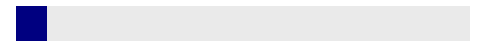

SDL

Eine Einführung
Tim Strazny

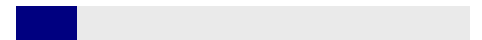
Gliederung

- Überblick – Was ist SDL?
- Entwicklung von SDL
- Anwendungen
- Ein Beispiel
- Überblick: Syntax
- Graphische und textuelle Repräsentation
- Tools
- Das Beispiel mit *Cinderella SDL*
- Verifikation
- Schlussbetrachtung



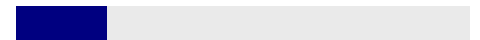
Überblick: Was ist SDL?

- „*Specification and Description Language*“
- Beschreibungssprache zur Modellierung von Verhalten und Struktur verteilter Systeme
- Graphische und textuelle Repräsentation (GR/PR)
- Eindeutige Semantik
- Parallel ablaufende erweiterte endliche Automaten
- Asynchrone Übertragung von Signalen



Entwicklung von SDL

- CCITT (International Telegraph and Telephone Consultative Committee)
- Entwicklung seit 1972, erste Version 1976
- Folgeversionen 1980, '84, '88, '92 '96, '99
- Objektorientiert seit SDL-92
- Entschlackung und große Erweiterung mit SDL-2000
- Ab 2004 Überarbeitung von SDL unter Einfluss von UML 2.0

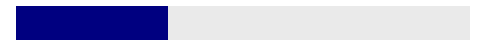


Anwendungen

- Ursprünglich für die Telekommunikation gedacht, aber auch einsetzbar für
 - Realzeitsysteme
 - Interaktive Systeme
 - Verteilte Systeme
- Gute Abstraktionsmöglichkeiten
- Kann als Implementierungssprache eingesetzt werden

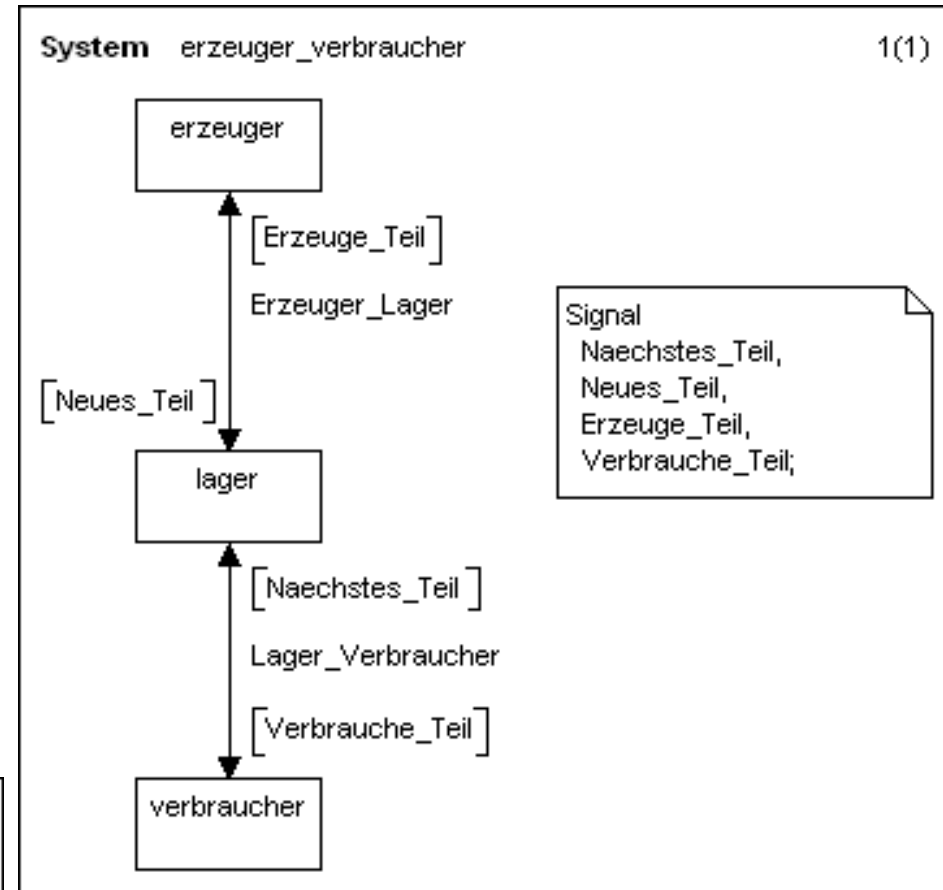
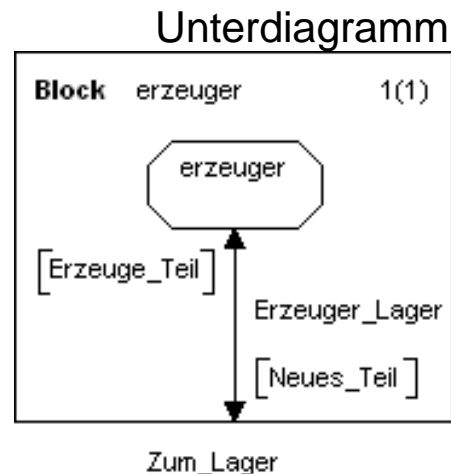
Beispiel: Erzeuger - Verbraucher

- Erzeuger, Lager und Verbraucher
- Die Lageristen fordern Teile beim Erzeuger an, bis das Lager voll ist
- Wenn Teile im Lager sind, werden sie auf Anfrage an den Verbraucher gegeben
- Der Erzeuger erzeugt auf Anfrage neue Teile und gibt sie an das Lager
- Der Verbraucher fordert beim Lager Teile an und konsumiert sie



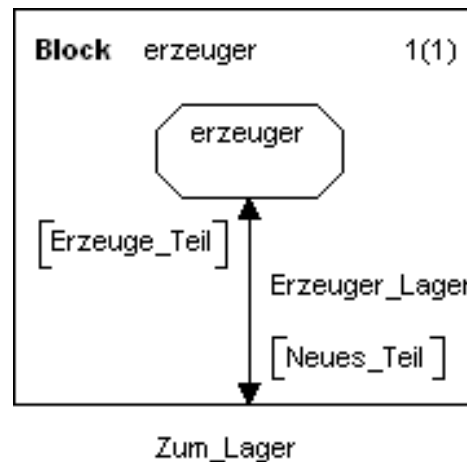
Konzept der Blöcke

- Physikalische Einheiten
- Strukturieren
- Verbunden über Kanäle
- Enthalten
 - Blöcke oder
 - Prozesse

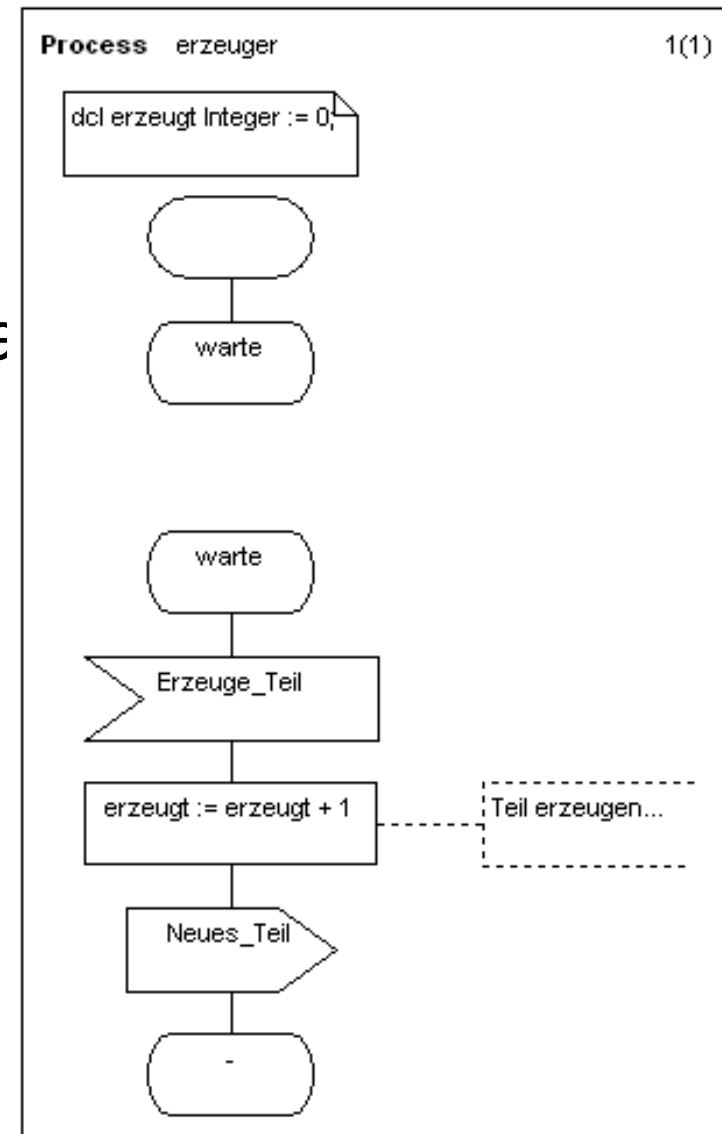


Konzept der Prozesse

- Parallel und gleichberechtigt
- Unabhängig voneinander
- Erzeugen / konsumieren *Signale*
- Festgelegte Signalwege
- Erweiterte endliche Automaten
- Viele Konstrukte zur Steuerung

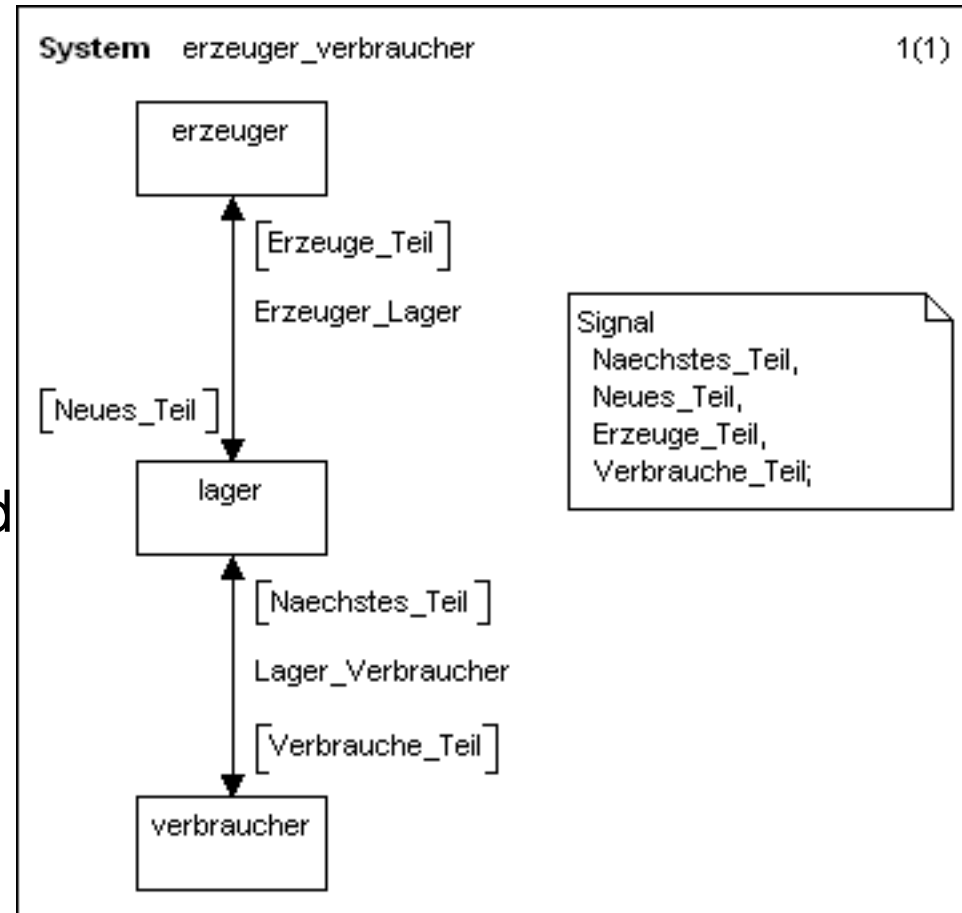


Unterdiagramm



Konzept der Signale

- Signale werden erzeugt / konsumiert von
 - Prozessen
 - Umwelt (ENV)
- Die festgelegten Routen sind
 - verlust- und störungsfrei
 - unbegrenzte FIFO-Puffer
- *signal routes / non-delaying channels* zwischen Prozessen, nicht zeitverzögert
- *channels* zwischen Blöcken, evtl. zeitverzögert

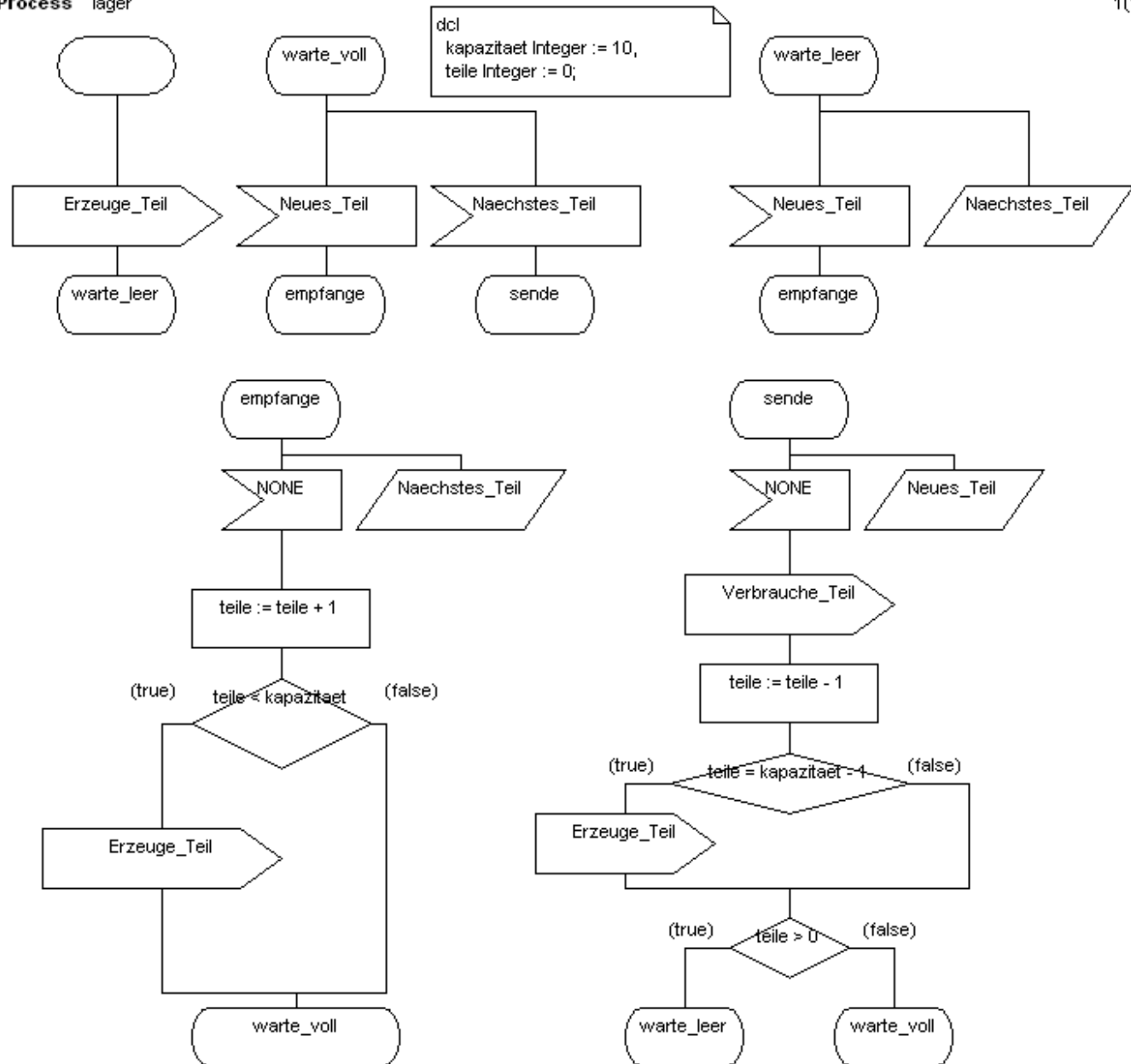


Erweiterte Automaten

- Start
- State
- Nextstate
- Input
- Output
- Save
- Task
- Decision

Process lager

1(1)

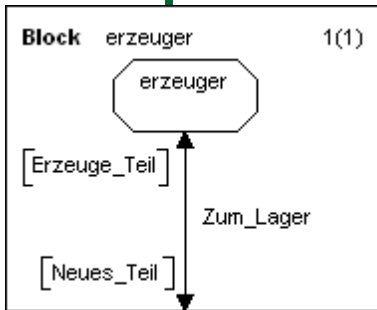


Erweiterte Automaten (Forts.)

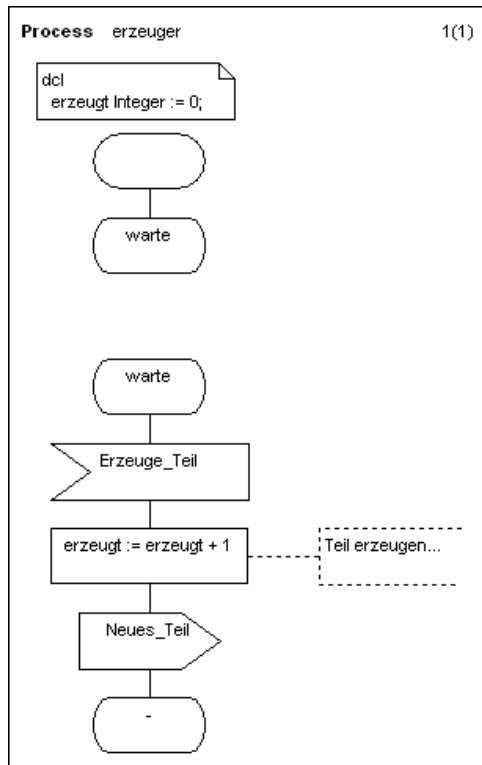
- Transition Start (direkt nach Zustand)
 - Input (+ Enabling Condition)
 - Spontaneous Transition
 - Priority Input (+ Enabling Condition)
 - Continuous Signal
 - Save
- Transition Body
 - Output
 - Task
 - Decision
 - Procedure Call
 - Process Creation
- Transition Terminator
 - Nextstate
 - Stop
 - Join
- Implizite Variablen eines Prozesses
 - self
 - offspring
 - parent
 - sender
- Einfache Datentypen
 - Boolean
 - Integer, Natural, Real
 - Character, Charstring
 - PId
 - Duration, Time
 - ...



Graphical / Phrase Representation

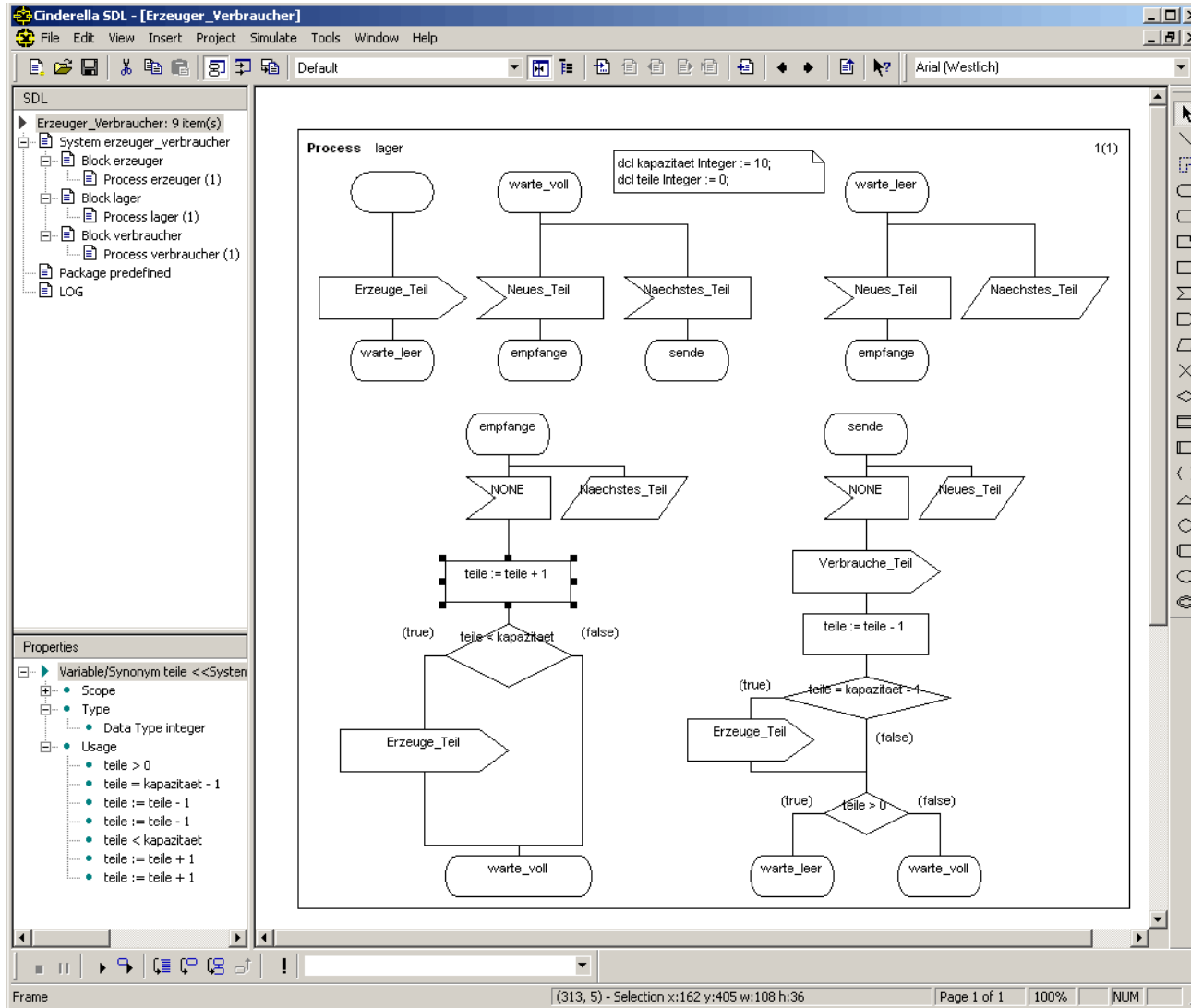


Erzeuger_Lager



```
BLOCK erzeuger ;  
  SIGNALROUTE zum_lager  
    FROM erzeuger TO ENV WITH Neues_Teil ;  
    FROM ENV TO erzeuger WITH Erzeuge_Teil ;  
  PROCESS erzeuger ;  
    dcl erzeugt Integer := 0 ;  
  START ;  
    NEXTSTATE warte ;  
  STATE warte ;  
    INPUT Erzeuge_Teil ;  
    TASK erzeugt := erzeugt + 1  
      COMMENT 'Teil erzeugen...' ;  
    OUTPUT Neues_Teil ;  
    NEXTSTATE - ;  
  ENDSTATE ;  
ENDPROCESS ;  
CONNECT erzeuger_lager AND zum_lager ;  
ENDBLOCK ;
```


Beispiel in *Cinderella SDL*



Verifikation

- Wegen *state explosion* SDL einschränken
- Übersetzung in semantisch äquivalente Petri-Netze
- SDL → M-Netze → LL-Netze → Verifikation

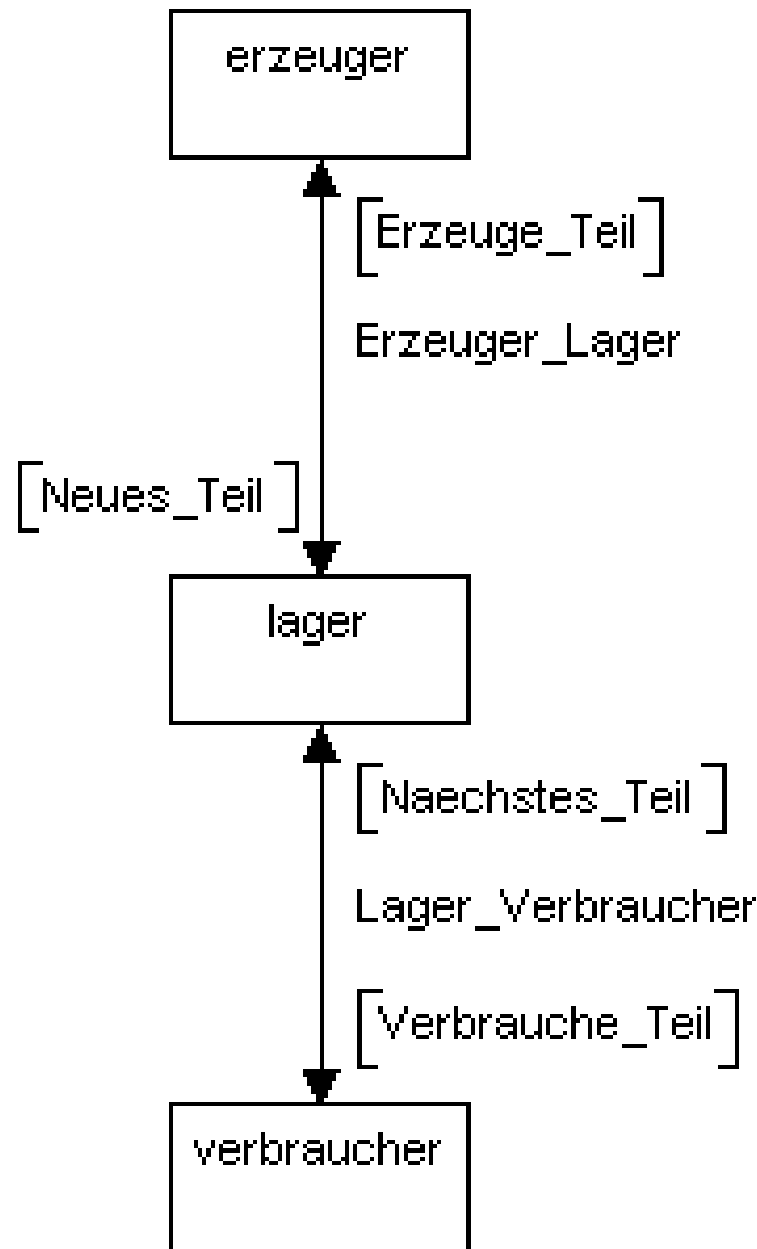
- Anderer Ansatz *Tau SDL Suite*
 - Statische Analyse
 - „an automatic exhaustive exploration of the SDL specification, and all included C code“
 - <http://www.telelogic.com/products/tau/sdl/index.cfm>

Schlussbetrachtung

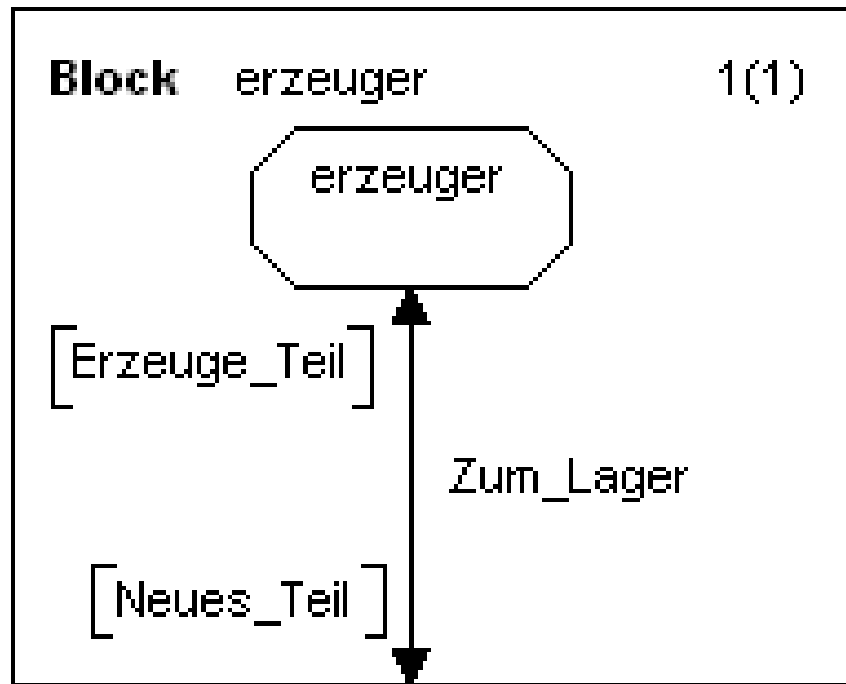
- Gut zum Modellieren komplexer verteilter Systeme
 - Hierarchische Schachtelung von Blöcken
 - „Dünne“ Schnittstellen zwischen Blöcken
 - Möglichkeit der Simulation / Validation / Verifikation
-
- Diagramme werden leicht unübersichtlich
 - Komplexe Datentypen machen SDL-Systeme „schwerer“ zu validieren / verifizieren
-
- Mächtige (kommerzielle) Tools vorhanden
 - Freie Tools können noch nicht mithalten

Quellen

- Jens Grabowski, Ekkart Rudolph, Michael Schmitt, „Die Spezifikationsprachen MSC und SDL – Teil 2: Specification and Description Language (SDL)“ in Automatisierungstechnik 50 (2002) 2, 2002, Oldenbourg Verlag
- Laurent Doldi, „Validation of Communications Systems with SDL“, 2003, Wiley
- Stefan Schwoon, „Übersetzung von SDL-Spezifikationen in Petri-Netze“, Diplomarbeit 1998, Universität Hildesheim
- <http://www.sdl-forum.org/sdl88tutorial/>
- <http://www.sdl-forum.org/SDL/>
- http://www.sintef.no/time/elb40/html/elb/sdl/sdl_t01.htm
- <http://www.telelogic.com/products/tau/sdl/index.cfm>
- <http://www.cinderella.dk/>

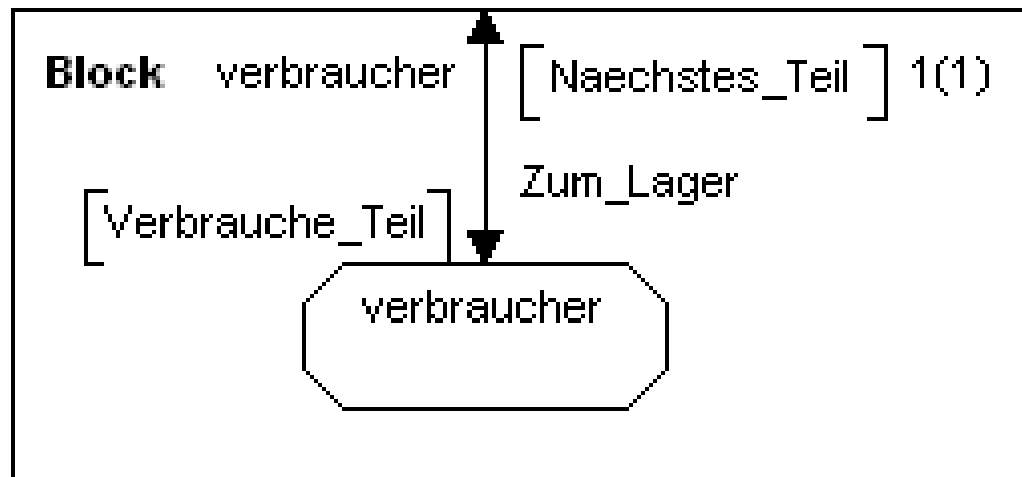


Signal
Naechstes_Teil,
Neues_Teil,
Erzeuge_Teil,
Verbrauche_Teil;

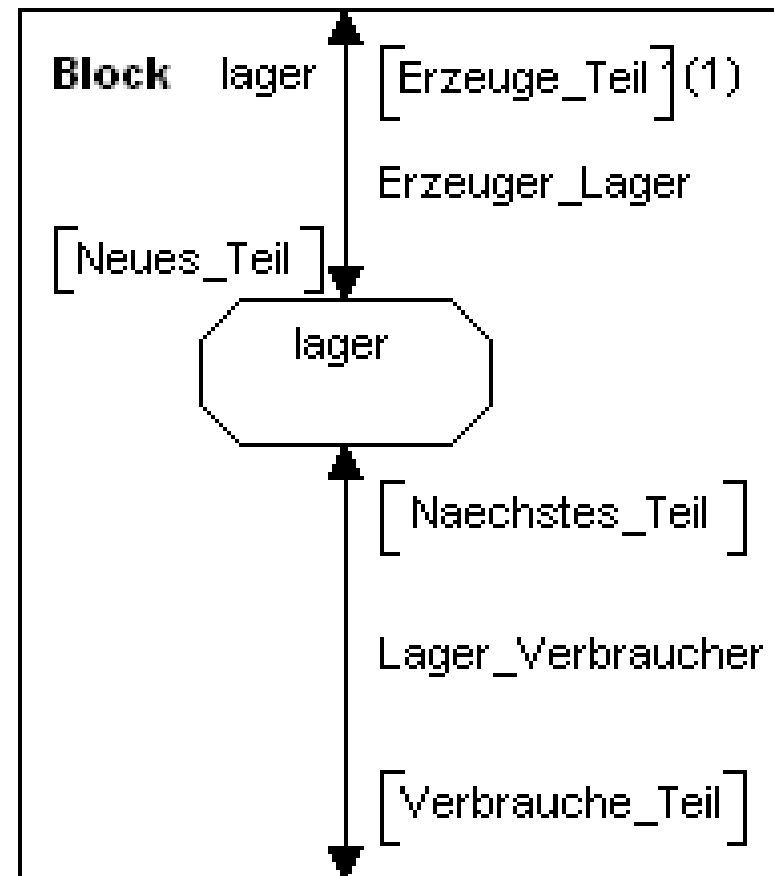


Erzeuger_Lager

Lager_Verbraucher



Zum_Erzeuger

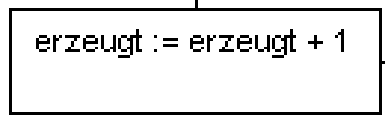
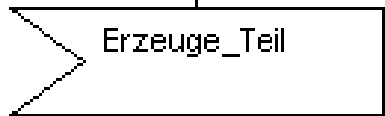
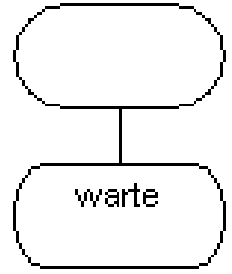


Zum_Verbraucher

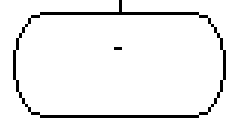
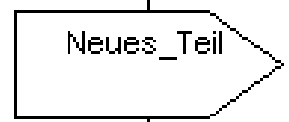
Process erzeuger

1(1)

dcl
erzeugt Integer := 0;



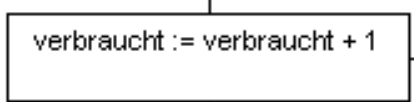
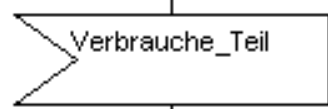
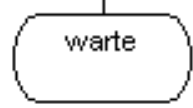
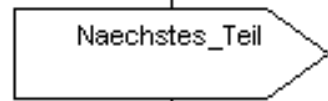
Teil erzeugen...



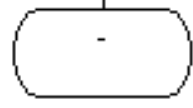
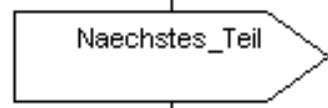
Process verbraucher

1(1)

dcl
verbraucht Integer := 0;

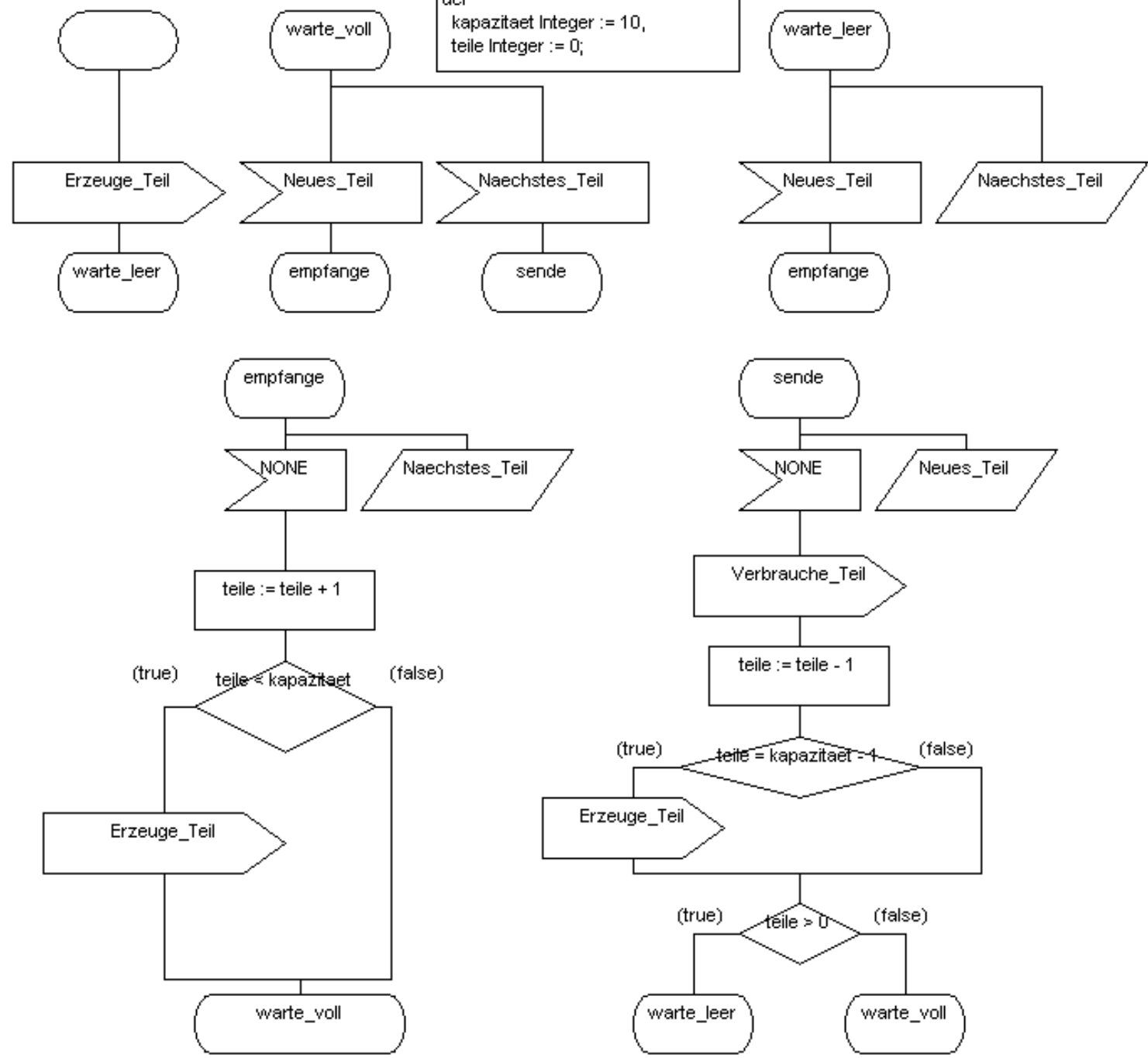


Teil konsumieren...



```

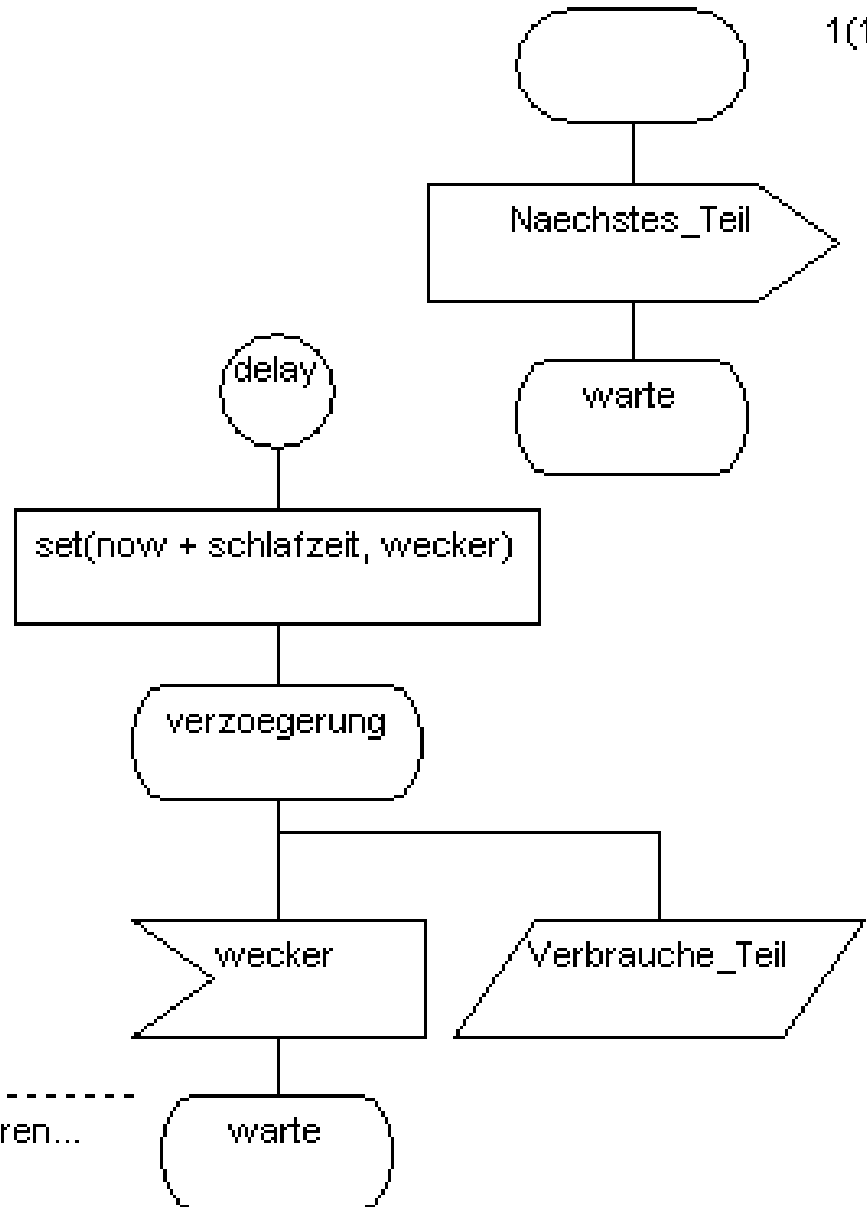
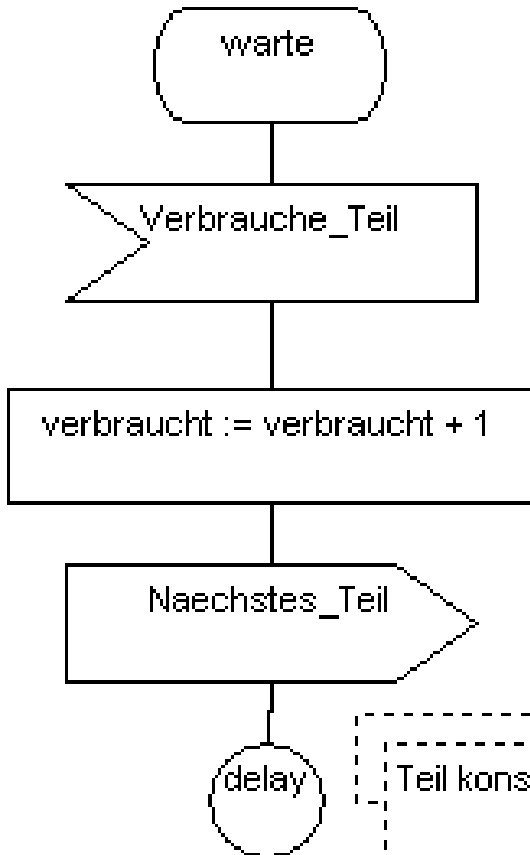
dcl
kapazitaet Integer := 10,
teile Integer := 0;
  
```



Process verbraucher

1(1)

```
dcl
verbraucht Integer := 0,
schlafzeit Duration := 0.5;
timer
wecker;
```



- Timer
- Join
- Nextstate / State
- Duration (einheitenlos)