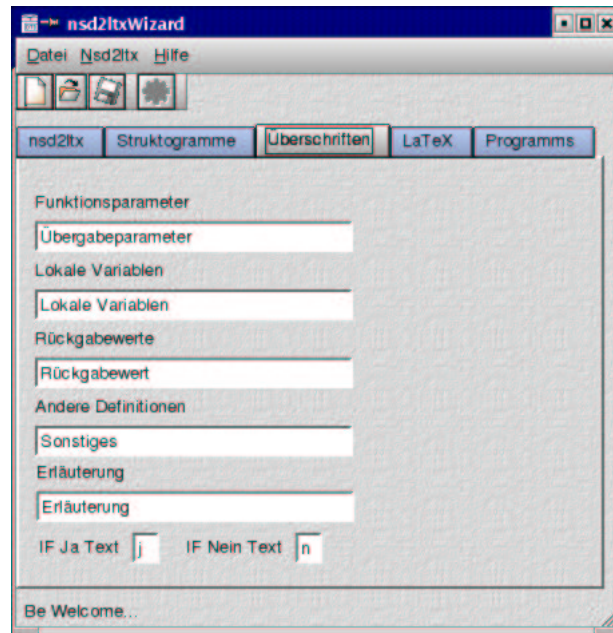


Abbildung 3.3: Eingabe der Überschriften

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

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

**Andere Definitionen** 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.

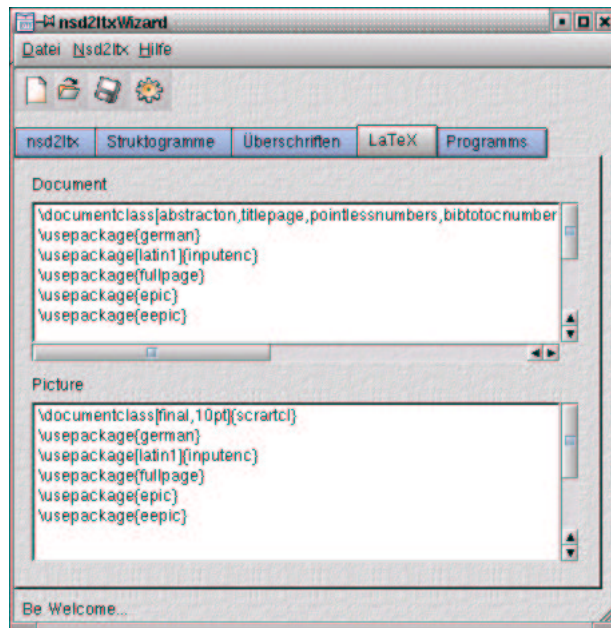


Abbildung 3.4: Eingabe der Dokumentklassen-Optionen

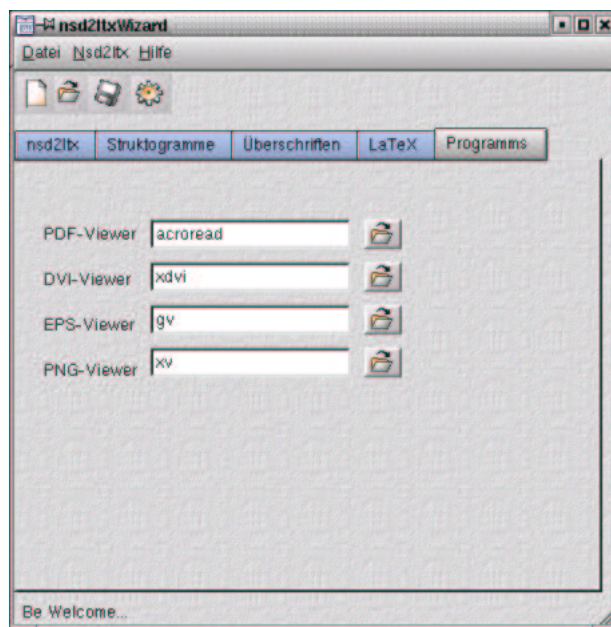


Abbildung 3.5: Eingabe der externen Programme

**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 Definition 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.

< <b>elementname</b> >	Name und Beschreibung des Elements			
<i>Attributname</i>	<i>Modus</i>	<i>Definiert in</i>	<i>Elementname</i>	<i>Wiederholung</i>
	Alle Attribute zum Element			<i>Contentmodel</i> 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	Definiert 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

E-Mail 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	Definiert in	Elementname	Wiederholung
name	Pflicht	struktogramm.dtd	description	Null oder einmal

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

```
paramdecl Null oder einmal
localdecl Null oder einmal
declaration Null oder viele
%block Null oder viele
```

width Optional struktex.dtd

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Breite

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

height Optional struktex.dtd

St<sub>u</sub>kT<sub>E</sub>X-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.

<i>Elementname</i>	<i>Wiederholung</i>
paramdecl	Einmal oder viele

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

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

<i>Attributname</i>	<i>Modus</i>	<i>Definiert in</i>	EMPTY
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.

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

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

<i>Elementname</i>	<i>Wiederholung</i>
localdecl	Einmal oder viele

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

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

<i>Attributname</i>	<i>Modus</i>	<i>Definiert in</i>	EMPTY
name	Pflicht	struktogramm.dtd	

Name der Variablen

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

Beschreibung der Verwendung

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

Variablentyp, derzeit ohne weitere Auswertung.

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

**returndecl** Eine Erläuterung des Rückgabewertes.

Attributname	Modus	Definiert in
description	Pflicht	struktogramm.dtd

EMPTY

Beschreibung des Rückgabewertes

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

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

Variablentyp, derzeit ohne weitere Auswertung.

**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	Definiert in
title	Optional	struktogramm.dtd

Elementname	Wiederholung
vardecl	Einmal oder viele

Titel für die nachfolgenden Deklarationen.

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

**vardecl** Eine einzelne Erläuterung einer Variablen.

Attributname	Modus	Definiert in
name	Pflicht	struktogramm.dtd

EMPTY

Name der Variablen

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

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

Beschreibung der Verwendung

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

Variablentyp, derzeit ohne weitere Auswertung.

**statement** Eine einfache Anweisung.

Attributname	Modus	Definiert in
height	Optional	struktex.dtd

PCDATA

StylX-Attribut für die Höhe

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

**assignment** Eine einfache Zuweisung.

Attributname	Modus	Definiert in
lvalue	Pflicht	struktogramm.dtd

Variable die den berechneten Wert erhält

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

Berechnung

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Höhe

EMPTY

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

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

Attributname	Modus	Definiert in
condition	Pflicht	struktogramm.dtd

Bedingung der IF-Anweisung

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für den Winkel der linken Linie

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für den Winkel der rechten Linie

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für den Text wahr

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für den Text falsch

Elementname	Wiederholung
then	Genau einmal
else	Einmal oder garnicht

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

**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	Definiert in
condition	Pflicht	struktogramm.dtd

Elementname	Wiederholung
switch	Genau einmal oder viele

Bedingung für die SWITCH-Auswahl
----------------------------------

<code>&lt;case condition='WarumWas' &gt; &lt;/case&gt;</code>
---

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

St <sub>u</sub> kT <sub>E</sub> X-Attribut für den Winkel der Linie
---

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

St <sub>u</sub> kT <sub>E</sub> X-Attribut für die Höhe
---

**switch** Auswahlsschalter zur Mehrfachauswahl

Attributname	Modus	Definiert in
choice	Pflicht	struktogramm.dtd

Elementname	Wiederholung
%block	Null oder viele

Wert der Auswahl
------------------

<code>&lt;switch choice='IstKlasse' &gt; &lt;/switch&gt;</code>
---

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

St <sub>u</sub> kT <sub>E</sub> X-Attribut für die Position des Auswahltextes
---

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

Attributname	Modus	Definiert in
condition	Pflicht	struktogramm.dtd

Elementname	Wiederholung
%block	Null oder viele

Bedingung, solange dies gilt wird die Schleife durchlaufen
--

<code>&lt;while condition='SoLangeWahrIst' &gt; &lt;/while&gt;</code>
---

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

St <sub>u</sub> kT <sub>E</sub> X-Attribut für die Breite
---

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

Attributname	Modus	Definiert in
condition	Pflicht	struktogramm.dtd

Elementname	Wiederholung
%block	Null oder viele

Bedingung, solange dies gilt wird die Schleife durchlaufen
--

<code>&lt;dowhile condition='SoLangeWahrIst' &gt; &lt;/dowhile&gt;</code>
---

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

St <sub>u</sub> kT <sub>E</sub> X-Attribut für die Breite
---

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

Attributname	Modus	Definiert 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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Breite

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

Attributname	Modus	Definiert in	Elementname	Wiederholung
countervar	Pflicht	struktogramm.dtd	%block	Null oder viele

Zählervariable

start            Pflicht    struktogramm.dtd

```
<for countervar='Jünger' start='Urknall'
end='JüngsterTag' increment='Gebet'>
</for>
```

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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Breite

**loop** Endlosschleife. Abbruch durch return

Attributname	Modus	Definiert in	Elementname	Wiederholung
width	Optional	struktex.dtd	%block	Null oder viele

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Breite

```
<loop> </loop>
```

**exit** Ausstrungenweisung aus einer Loop

Attributname	Modus	Definiert in
value	Optional	struktogramm.dtd

Rückgabewert

height Optional struktex.dtd

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Höhe

EMPTY

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

**call** Aufruf einer Unterroutine

Attributname	Modus	Definiert in
subroutine	Pflicht	struktogramm.dtd

Name der Subroutine die aufgerufen 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

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Höhe

EMPTY

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

**return** Rücksprungenweisung aus dem Modul

Attributname	Modus	Definiert in
value	Optional	struktogramm.dtd

Rückgabewert

height Optional struktex.dtd

St<sub>u</sub>kT<sub>E</sub>X-Attribut für die Höhe

EMPTY

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

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

<i>Elementname</i>	
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 Konfigurationsdefinition

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

**nsd2ltx** Hauptelement der Konfigurationsdatei

<i>Elementname</i>	<i>Wiederholung</i>
application	Null oder einmal
modul	Null oder einmal
struktogramme	Null oder einmal

**application** Element zur Steuerung der Applikation

<i>Attributname</i>	<i>Modus</i>	<i>Definiert in</i>
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.

asyncon	Optional	application.dtd
---------	----------	-----------------

Die externen Programme werden asyncon aufrufen.

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

Ausgabemodus

<i>Elementname</i>	<i>Wiederholung</i>
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

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

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

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

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

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

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

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

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

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

**modul** Element zur Steuerung der L<sup>A</sup>T<sub>E</sub>X-Definitionen

*Attributname*   *Modus*   *Definiert in*

<i>Elementname</i>	<i>Wiederholung</i>
texheader	Null oder einmal
picheader	Null oder einmal

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

**texheader** Element für den Standardtext zu den L<sup>A</sup>T<sub>E</sub>X-Dokumentdateien

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

**picheader** Element für den Standardtext zu den L<sup>A</sup>T<sub>E</sub>X-Picture Dateien

<i>Elementname</i>	<i>Wiederholung</i>
PCDATA	Null oder einmal

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

**struktogramme** Element zur Steuerung der Struktogramme

<i>Attributname</i>	<i>Modus</i>	<i>Definiert in</i>	<i>Elementname</i>	<i>Wiederholung</i>
withFloat	Optional	application.dtd	description	Null oder einmal
			paramdecl	Null oder einmal
			localdecl	Null oder einmal
			returndecl	Null oder einmal
			ifyestext	Null oder einmal
			ifnotext	Null oder einmal

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

floatPosition	Optional	application.dtd
---------------	----------	-----------------

Welchen Positionsparameter 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.

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

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

<i>Elementname</i>	<i>Wiederholung</i>
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

```
<deklacration>
Variablen </deklacration>
```

**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

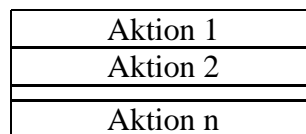
## Was sind Struktogramme

**Struktogramme**, auch *Nassi - Shneiderman - Diagramme* genannt, sind eine grafische Darstellungsmethode von Programmen im Sinne der strukturierten Programmierung.

Jeder Arbeitsschritt (hier *Aktion* genannt) eines Programmes bildet einen Strukturblock. Algorithmen stellen Kombinationen von Strukturblocken 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

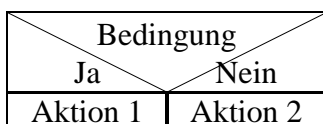


Struktogramm 2: Sequenz

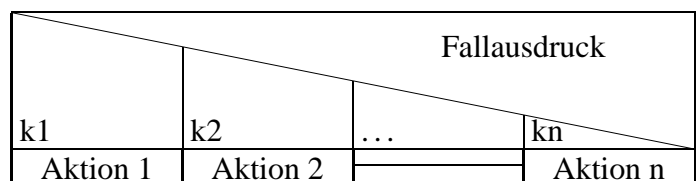
Aktionen in der angegebenen Reihenfolge ( $\hookrightarrow$  Struktogramm 2).

**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 ( $\leftrightarrow$  Struktogramm 3).

**Fallauswahl.** Eine Erweiterung stellt die Fallauswahl dar ( $\leftrightarrow$  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<sup>1</sup>.



Struktogramm 3: Vollständige Alternative



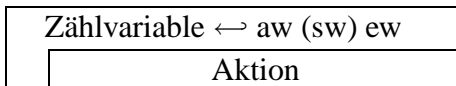
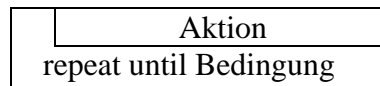
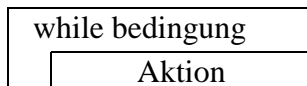
Struktogramm 4: Fallauswahl

<sup>1</sup>StylTeX 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 ( $\leftrightarrow$  Struktogramm 5). Bei *Abweisschleifen* (kopfgesteuerte Schleifen) wird vor *jedem Schleifendurchlauf* geprüft, ob die *Ausführbedingung* erfüllt ist. Der *Extremfall* ist das *nullmalige* Ausführen einer Schleife.

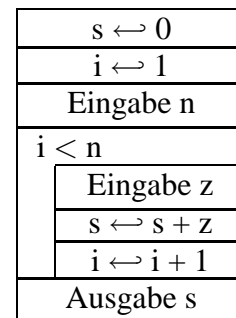
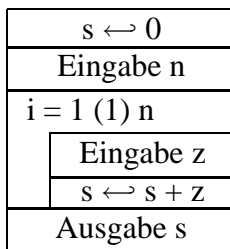
Bei *Wiederholschleifen* (fussgesteuerte Schleifen) wird *nach jedem Schleifendurchlauf* geprüft, ob die Abbruchbedingung erreicht ist oder die Schleife wiederholt zu durchlaufen ist. Die Schleife wird mindestens einmal durchlaufen.

Ein Beispiel zeigt Struktogramm 6.



aw, sw, ew bedeuten Anfangswert, Schrittweite, Endwert

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 Problems 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

Dieses Programm ist freie Software. Sie können es heraus- und/oder weitergeben. Modifizierung des Programms ist nur dann erlaubt, wenn Sie sich das Einverständnis des Lizenz-Inhabers einholen.

Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber **OHNE JEDE GEWÄHRLEISTUNG** - sogar ohne die implizite Gewährleistung der **MARKTREIFE** oder der **EIGNUNG FÜR EINEN BESTIMMTEN ZWECK**.

## Gewährleistung

Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

## 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 Konfigurationsdatei und im Wizard beeinflussbar.
- Weitere Ausgabemodi hinzugefügt: `psLATEX`, `dvipdfm`.

### 2.0 (2002-07-25)

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

### 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 Definition ergänzt.
- Wenn die Struktogramme in eine laufende Dokumentation ohne weitere Erklärung eingebaut werden, sind diese Struktogramme nicht unbedingt selbstredend, deswegen wurde die Dokument Typ Definition 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 Definition aufgenommen.

### 1.2.1 (2002-02-25)

- Bugfix: Im sogenannten Picture-Mode sind die Unterschriften der Fliessumgebung `strukto` überflüssig.
- 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 `Makefile` ergänzt.

### 1.1 (2002-02-21)

- Das Element `declaration` wurde aufgeteilt in `paramdecl` und `localdecl`.
- Die `StyTeX`-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 Definition 17  
Dokument Typ Definition 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

Makefile 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