Textuell-grafischer Editor für digitale Steuerungssysteme

P. Lüders
R. Ernst
Institut für
Datenverarbeitungsanlagen
Technische Universität Braunschweig

IDA | Workshop für Computergrafik und automatisierte Layoutsynthese | Bild 1.1

1 Einleitung

2 Spezifikation von Steuerungssystemen

3 Vorstellung des Entwurfswerkzeugs für endliche Automaten

3.1 Regeln zur Layoutdarstellung

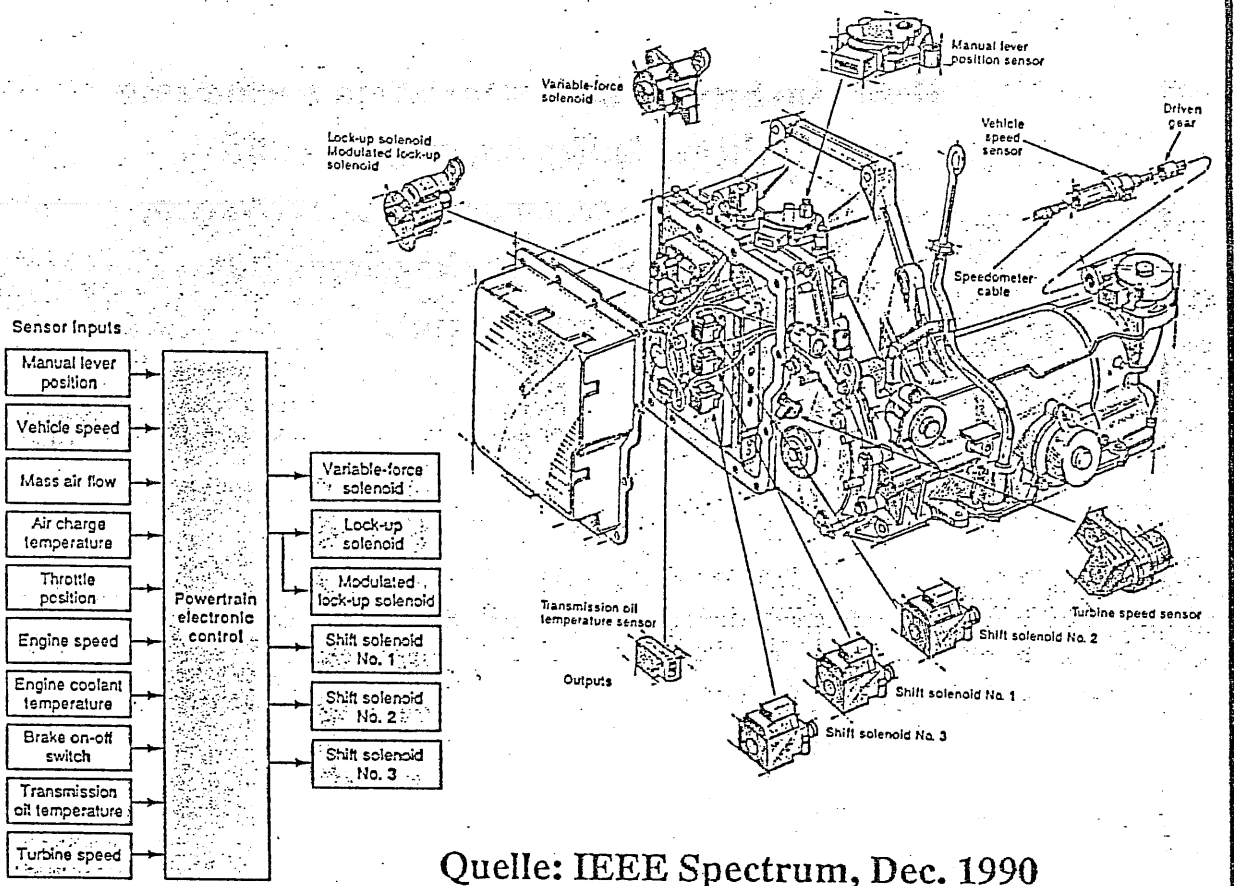
3.2 Aufbau des Informationsfilters

3.3 Algorithmen zum Graphlayout

4 Zusammenfassung und Ausblick

Def.: Ein Steuerungssystem ist ein hochgradig parallel ablaufendes, aus Soft- und Hardwareanteilen bestehendes System, das stark mit seiner Umgebung interagiert (reaktives System).

Ziel: Ein Werkzeug zur textuell-grafischen Spezifikation komplexer Steuerungssysteme, das den Entwickler von zeitaufwendigen grafischen Edierarbeiten befreit und ihn bei der Validierung des spezifizierten Systems unterstützt.



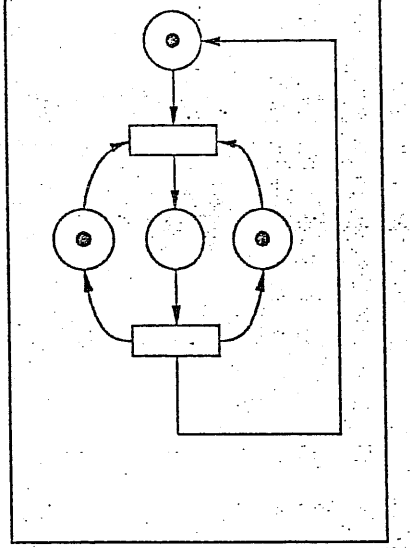
Quelle: IEEE Spectrum, Dec. 1990

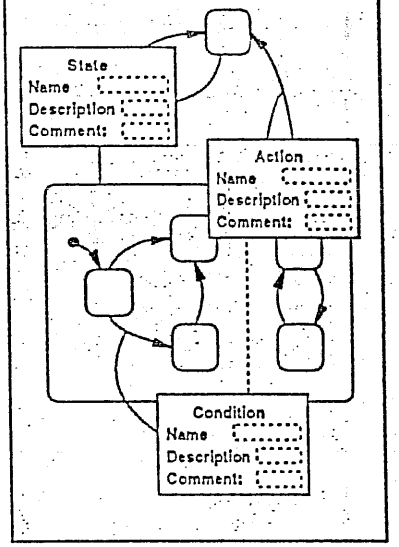
IDA	Beispiel für ein Steuerungssystem	Bild 1.4

Spezifikation von Steuerungssystemen:

- Schematic Entry
- (erweiterte) endliche Automaten
- (farbige) Petri-Netze
- Echtzeit- oder Hardwarebeschreibungs-sprachen (z.B. PEARL / VHDL)

```
      .
      .
      .
proc: process
begin
  wait on a = '0';
  ctrl <= '1' after 10 ns;
  sig  <= '0' after 7 ns;
  if input = '1' then
    b <= '1' after 2 ns;
  else
    b <= '0' after 4 ns;
  endif
end
end process;
      .
      .
      .
```





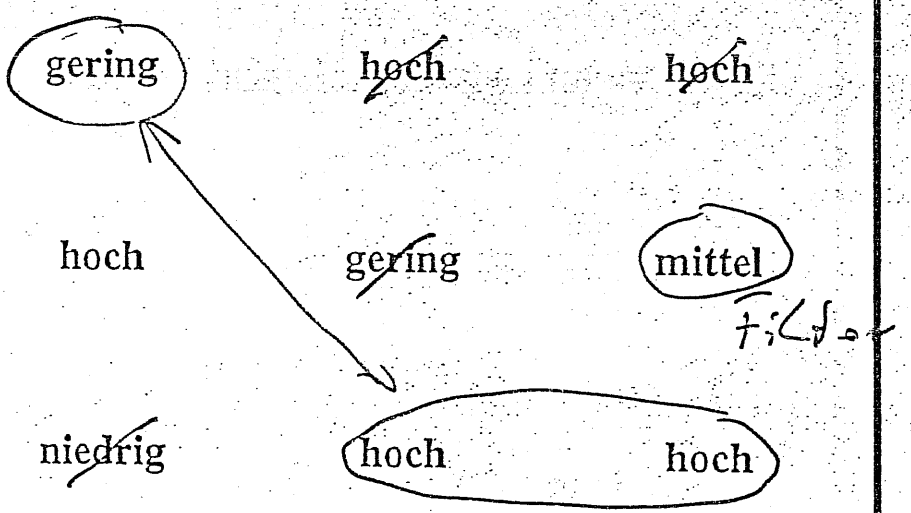
textuell:	grafisch:	grafisch-textuell:
VHDL	Petri-Netze	State-Charts farbige Petri-Netze

IDA

Gegenüberstellung von textueller, grafischer und gemischt grafisch-textueller Spezifikation

Bild 2.2

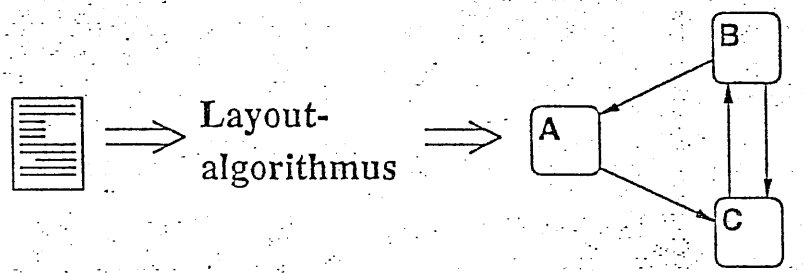
	textuell (VHDL)	grafisch (Petri-Netz)	textuell-grafisch (State-Charts)
Aufwand bei Änderung	gering	hoch	hoch
Informations-dichte (Daten)	hoch	gering	mittel
Erkennbarkeit der Struktur	niedrig	hoch	hoch



IDA | Eigenschaften der Spezifikationen bei Verwendung rein textueller bzw. grafischer Editoren | Bild 2.3

Ziel: gemischt grafisch-textuelle Spezifikation

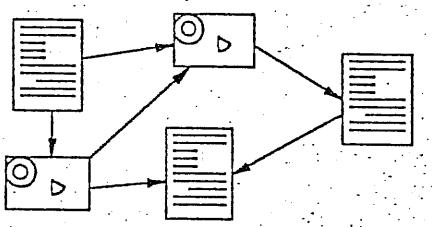
a) Verringerung des Zeitaufwands bei Ediervorgängen
=> automatisches Layout



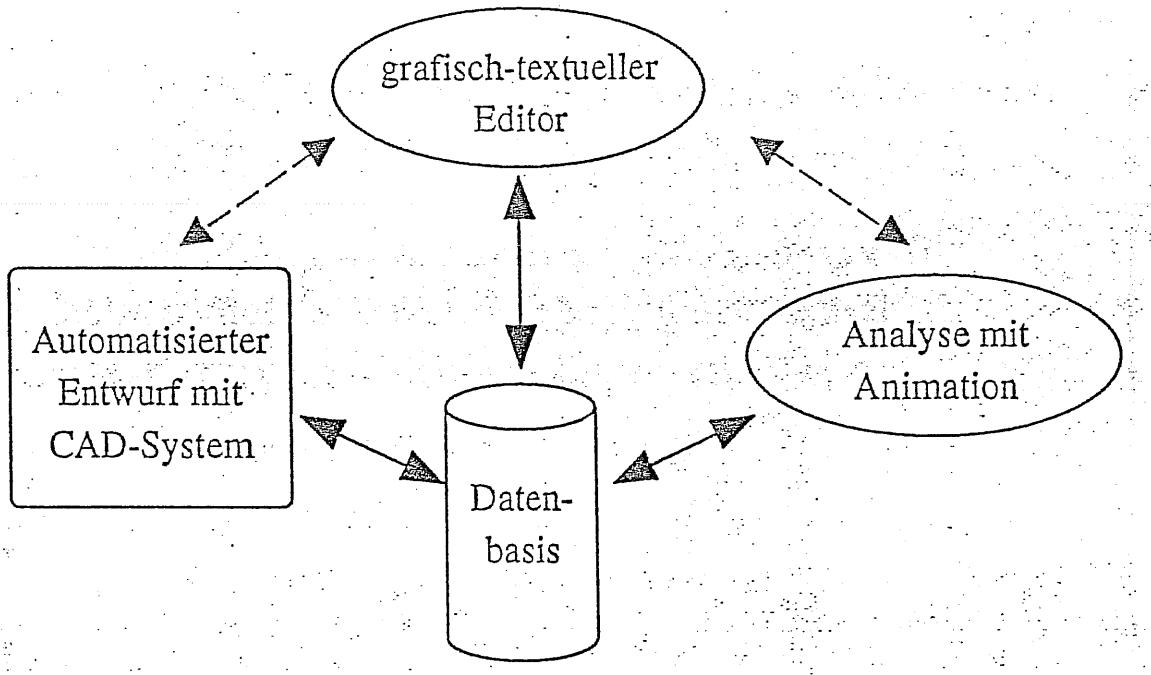
Probleme:
- unterschiedliche Sprachcharakteristik
- hohe Zahl von graphischen Elementen
- Rechenzeit

Ziel: gemischt grafisch-textuelle Spezifikation

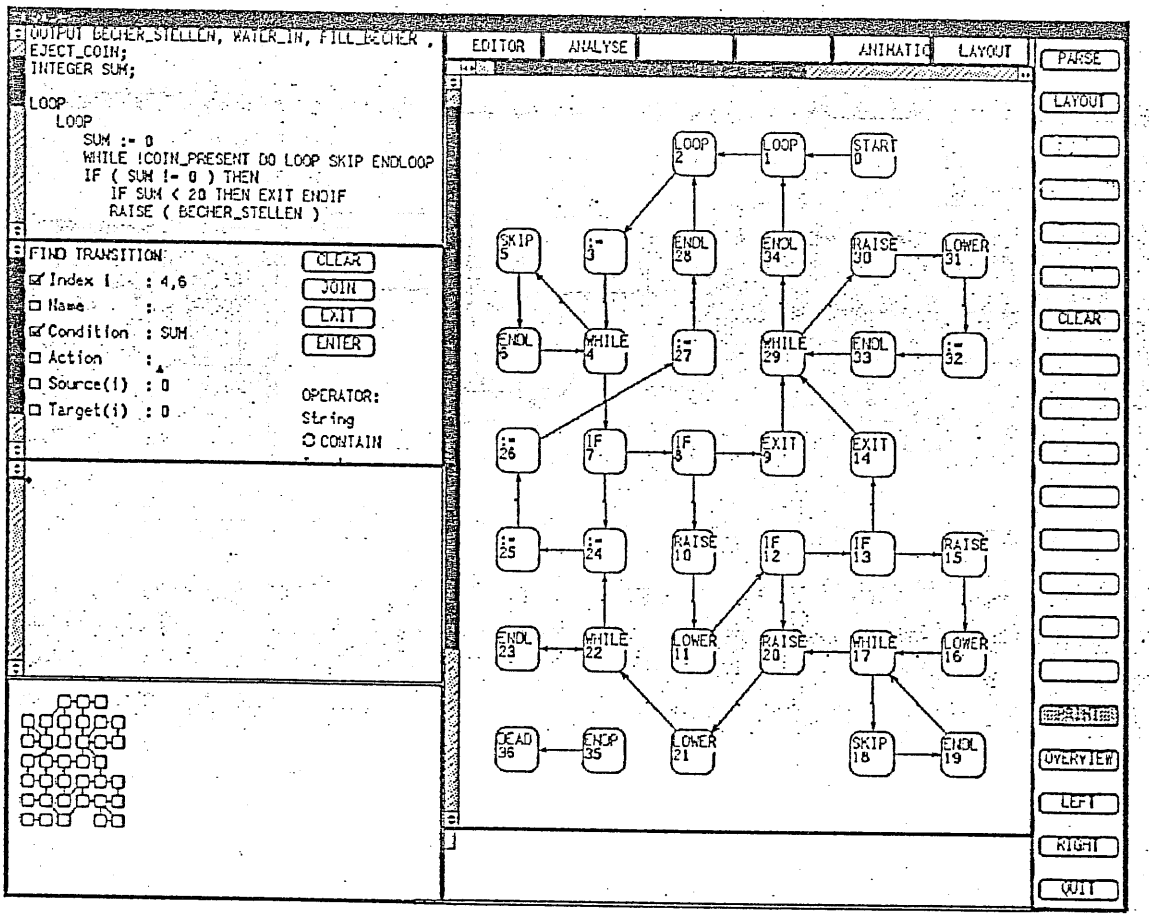
b) Verbesserung der Informationsdichte durch
Reduzierung der darzustellenden Information
=> Informationsfilter (Stichwort "Hypertext")



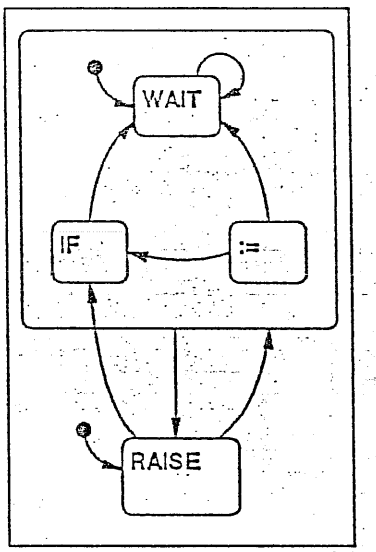
Probleme: - interaktive Steuerung des Filters
- Verständlichkeit der Darstellung



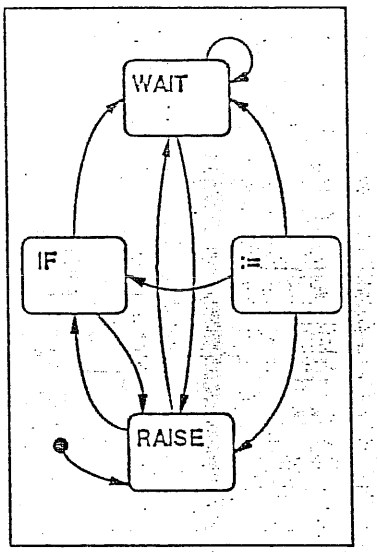
IDA | Struktur des Spezifikationssystems | Bild 3.1



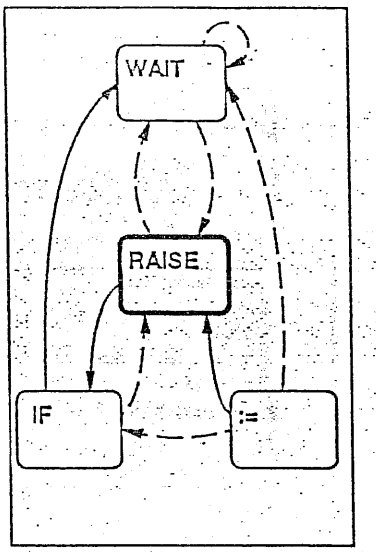
IDA Benutzeroberfläche des Entwurfswerkzeugs für endliche Automaten **Bild 3.2**



allg.: Hierarchie
Knotentyp anz.
Kanten voll anz:
spez.: -

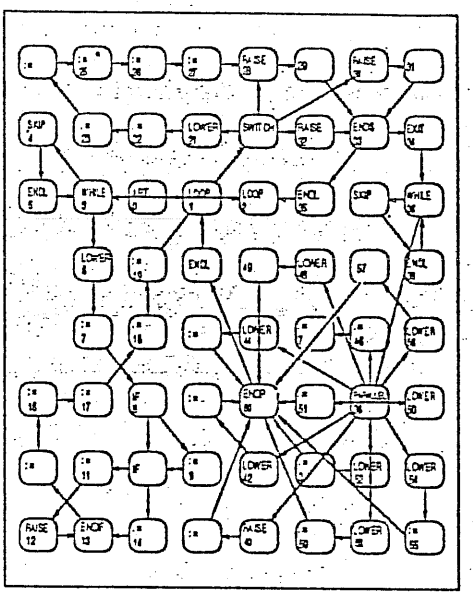


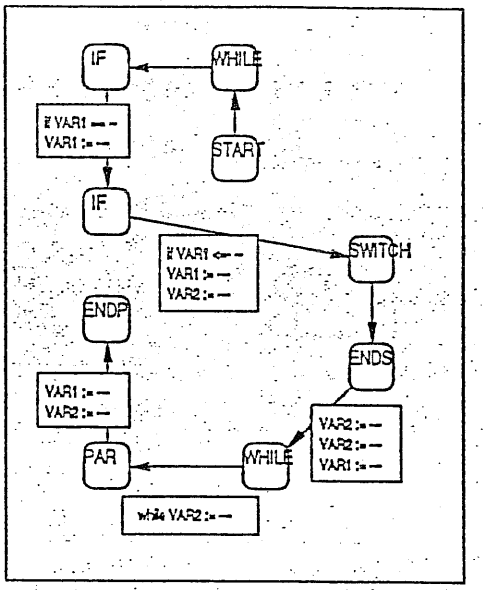
allg.: keine Hierarchie
Knotentyp anz.
Kanten voll anz.
spez.: -



allg.: keine Hierarchie
Knotentyp anz.
Kanten voll anz.
spez.: Kanten I/O:
gestrichelt anz.
aktueller Knoten:
mittig anz.
dick anz.

IDA | Sprache zur Angabe von Regeln für die Graphdarstellung | Bild 3.3

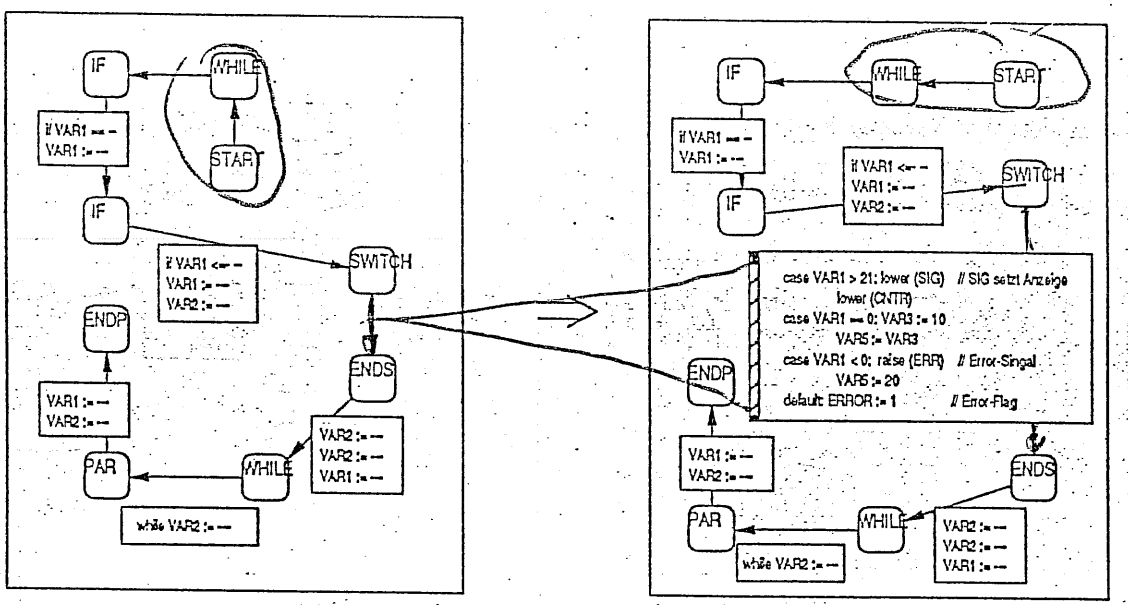




Informationsfilter:

Zeige alle Knoten vom Typ
IF,WHILE, PAR, START, SWITCH
und stelle Bedingungen und Zuweisungen der
Variablen VAR1, VAR2 dar.

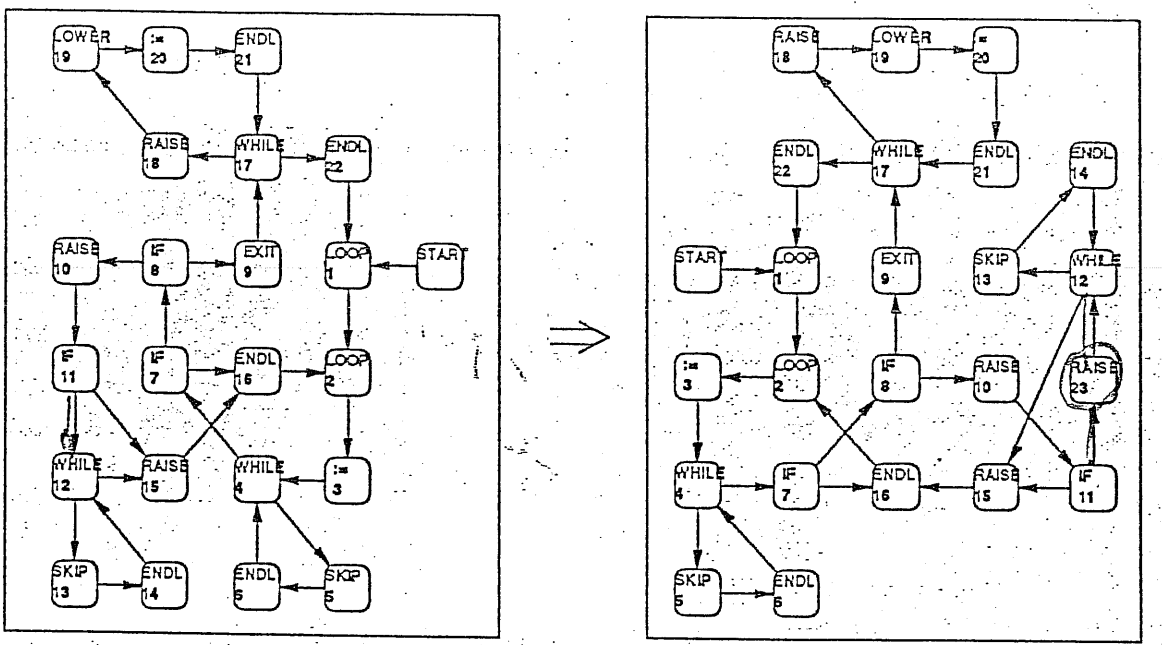
IDA	Verwendung eines Informationsfilters: Beispiel	Bild 3.4



<u>Geänderter Informationsfilter:</u>

Es soll weiterhin die gesamte textuelle Information innerhalb der SWITCH-Anweisung dargestellt werden.

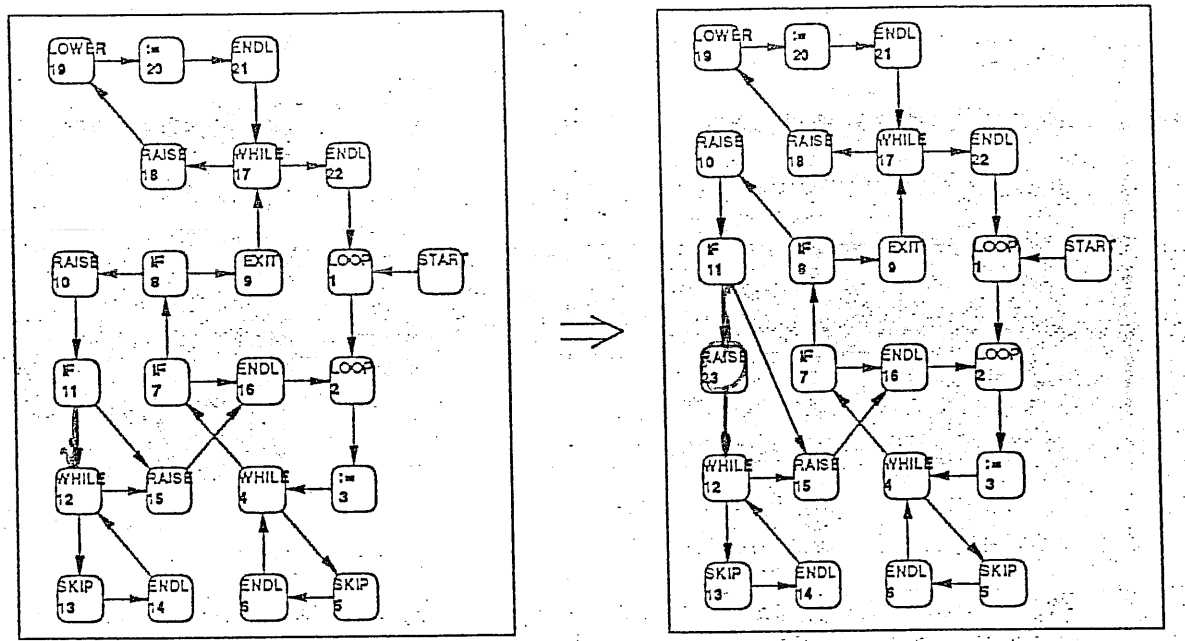
IDA	Neuplazierung durch Änderung des Informationsfilters	Bild 3.5



Beim Edieren oder bei Änderungen des Informationsfilters wird der Graph modifiziert.

Die Generierung eines Graphlayouts erfolgt i.a. ohne Rücksicht auf die bestehende Graphstruktur.

IDA	Inkrementelle Änderung mit Neuplazierung	Bild 3.6

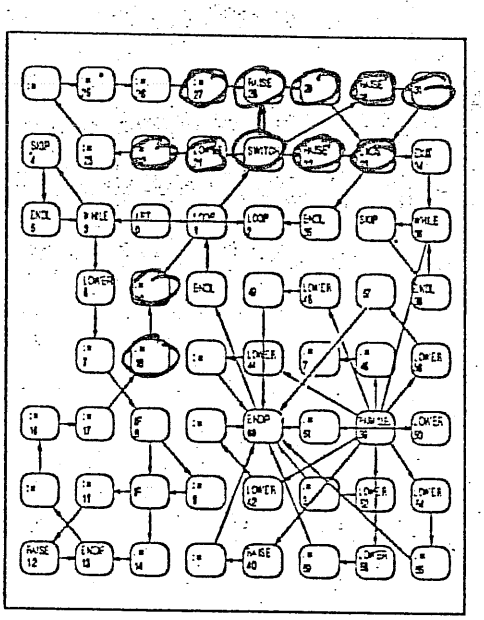


Gefordert wird Ähnlichkeit der Graphstruktur:

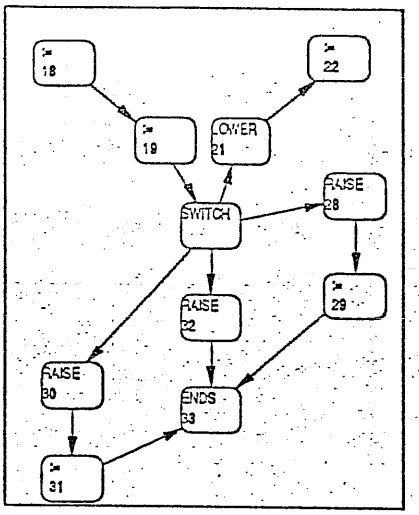
=> Topologische Konsistenz

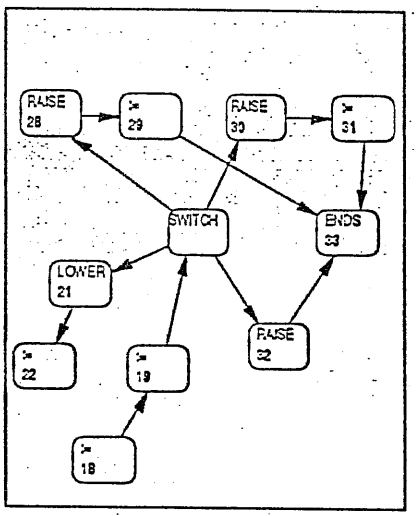
Die Generierung des Graphlayouts erfolgt mit Rücksicht auf die bestehende Graphstruktur.

IDA	Inkrementelle Änderung mit Neuplazierung mit topologischer Konsistenz	Bild 3.7



unabhängige
Neuplazierung





Neuplazierung
unter Beachtung
der topologischen
Konsistenz

IDA

Topologische Konsistenz bei Verwendung eines Informationsfilters

Bild 3.8

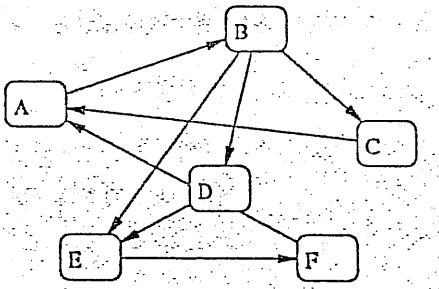
Anforderung an den Layoutalgorithmus für den Einsatz in einem textuell-grafischen Editor:

- Rechenzeit im Sekundenbereich (interaktives Entwurfswerkzeug)
- Plazierung unterschiedlich großer Objekte (Größe variabel)
- "übersichtliches", "ästhetisches" Graphlayout
- topologische Konsistenz
- Geringe Anzahl graphischer Elemente (< 50)

Lösungsansatz durch Untersuchung und Entwicklung geeigneter Kostenfunktionen zur Steuerung des Layoutalgorithmus (Flexibilität).

Vorschlag für statische Kostenfunktionen (Einzelbild):

- Kantenlänge minimieren
- Kantenschnittpunkte minimieren
- Gleichmäßige Dichte erreichen
- Verbundene Knoten gegenseitig sichtbar



Beispielgraph

Vorschlag für dynamische Kostenfunktionen (Bildsequenz):

- Korrelation
- Strukturelle Ähnlichkeit

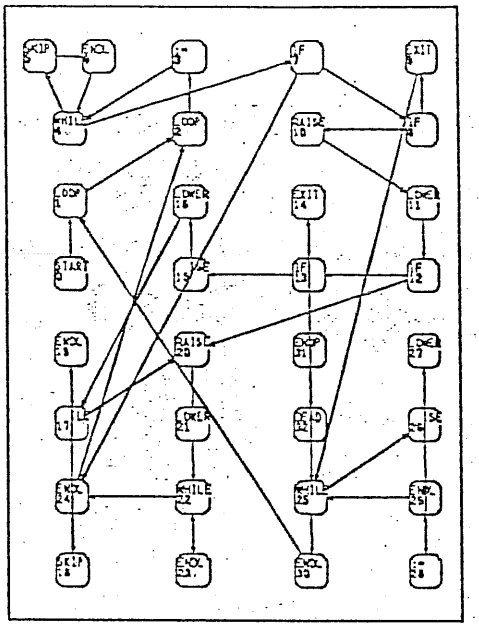
Implementierte Layoutalgorithmen:
(SUN Sparc1, ´C++´)

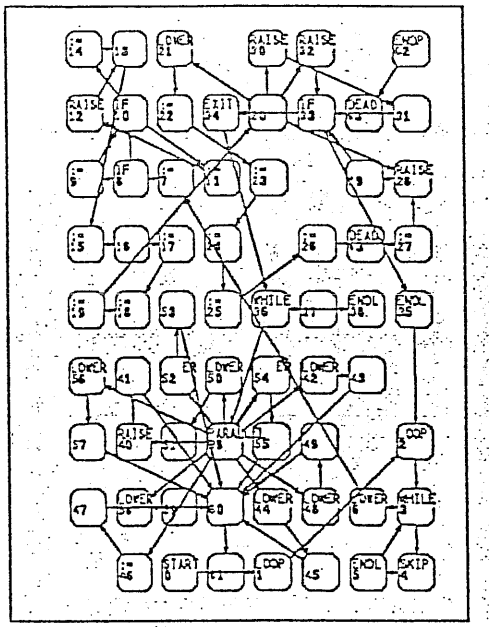
Min-Cut
Simulated Annealing
Genetischer Algorithmus

Ziel ist Aufspaltung des Layoutalgorithmus:

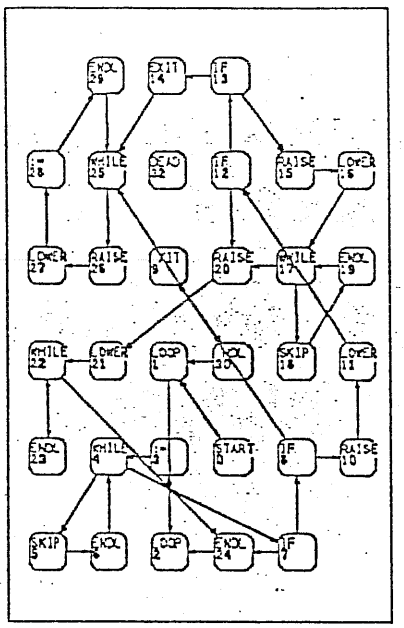
a) Globales Layout zur Vorplazierung, gesteuert durch dynamische Kostenfunktion.

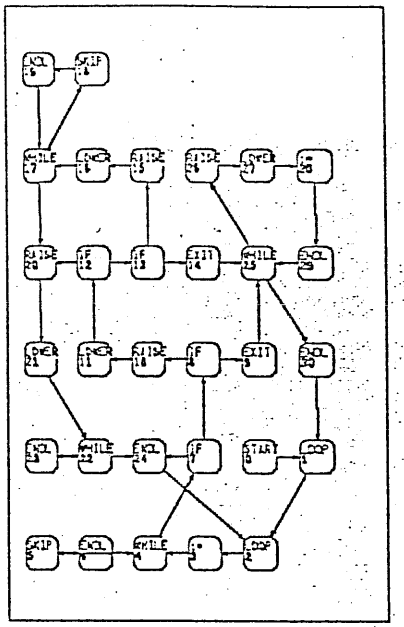
b) Lokales Layout zur Endplazierung, gesteuert durch statische Kostenfunktion.

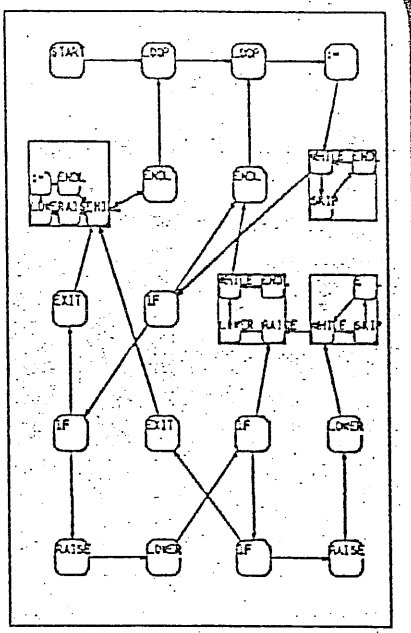




Knoten/Kanten	Bedingungen	Zeit [s] / SUN Sparc1, C++
33/44	Neuplazierung	3.06
64/80	Neuplazierung	28.8







Knoten/Kanten	Bedingungen	Zeit [s] / SUN Sparc1, C++
33/44	Neuplazierung (10 % - Schwelle)	3.23
33/44	Neuplazierung (0,1 % - Schwelle)	11.7
37/44	Neuplazierung (0,1 % -Schwelle)	22.8

- Editor für textuell-grafische Spezifikation von reaktiven Systemen
- Reduzierung der dargestellten Information auf das momentan Notwendige (Informationsfilter)
- Automatisches Graphlayout
- Zweidimensionale Kostenfunktion
- Generieren des Graphlayouts durch Regeln gesteuert
- Algorithmen: Min-Cut, Simulated Annealing, Genetische Algorithmen

- textuelle Spezifikation mit VHDL
- Weiterere Layoutalgorithmen
- Geeignete Algorithmen zum Routen der Kanten
- dynamische Kostenfunktion (topologogische Konsistenz)