

# Generierung von Diagrammeditoren mit Benutzerassistenz und Sketching-Schnittstelle

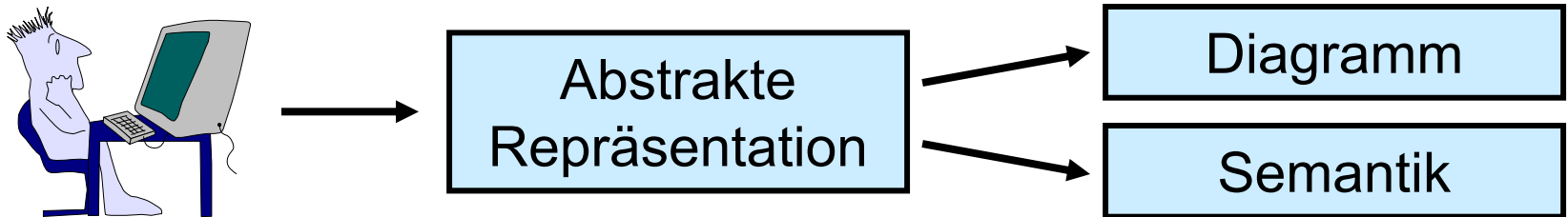
**Mark Minas**

Universität der Bundeswehr München

**Mark.Minas@unibw.de**

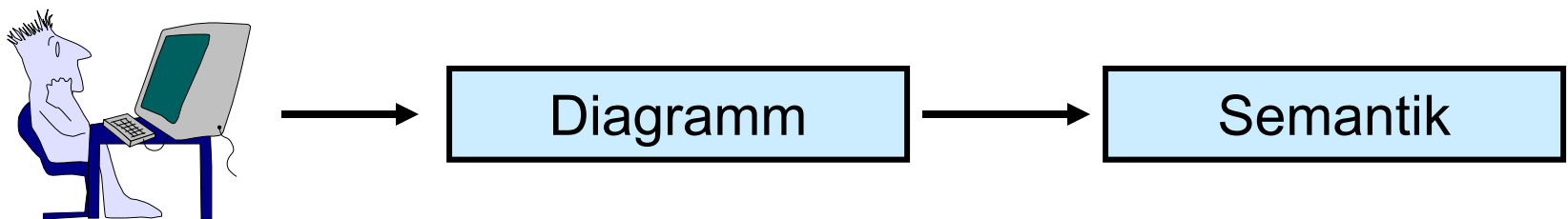
- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

- **Strukturiertes Editieren**



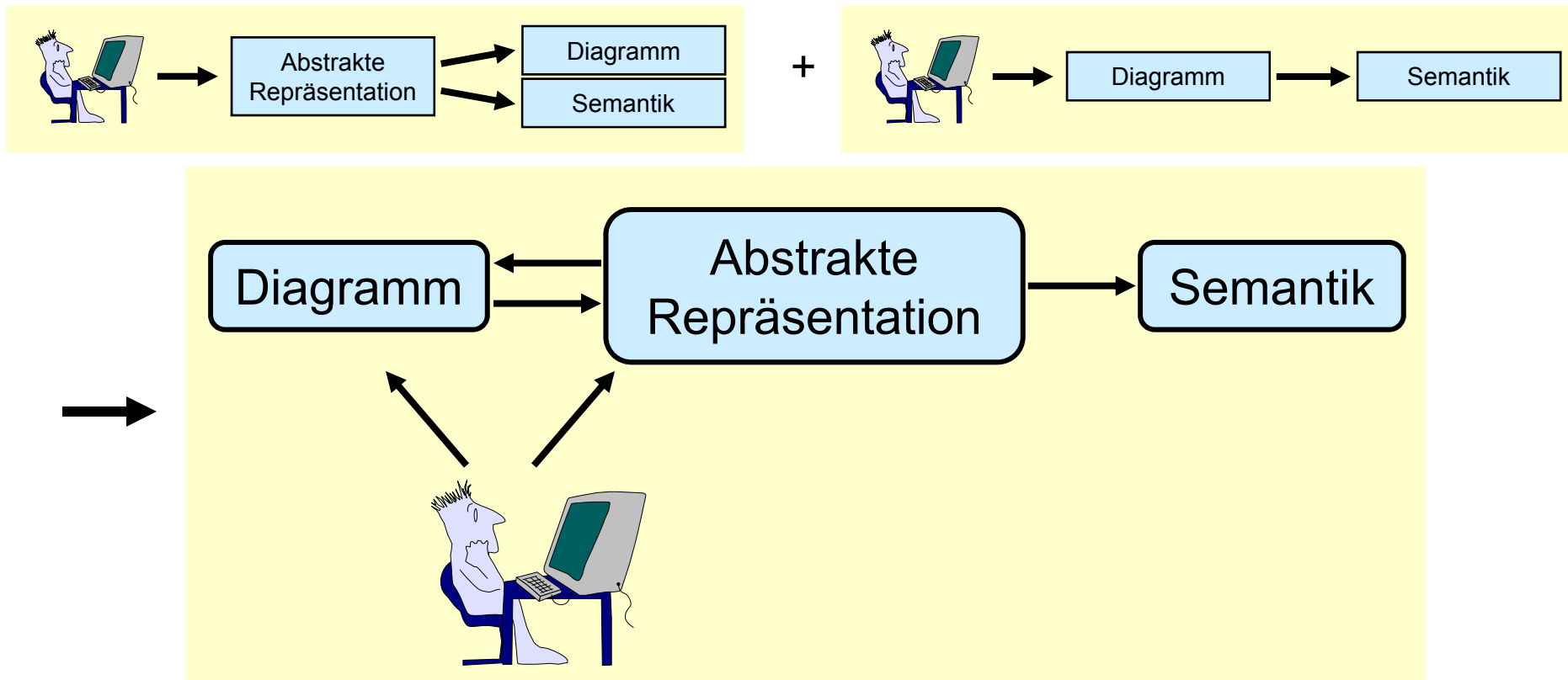
- + Relativ einfach zu realisieren
- + Stellt dem Nutzer eine Menge von Operationen zur Verfügung
- Schränkt den Benutzer ein

- **Freihändiges Editieren**



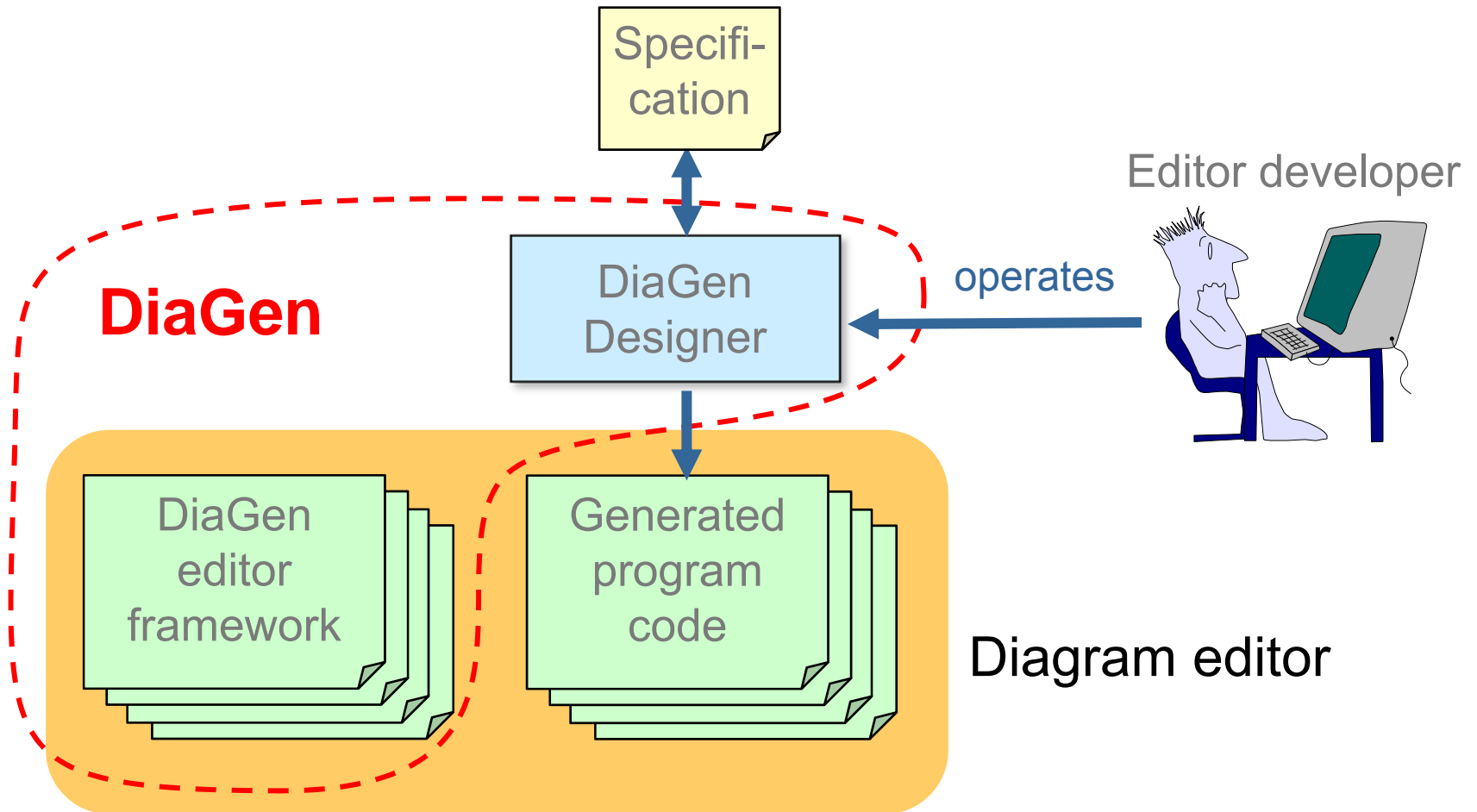
- Erfordert aufwändige Diagrammanalyse
- + Lässt dem Nutzer größtmögliche Freiheit
- Gibt dem Nutzer keine „Anleitung“

- Kombination von **strukturiertem** und **freihändigem** Editieren
  - + Stellt dem Nutzer eine Menge von Operationen zur Verfügung
  - + Lässt dem Nutzer größtmögliche Freiheit
  - Erfordert aufwändige Diagrammanalyse

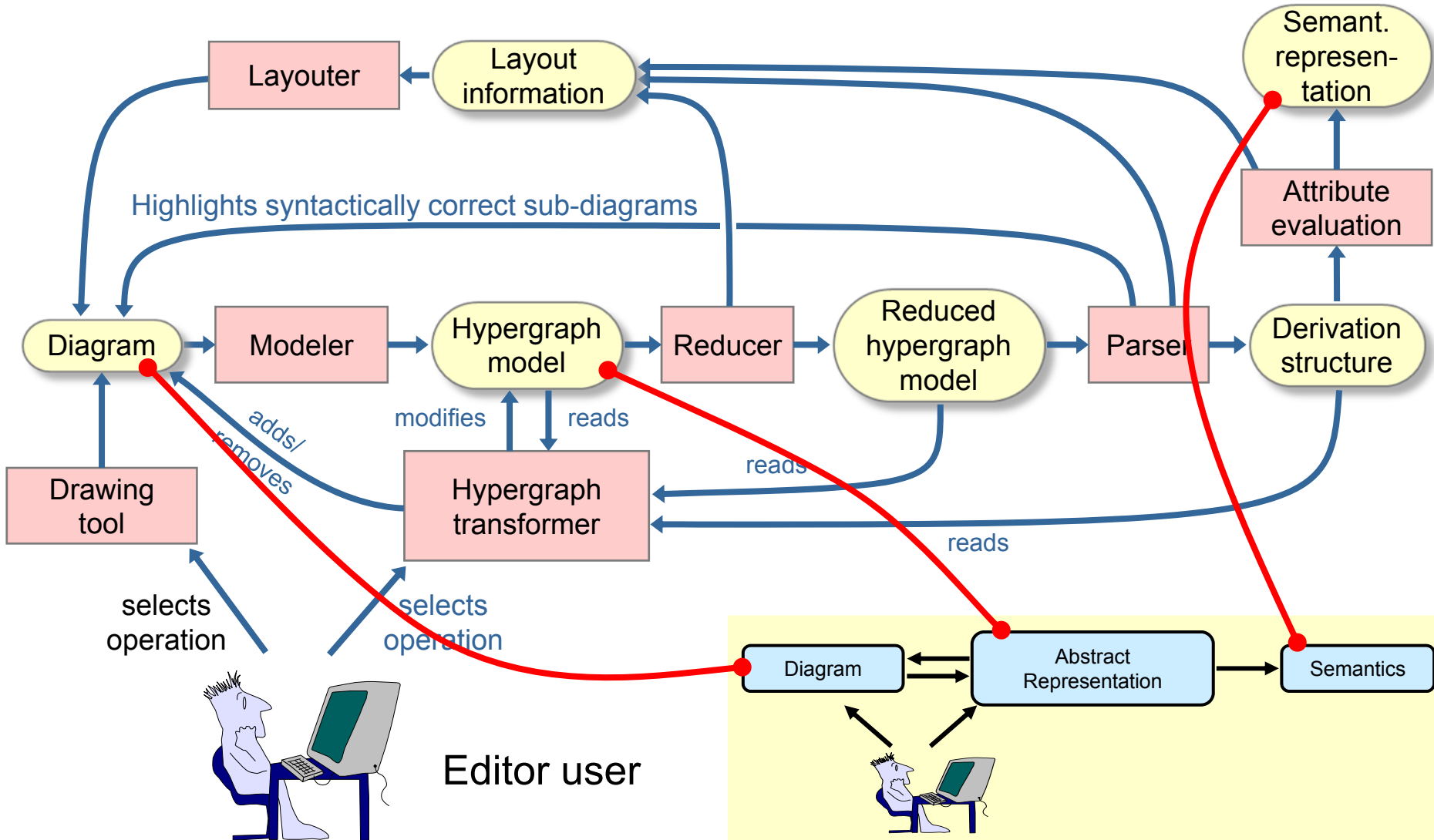


- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

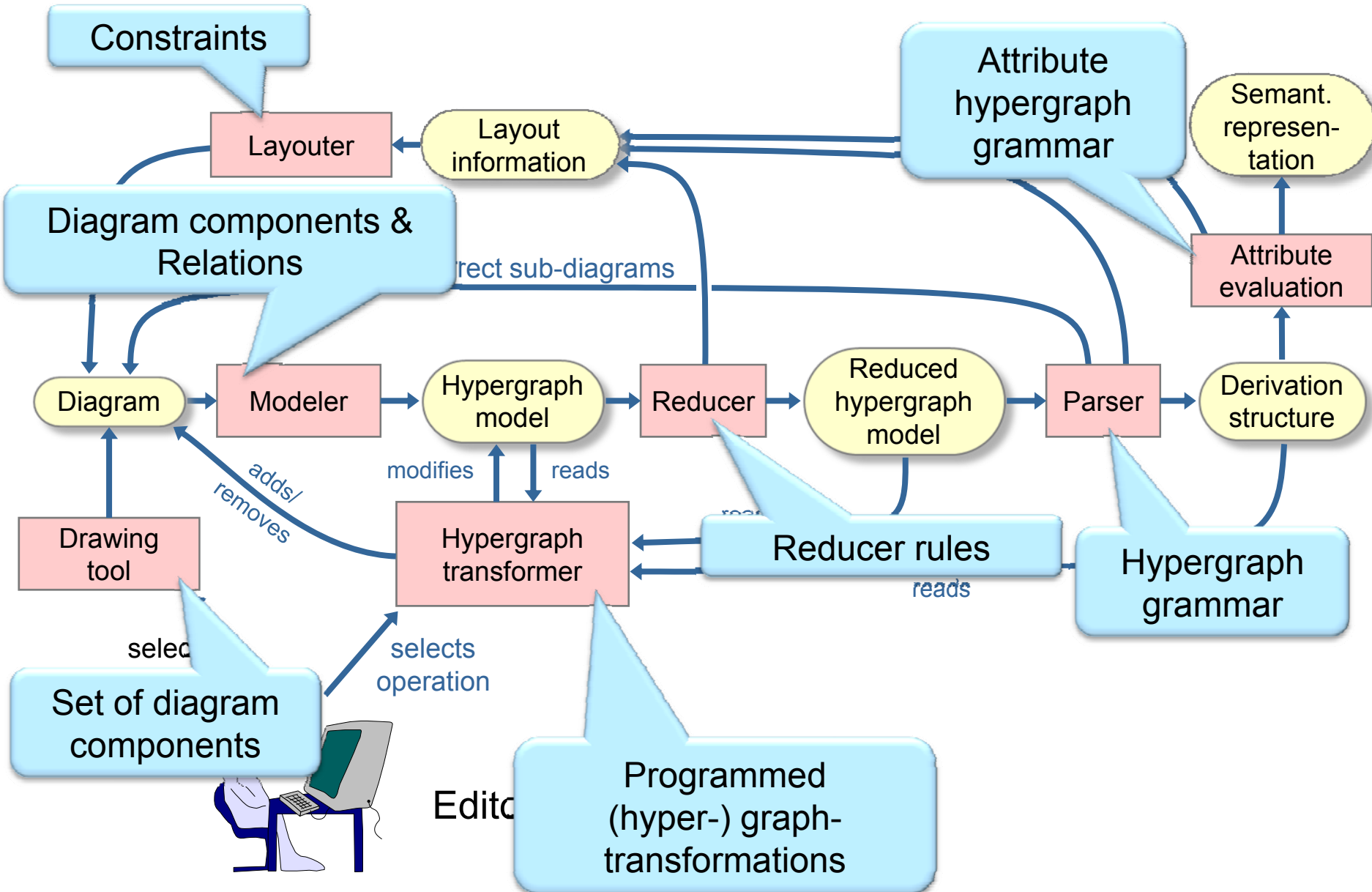
- *Rapid Prototyping*-Werkzeug zur Erstellung von Diagrammeditoren aus einer Spezifikation
- Freihändiges Editieren:
  - Zeichenprogramm
  - Lexikalische, syntaktische und semantische Analyse
- Strukturiertes Editieren (optional)
- Automatisches Layout (optional) auch bei freihändigem Editieren
- Unterstützung hierarchischer Diagramme (optional)
- Unparsing von Diagrammen, d.h. Semantik → Diagramm (optional)
- Unterstützung großer Diagramme: Semantisches Zoomen (optional)



# DiaGen: Editorarchitektur

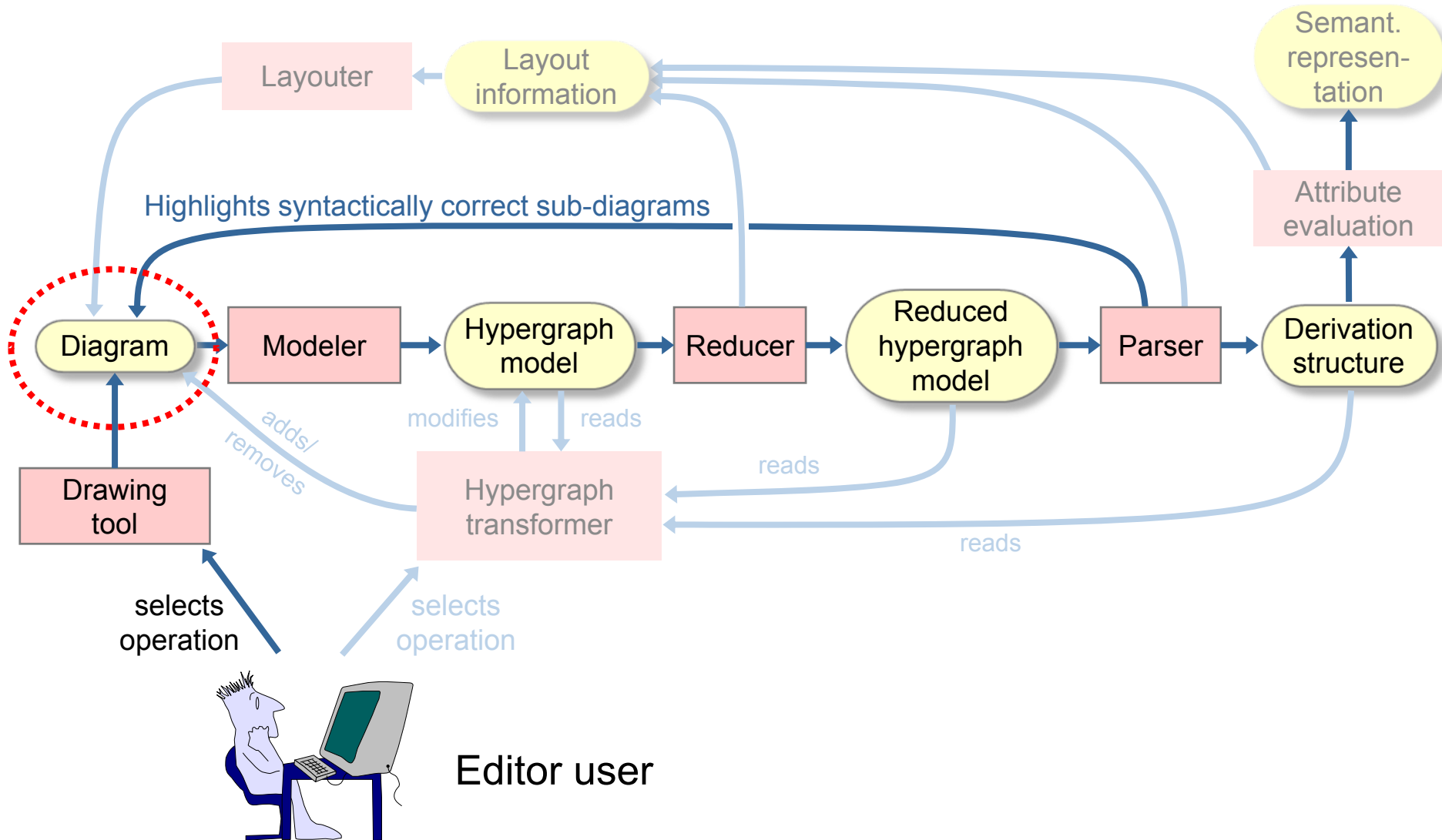


# DiaGen: Editorarchitektur



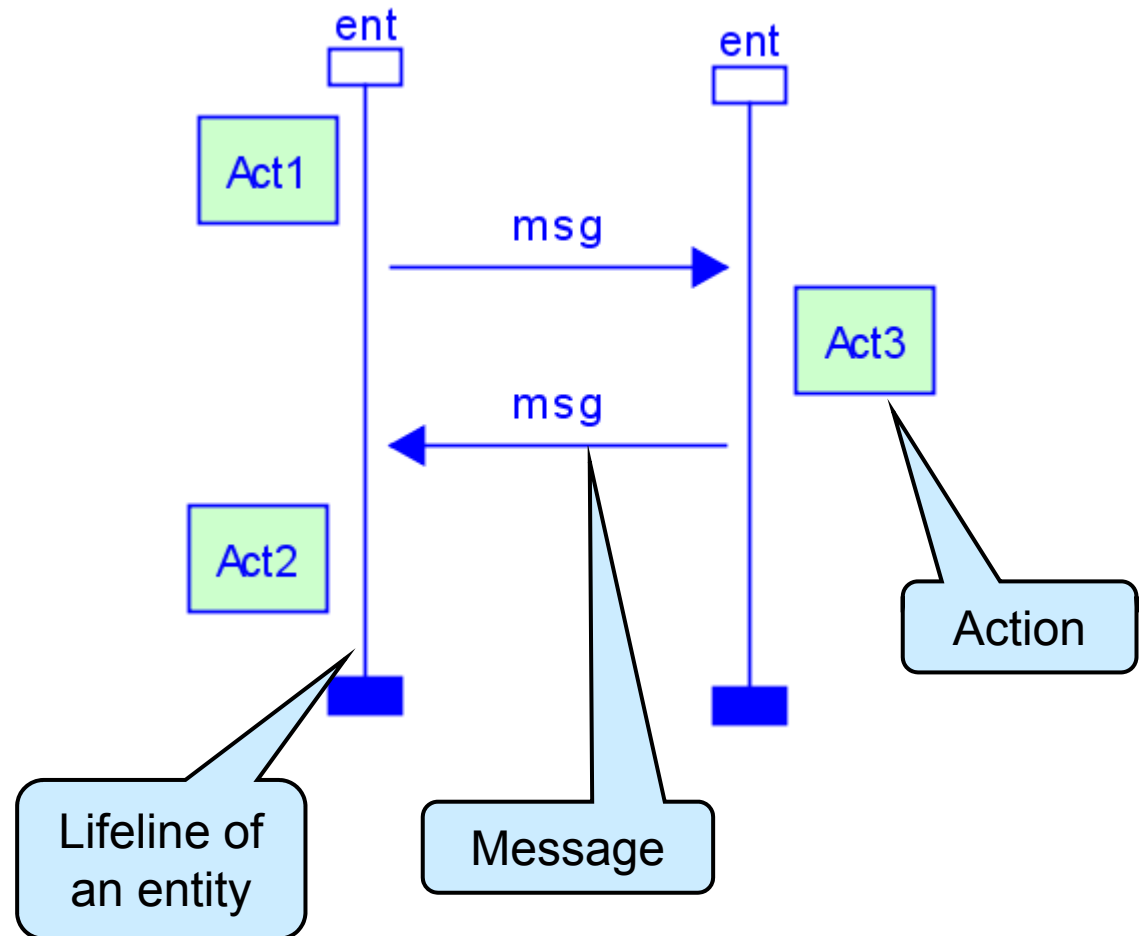
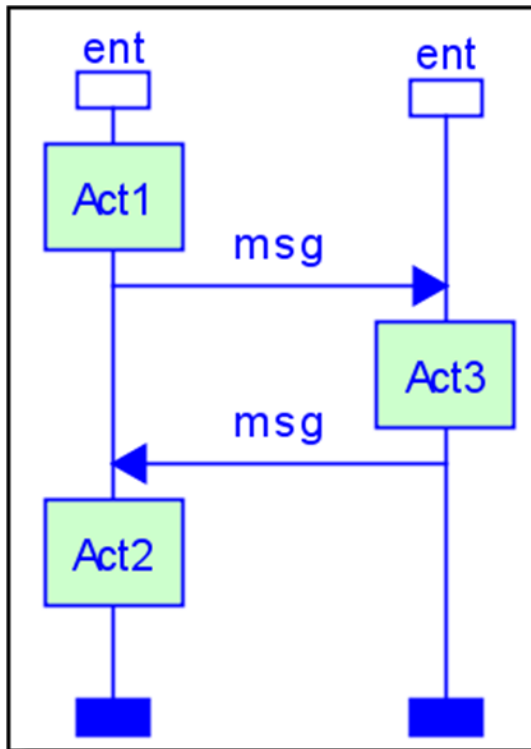
- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

# DiaGen: Editorarchitektur

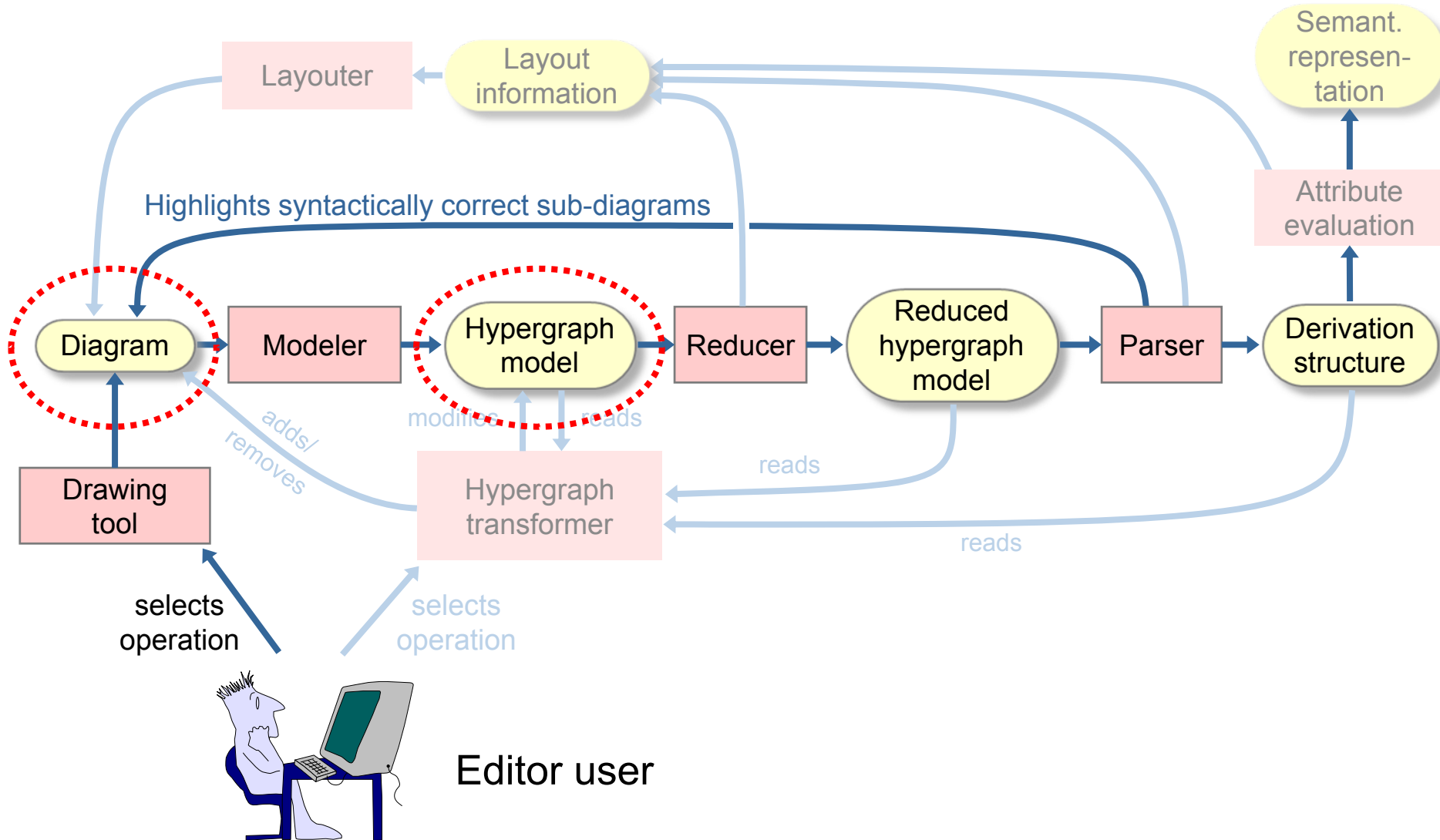


# Beispiel: Message Sequence Charts (MSC)

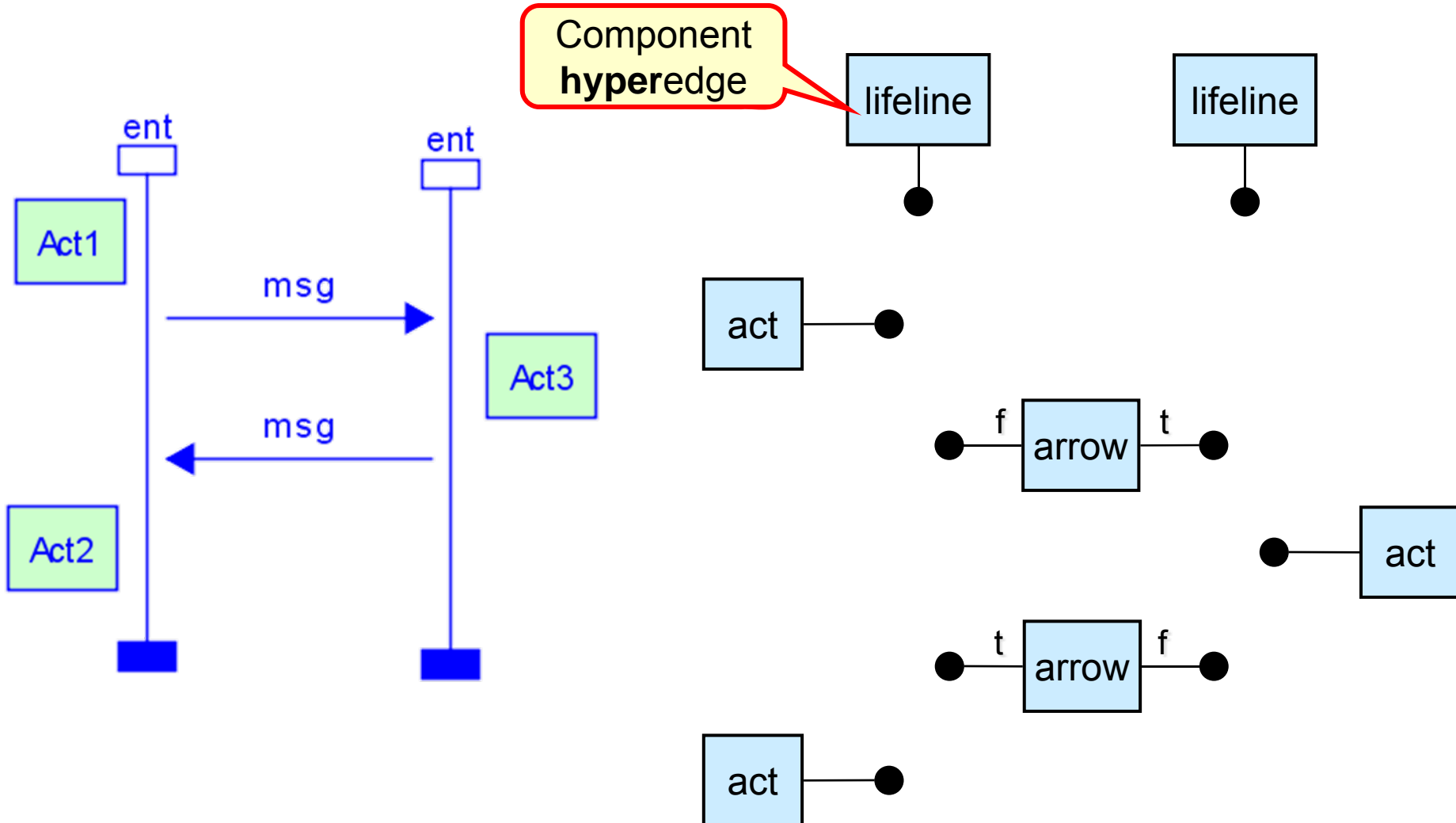
Diagrammkomponenten:



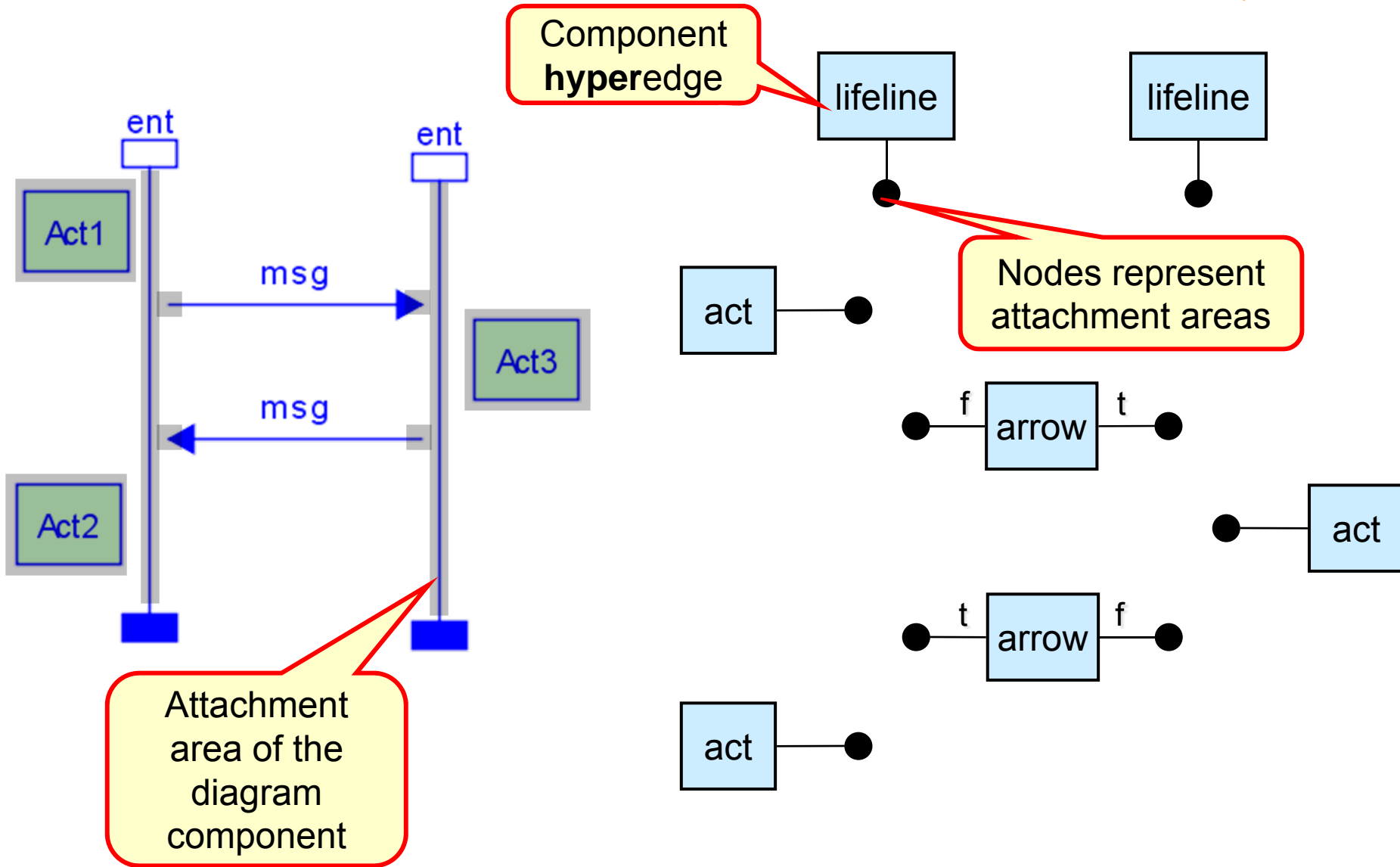
# DiaGen: Editorarchitektur



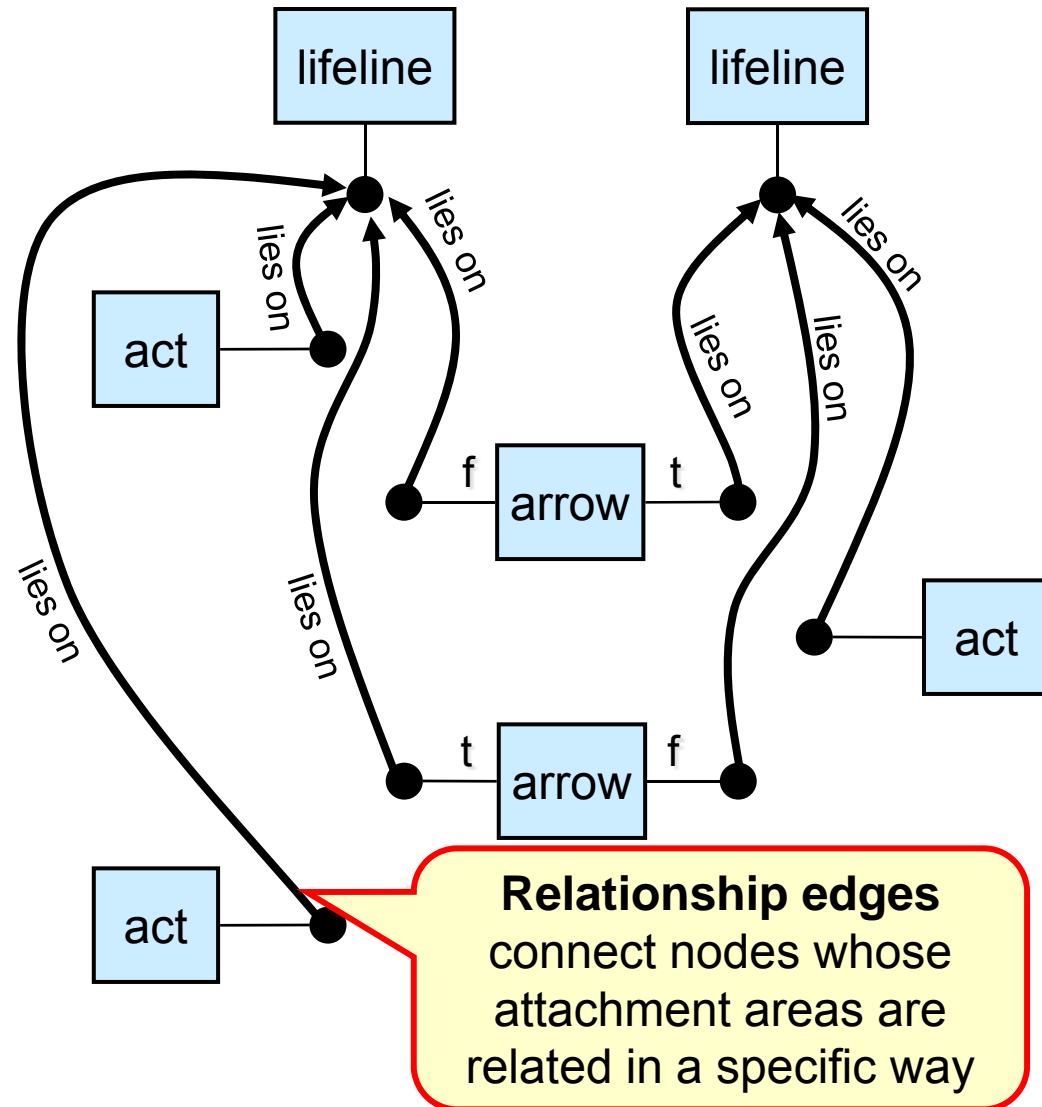
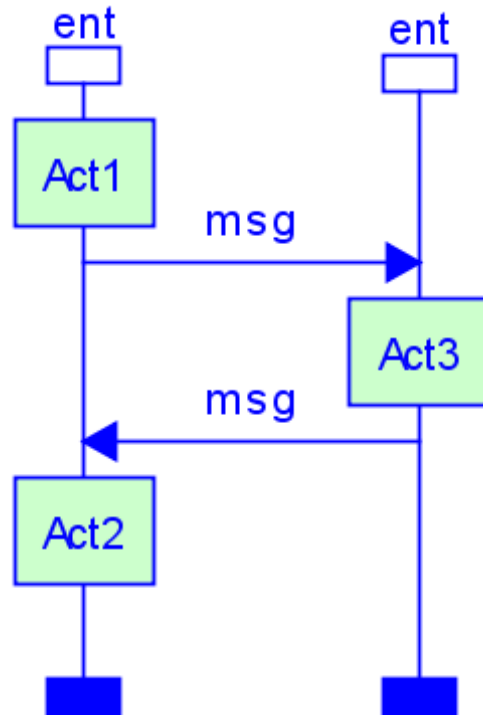
# Hypergraph-Modell



# Hypergraph-Modell

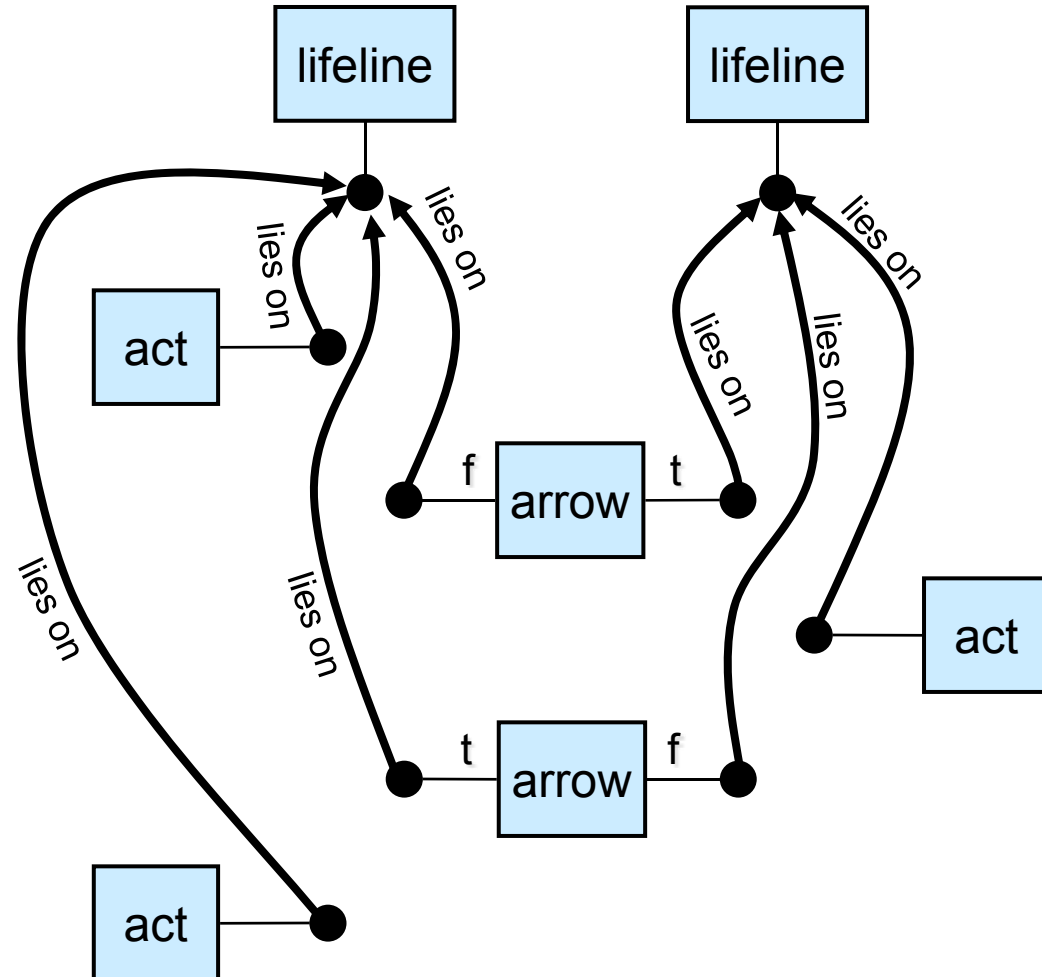
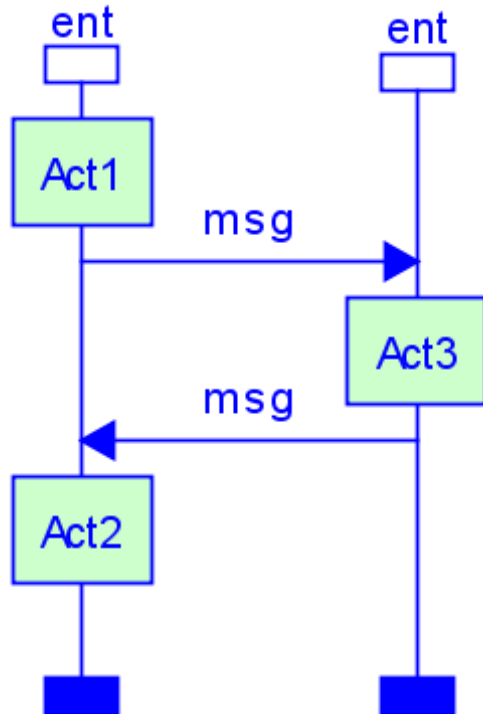


# Hypergraph-Modell

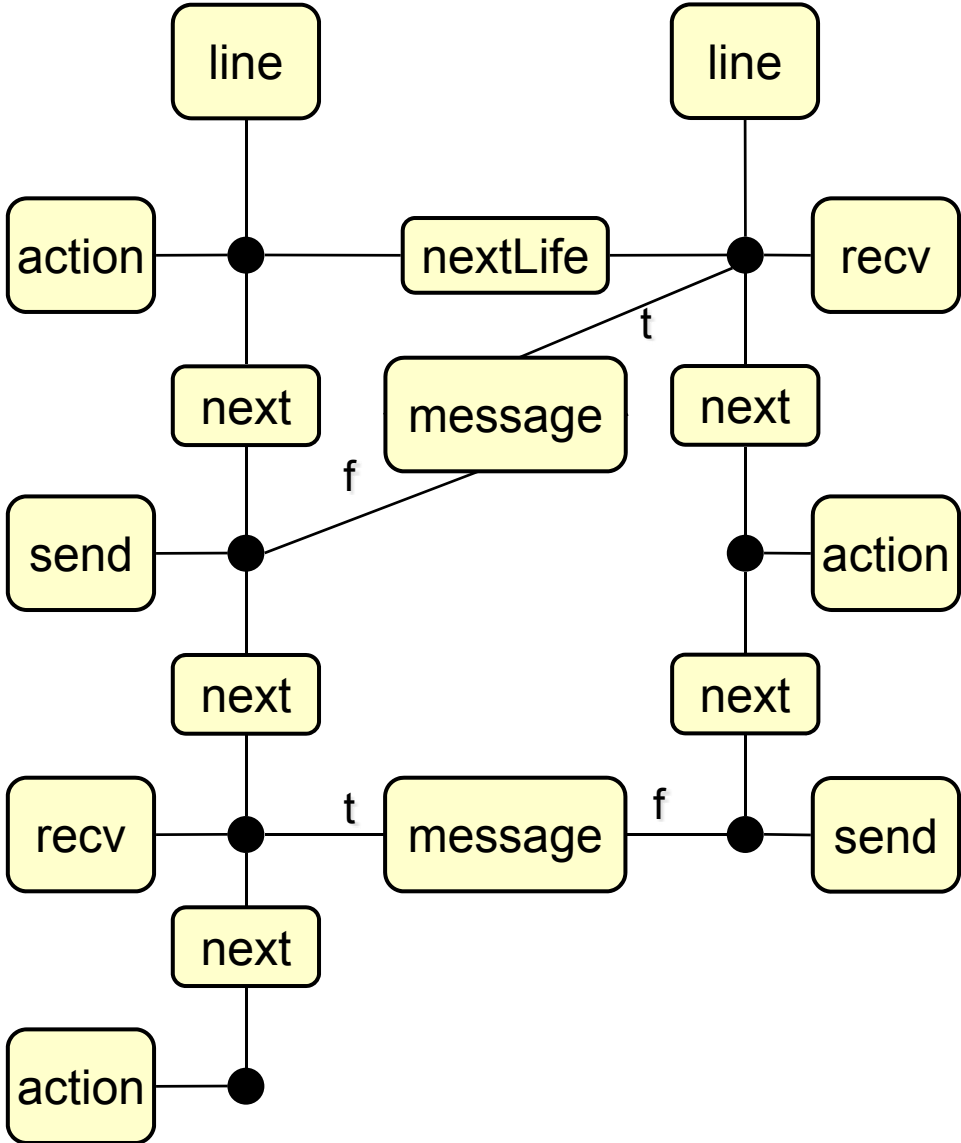
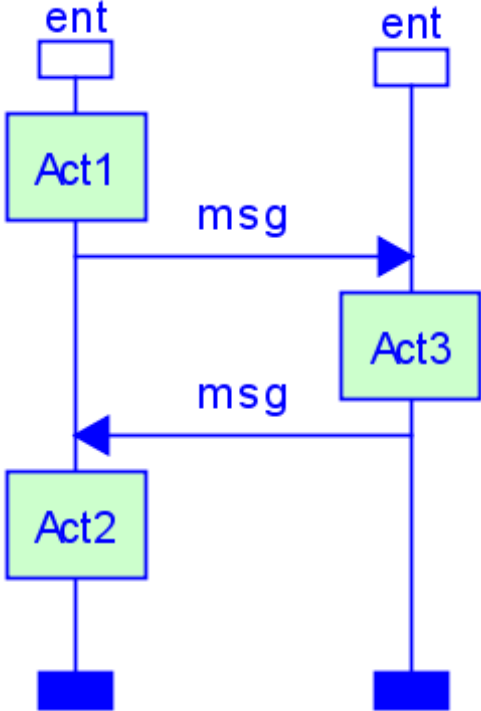




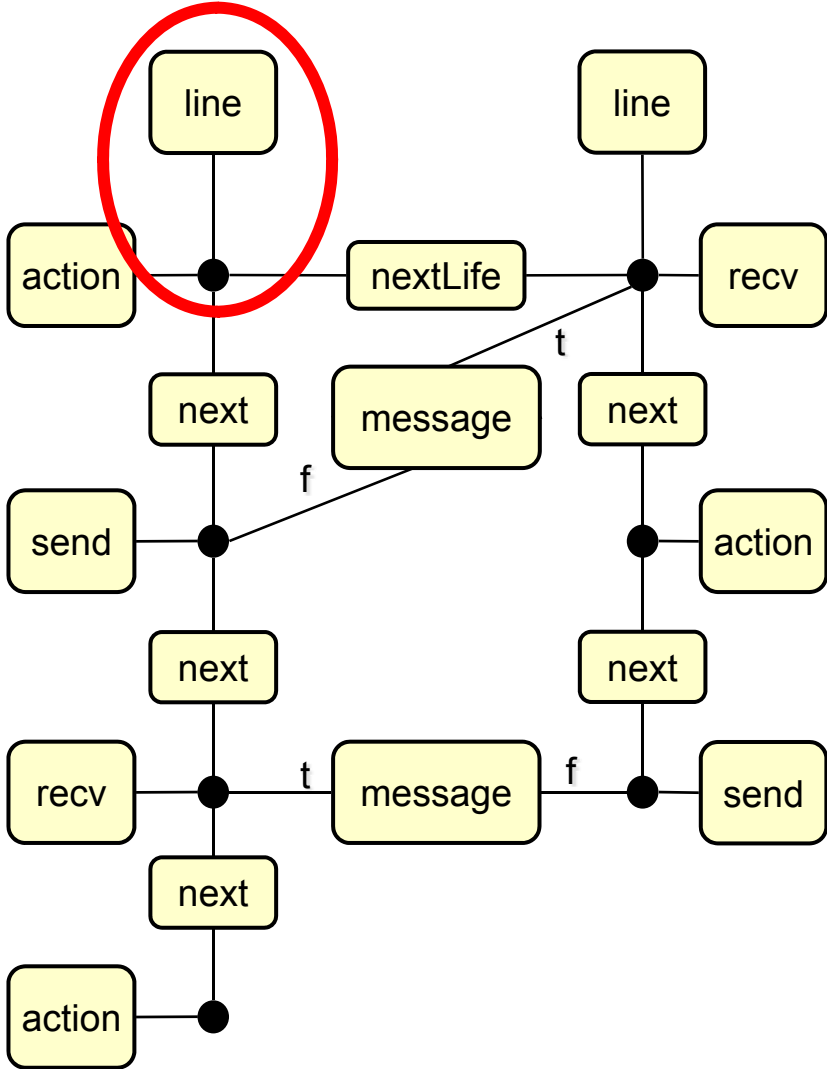
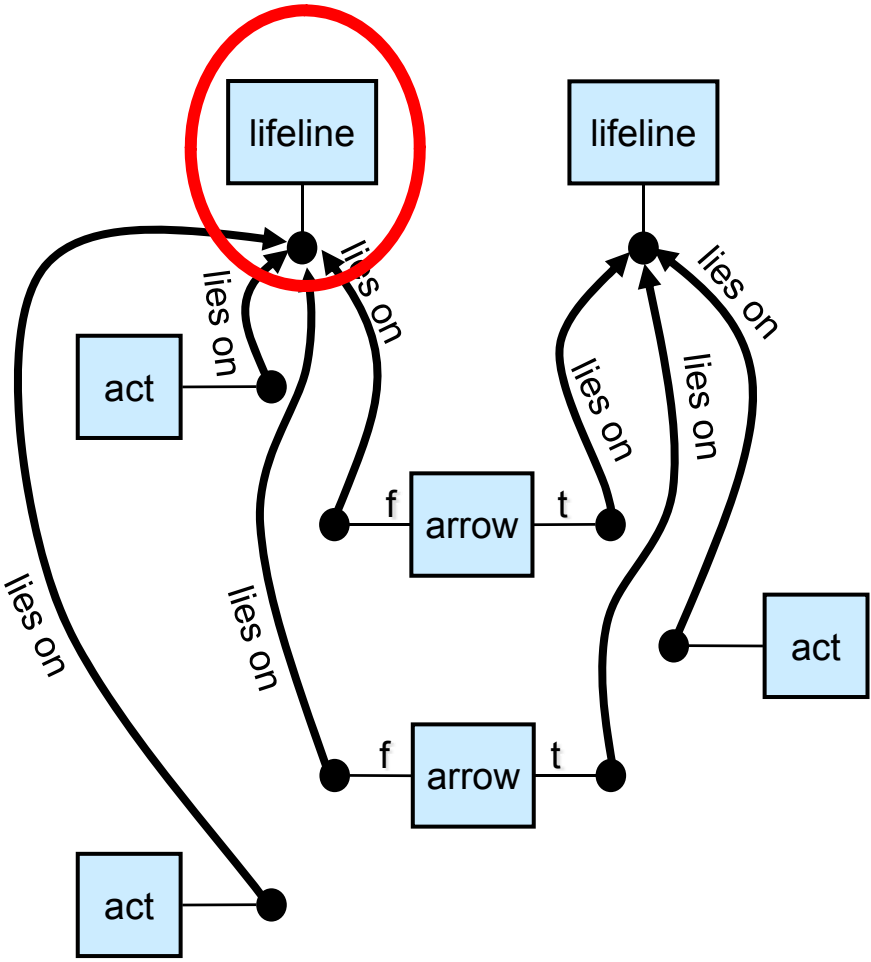
# Hypergraph-Modell



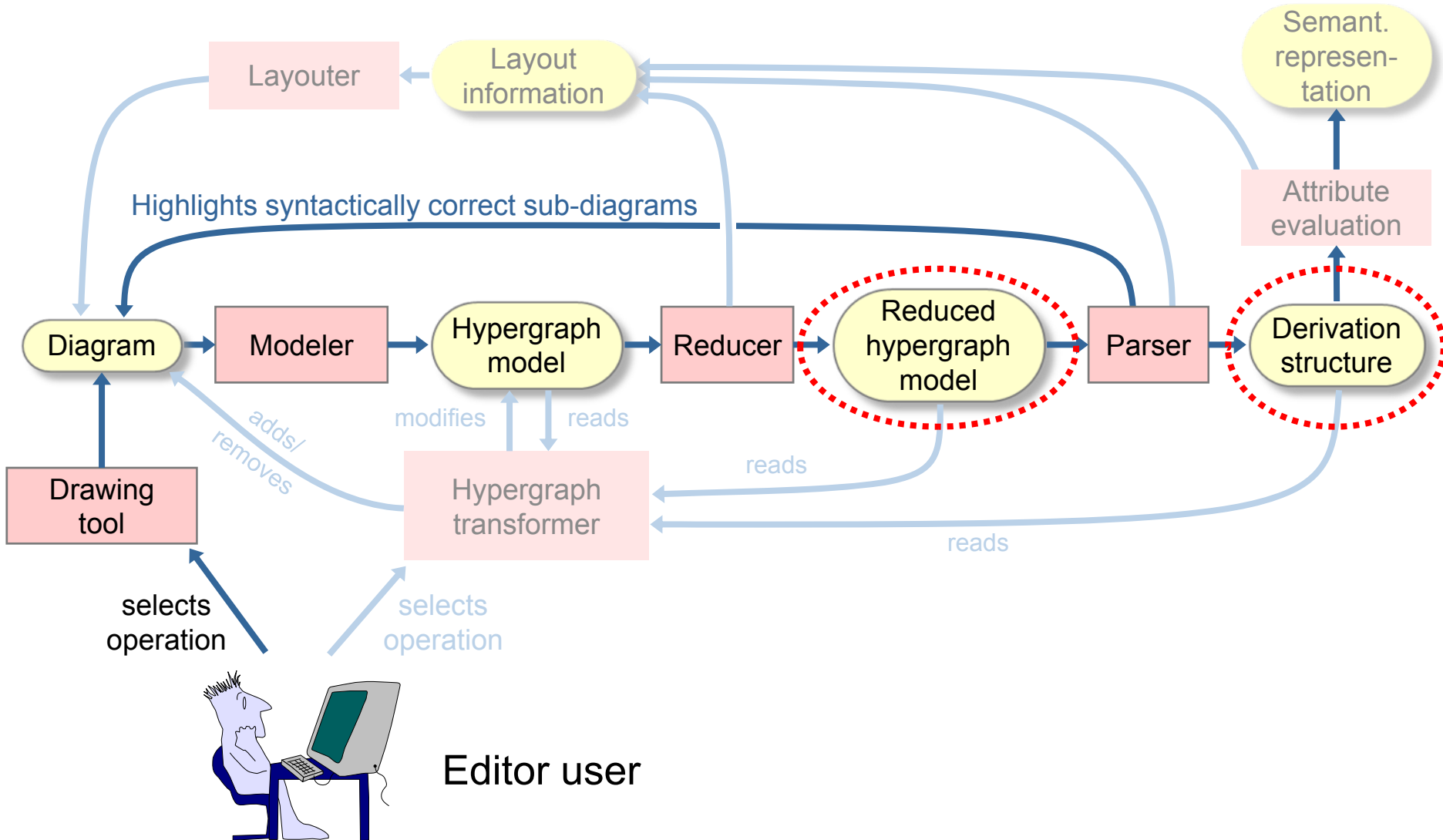
# Reduziertes Hypergraph-Modell



# Reduzieren des Hypergraph-Modells




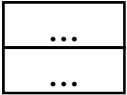
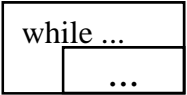
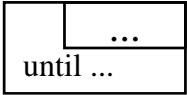
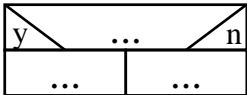
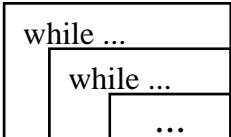
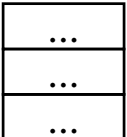
# DiaGen: Editorarchitektur

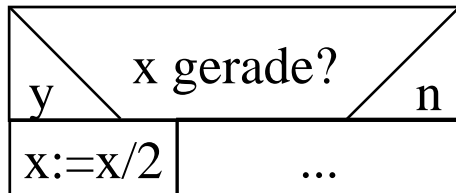


- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

# Diagrammeditoren: Möglichkeiten der Nutzerunterstützung (1)

**Was ist ein NSD?**

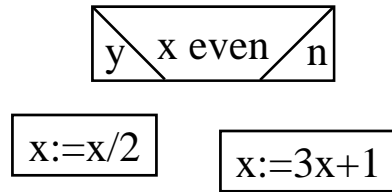
Anzahl der Komponenten	Beispieldiagramme
1	
2	  
3	  



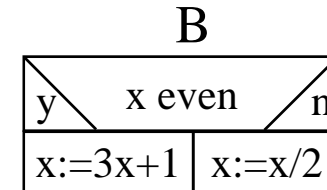
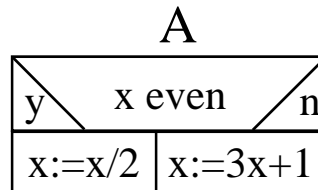
**Was ist hier falsch?**

# Diagrammeditoren: Möglichkeiten der Nutzerunterstützung (2)

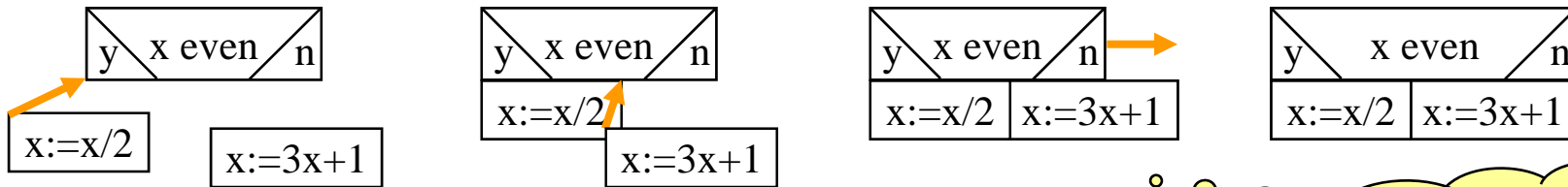
gegeben:



zwei Korrekturmöglichkeiten:



manuelle Konstruktion von Diagramm A

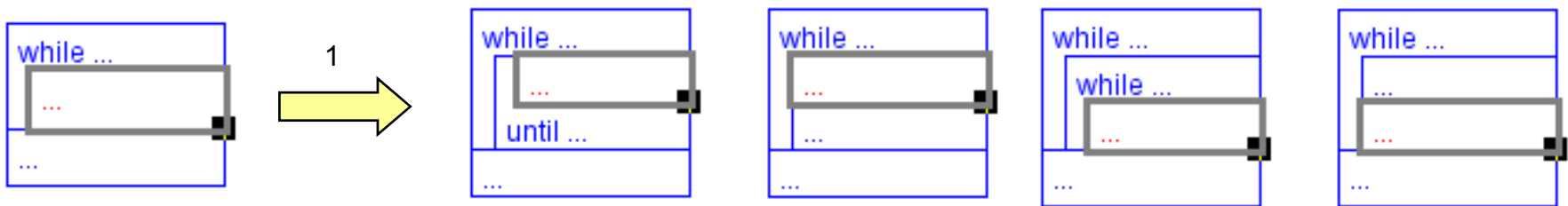


**Das ist sehr aufwändig...**

- Lösung: Diagrammkontraktion, „I‘m Feeling Lucky“-Button

# Diagrammeditoren: Möglichkeiten der Nutzerunterstützung (3)

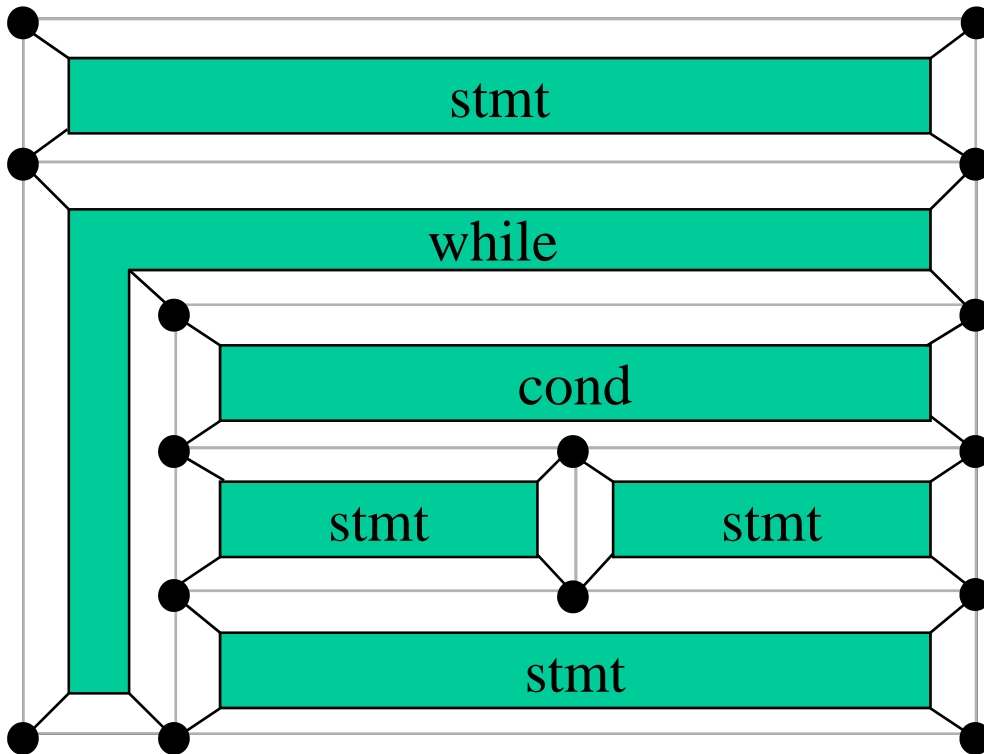
- vordefinierte, komplexe Editieroperationen sind hilfreich
- in DiaGen (und anderen, z.B. Tiger): Graphtransformationsregeln
- aber:
  - aufwändig zu definieren
  - schwierig, Korrektheit sicherzustellen
  - Editor-Entwickler weiß ggf. nicht, was der Nutzer braucht
- Lösung: Generierung lokaler Operationen "on the fly"



- Vorteile:
  - vollständige Menge aller möglichen Operationen
  - automatisch korrektkeitserhaltend

- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

# Hypergraphen als Modell für Diagramme

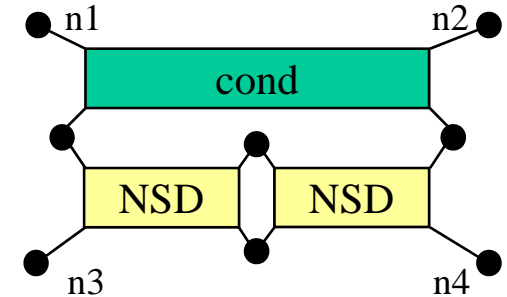
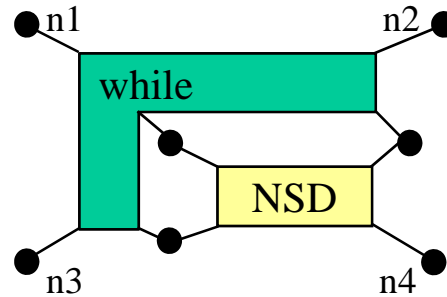
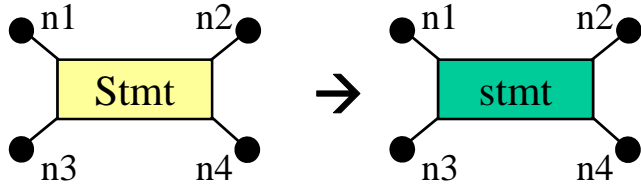
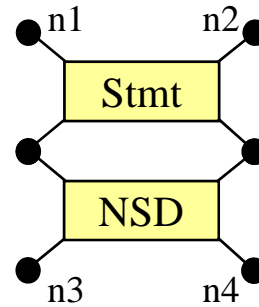
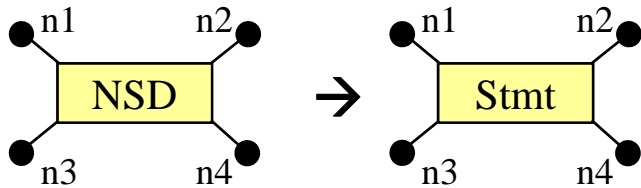


Hypergraph-Parsing

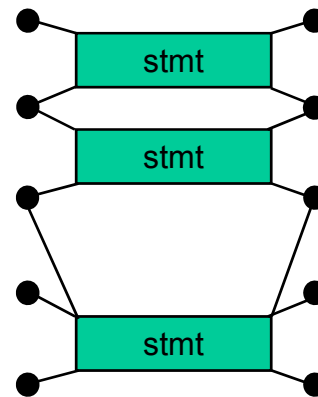
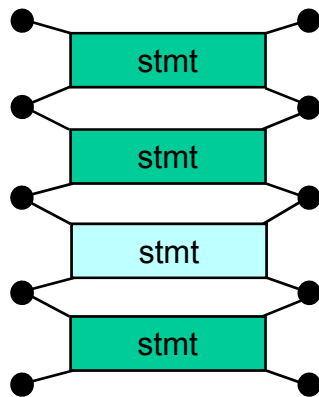


Diagramm-Parsing

# Hyperkantenersetzungsgrammatik von NSDen



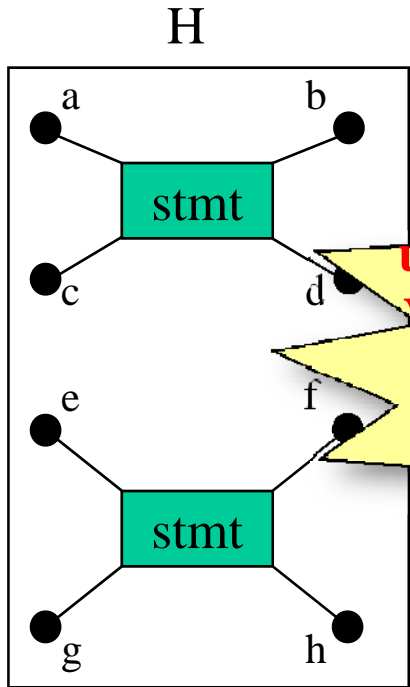
- **Hypergraph-Vervollständigung:** Veränderung eines Hypergraphen, so dass er zu einer gegebenen Sprache passt
- **Beispiel: Struktogramm-Hypergraphen**



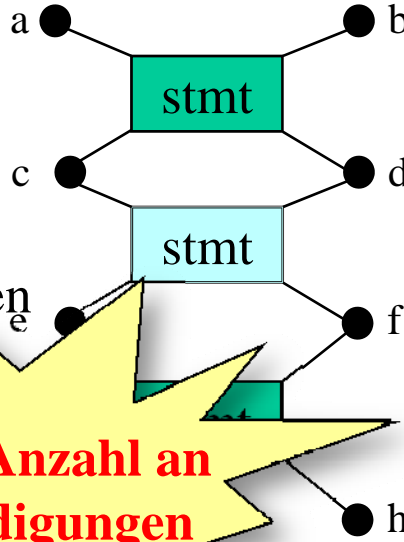
- ... ist eine hypergraph-spezifische, nicht-destruktive, korrektkeitserhaltende oder korrigierende Hypergraph-Transformation
- gegeben:
  - Grammatik  $G$  und
  - Hypergraph  $H$  (ggf. unvollständig/fehlerhaft)
- eine Vervollständigung von  $H$  entsteht durch Anwendung einer beliebigen Anzahl der folgenden Operationen:
  - neue Hyperkante zu  $H$  hinzufügen (möglicherweise mit neuen Knoten)
  - Knoten von  $H$  verschmelzen

**so dass der resultierende Graph korrekt ist bzgl.  $G$**

# Hypergraph-Vervollständigung am Beispiel

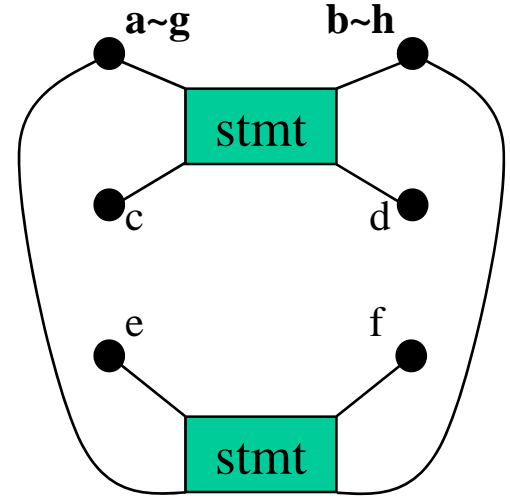
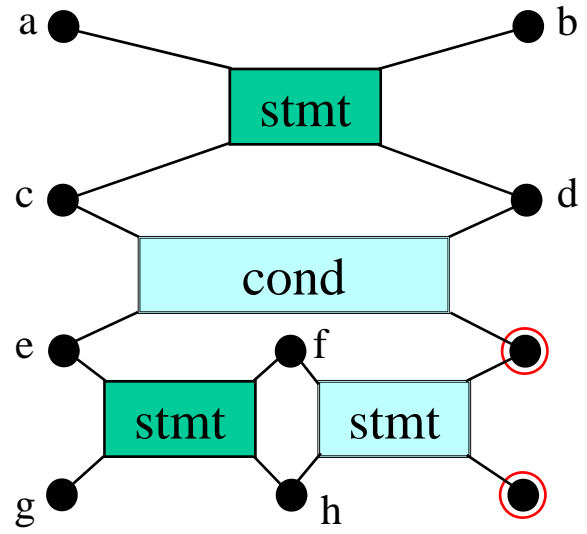
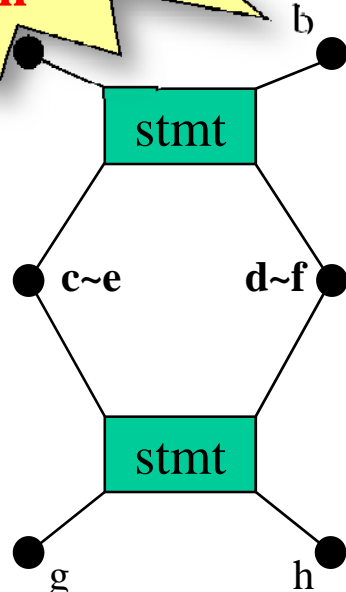


Kanten hinzufügen



**unendliche Anzahl an Vervollständigungen möglich**

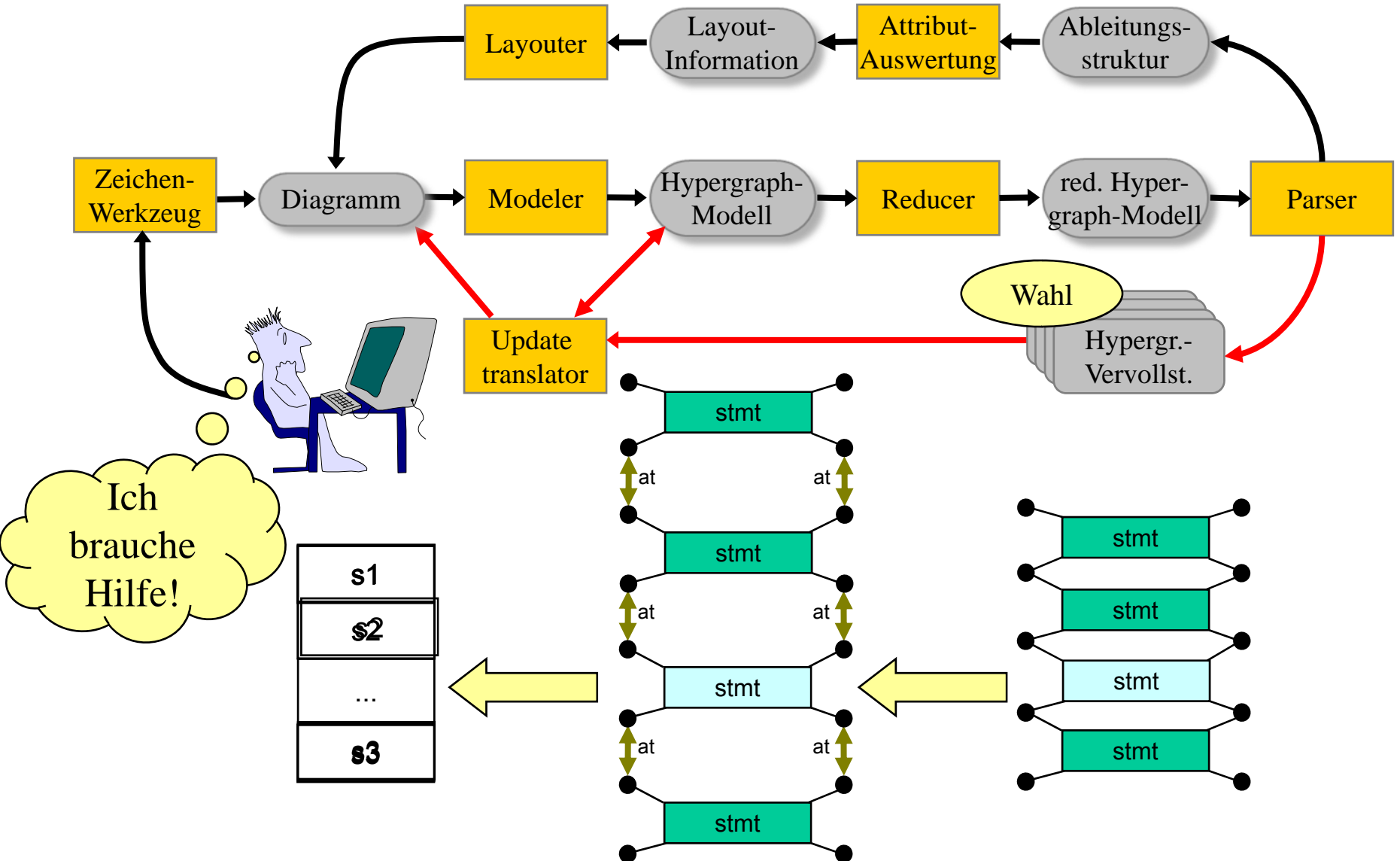
Knoten verkleben



- Parsing-Algorithmus ähnlich Cocke-Younger-Kasami-Algorithmus für Zeichenketten
- Ableitungsbäume werden *bottom up* berechnet, d.h. Rückwärtsanwendung von Produktionen
- für Vervollständigung:
  - Kanten „vortäuschen“ wo möglich
  - Knoten verkleben wo nötig
  - ➔ Äquivalenzklassen von Knoten (Quotientengraph)

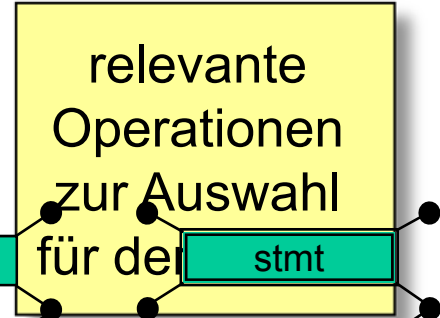
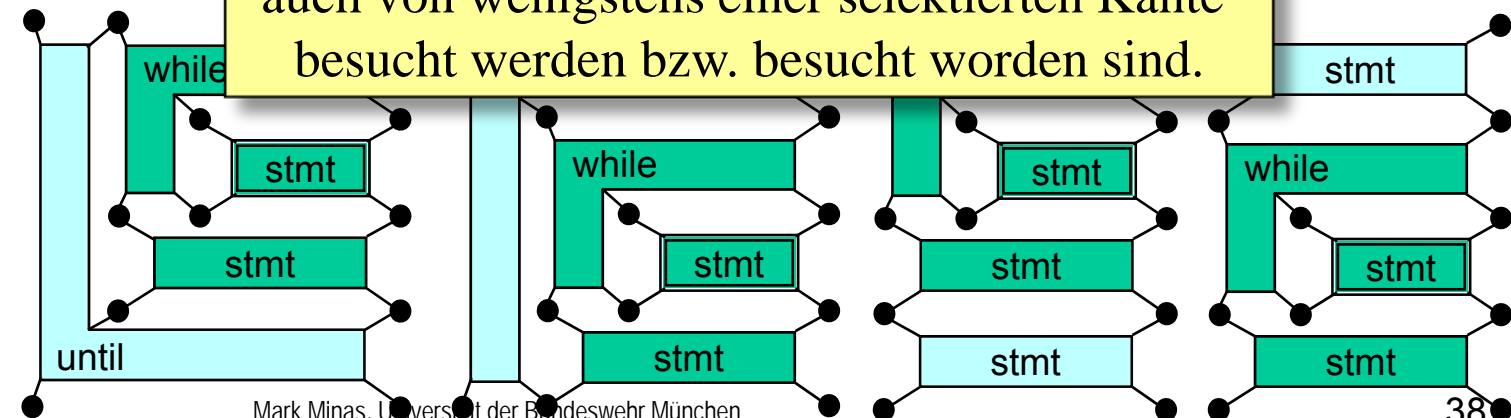
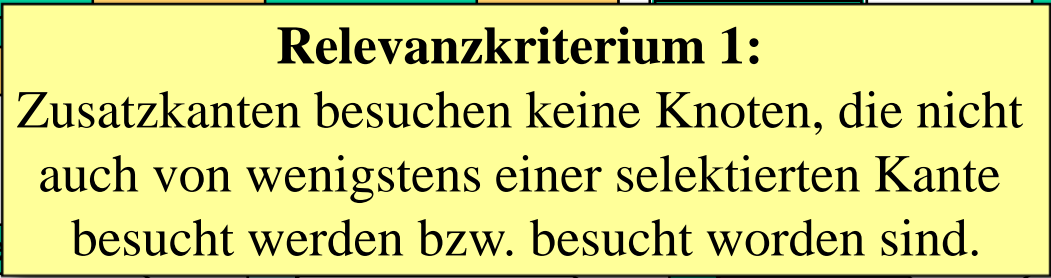
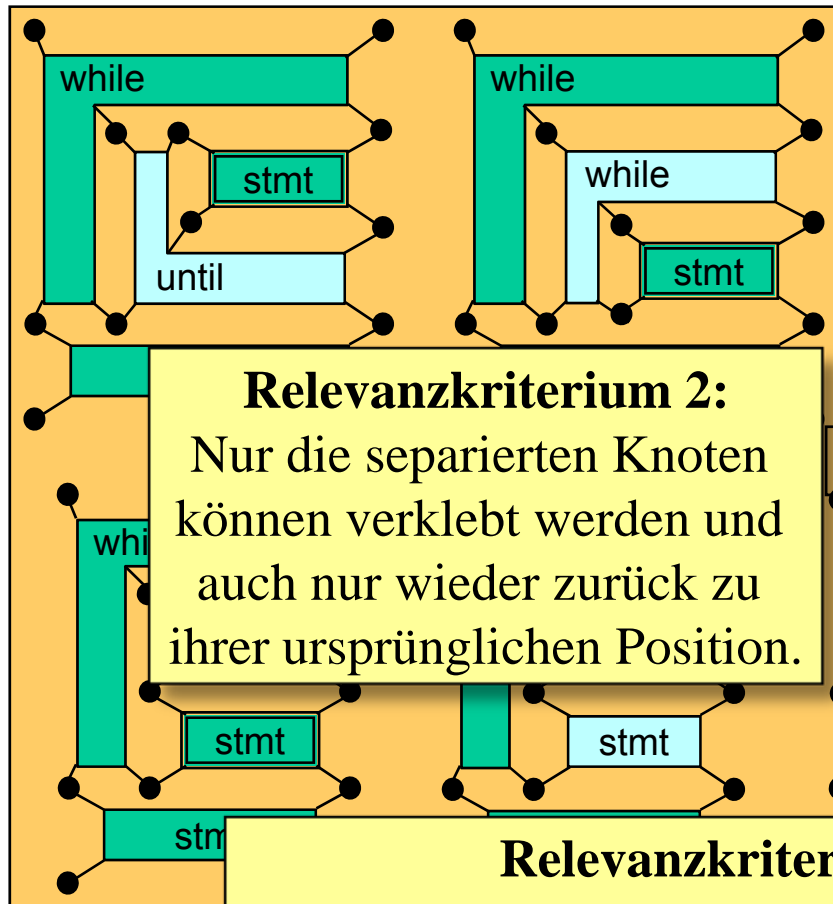
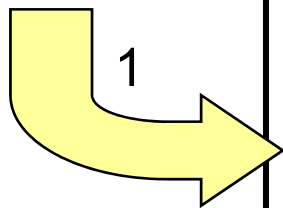
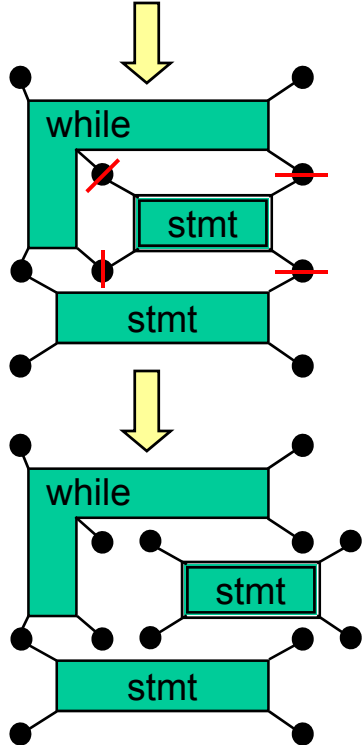
**zusätzlicher Parameter:**  
maximale Anzahl Kanten, die  
vorgetäuscht werden dürfen

# Nutzung in DiaGen

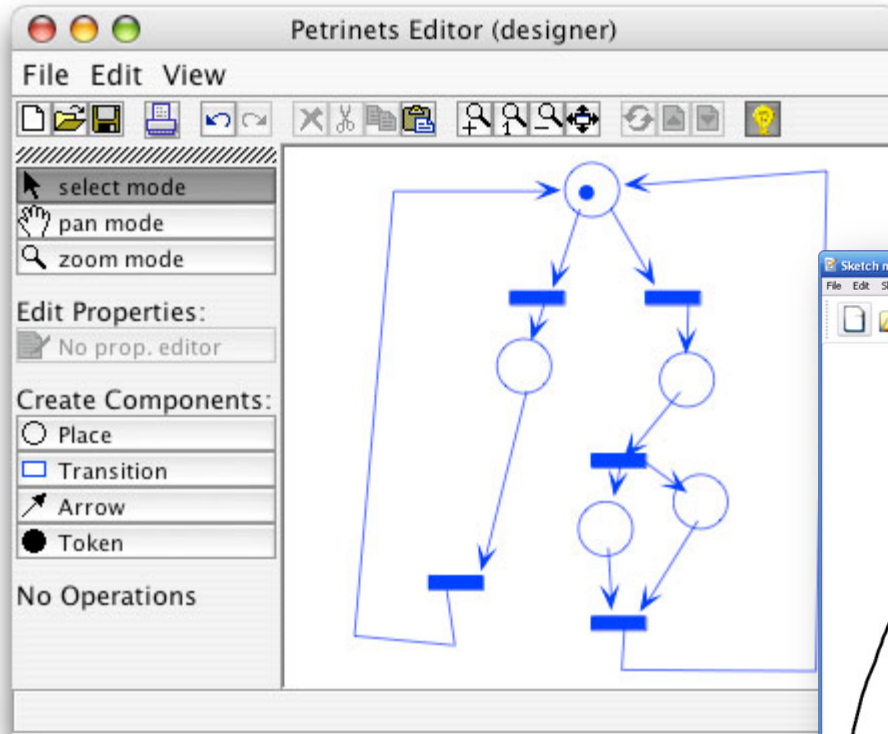


- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

- Nutzer selektiert Komponente(n)
- zugehörige terminale Hyperkanten werden aus ihrem Kontext "herausgelöst"
  - Aufspaltung von Schnittknoten
- Hypergraphvervollständigung anwenden
- Relevanzkriterien prüfen

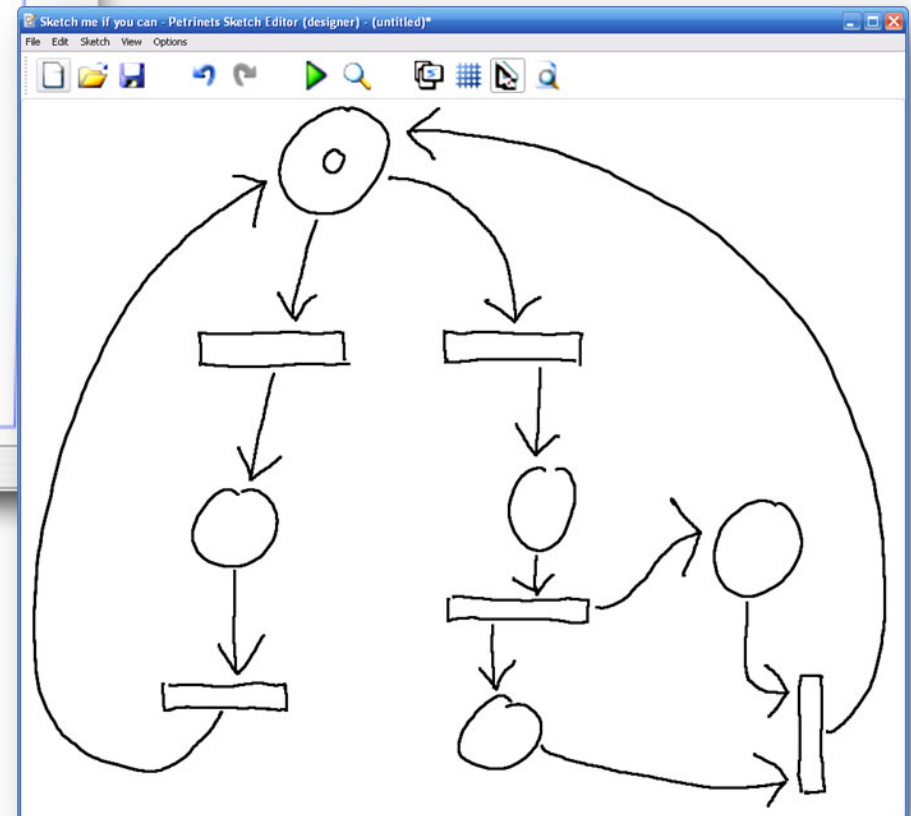


- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung



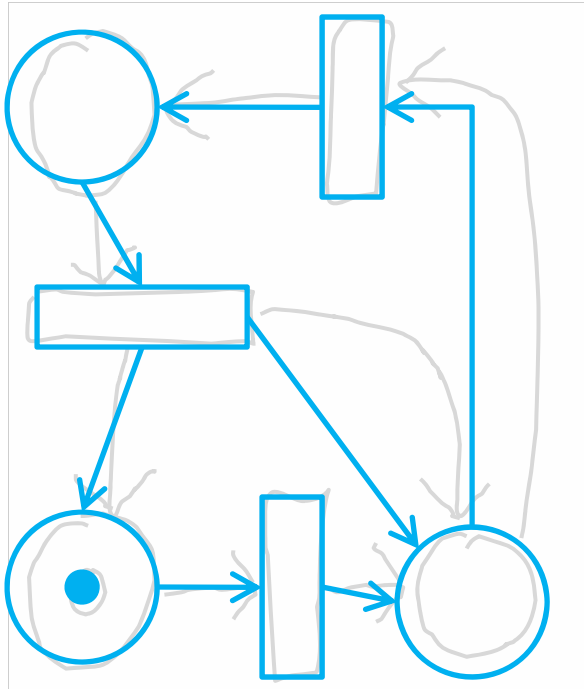
Herkömmlicher  
Ansatz

Sketching-Editor

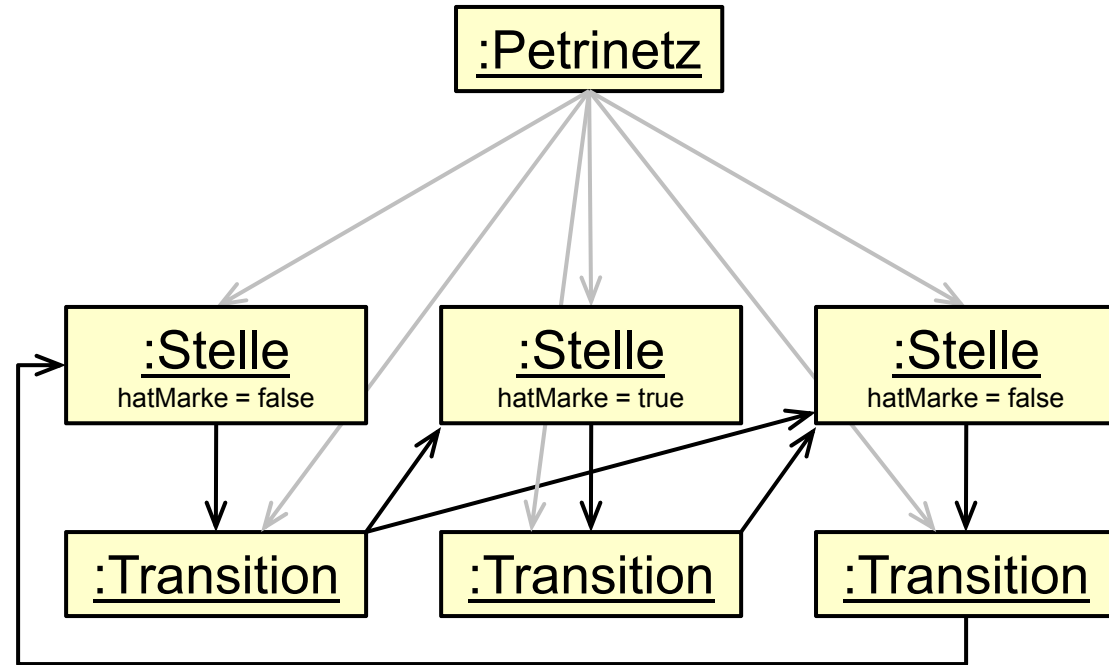


- Beobachtung bei herkömmlichen Diagrammeditoren:
  - häufig erster Entwurf auf Papier, dann mühevoll auf Computer übertragen
- Sketching:
  - Strichzeichnungen mit anschließender Erkennung
  - Sehr natürlich und intuitiv
- Vorteil gegenüber herkömmlichen Diagrammeditoren:
  - einfaches User-Interface
- Vorteil gegenüber Papier:
  - Laden, Speichern, Kopieren, Versionieren, etc.
  - kein Übertragen der Skizze auf den Rechner nötig

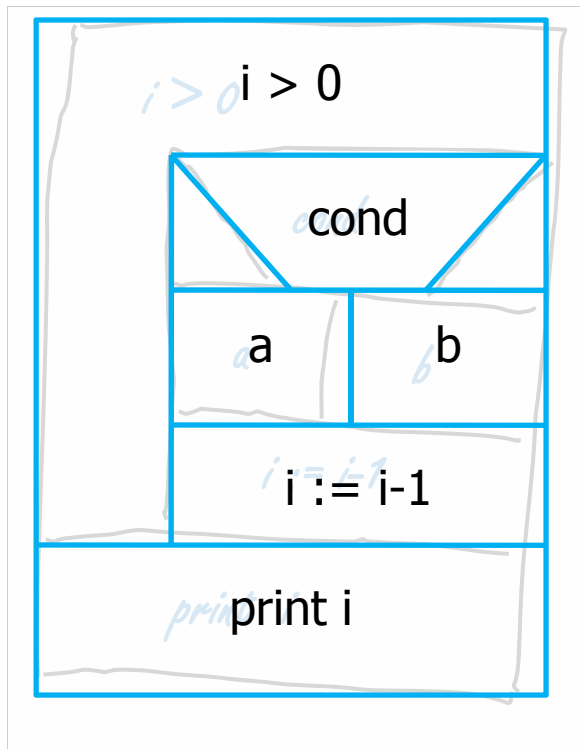
## Petrinetz



## Objektmodell



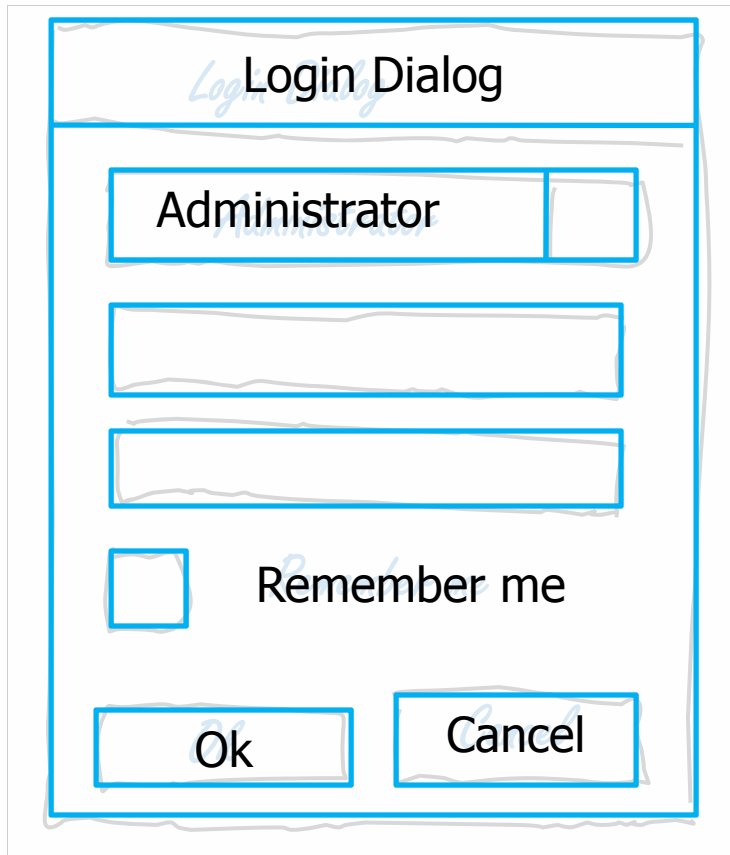
## Struktogramm



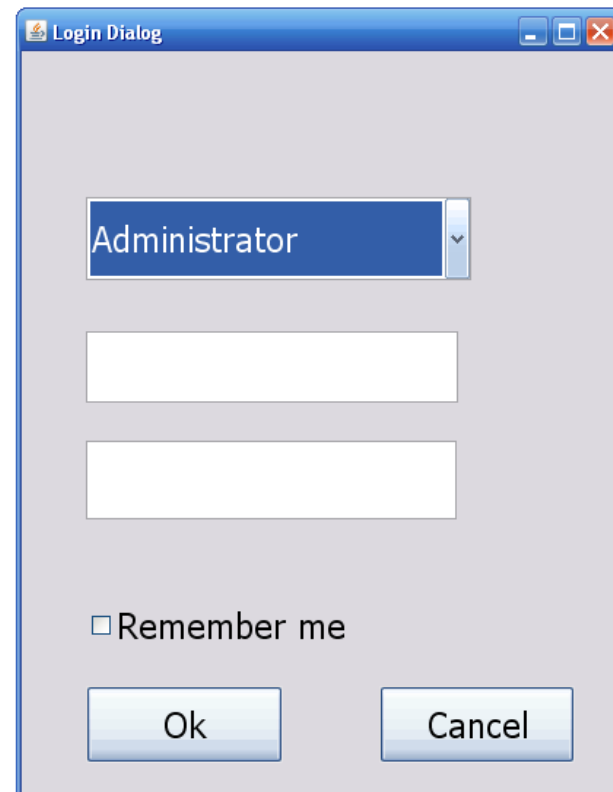
## Programmtext

```
while i > 0 loop
    if cond then
        a
    else
        b
    end if
    i := i-1
end loop
print i
```

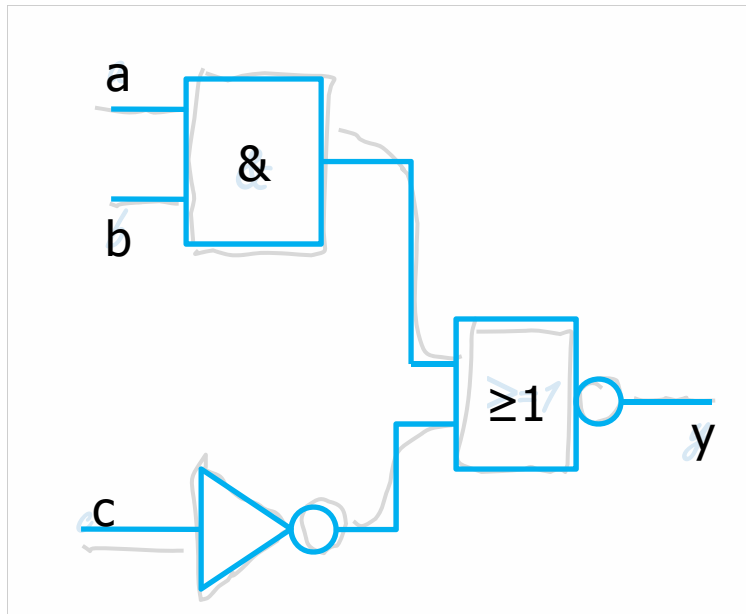
## GUI Builder



## Fenster



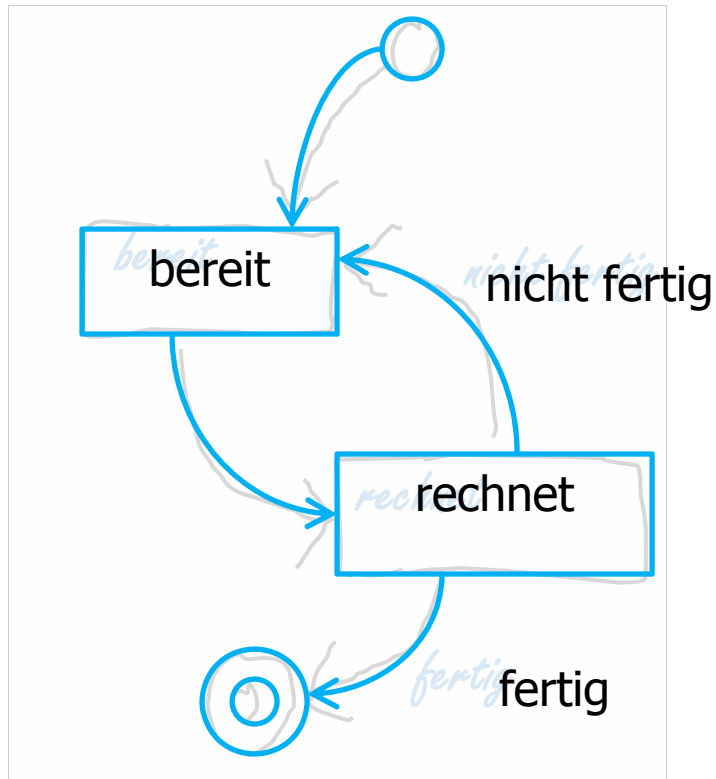
## Logikschaltungen



## Ausdruck

$y := \text{not} ((a \text{ and } b) \text{ or } (\text{not } c))$

## Zustandsdiagramme



## Textuelle Repräsentation

### Startzustand

→ bereit

### Bereit

→ rechnet

### Rechnet

→ bereit (nicht fertig)

→ Endzustand (fertig)

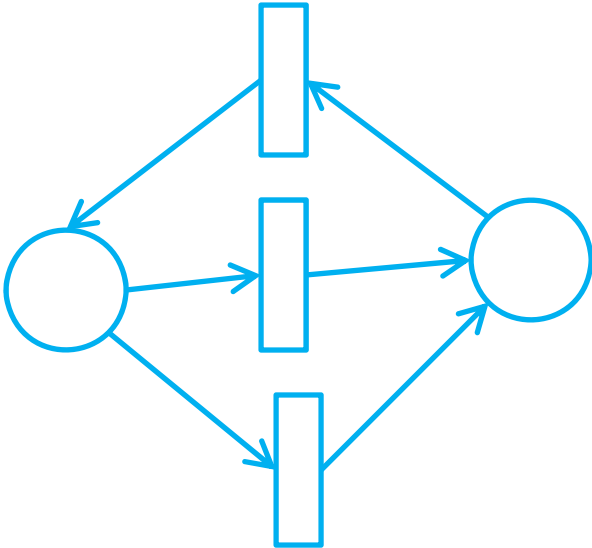
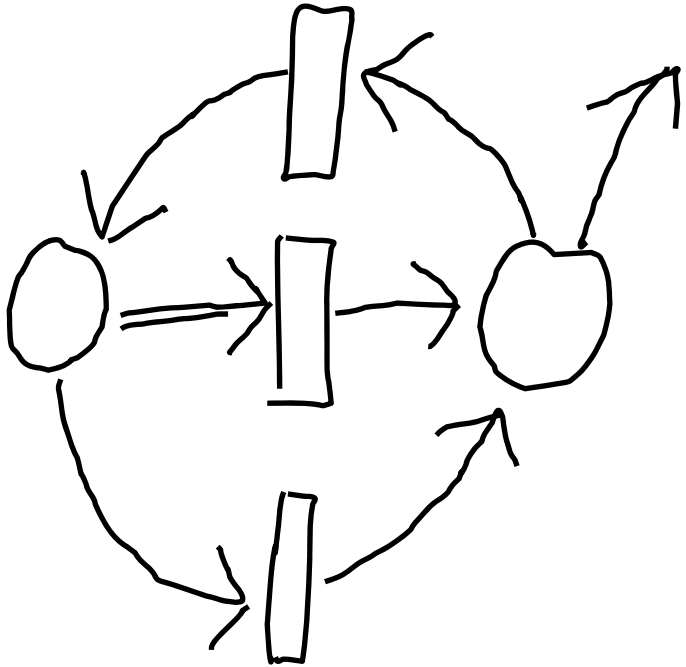
### Endzustand

- Größte Herausforderung:
  - Ungenauigkeit der Benutzereingabe
  - Ungenauigkeit der Erkennung
- Stand der Technik:
  - Annahmen → Einschränkungen:
    - ▶ Diagrammkomponenten mit einem Strichzug gemalt
    - ▶ Diagrammkomponenten müssen in vorgegebener Art und Weise gemalt werden
    - ▶ keine nachträgliche Veränderung von Diagrammkomponenten
  - Mängel:
    - ▶ keine Berücksichtigung von Kontextinformation (=Berücksichtigung der Diagrammsyntax)

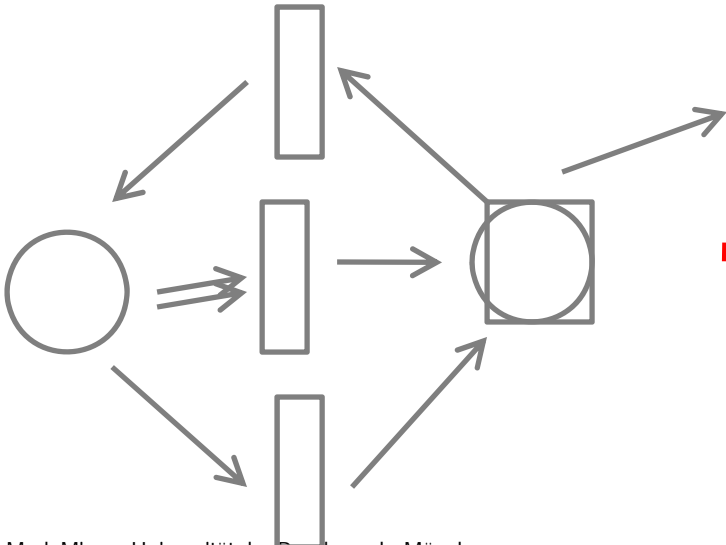
- Online- vs. Offline-Erkennung
- Generischer vs. spezifischer Ansatz
- Frameworks vs. Anwendungen
- Laufende Erkennung vs. Explizites Anstoßen
- Ausnutzen von Kontextinformation vs. kein Ausnutzen ...
- Integration in bzw. Erweiterung von DiaGen
  - Spezifikation von Diagrammkomponenten und Syntax
- Beschränkung auf modusbasierte Texteingabe
  - Rubine-Texterkennung
  - Microsoft-Texterkenner
  - Tastatur

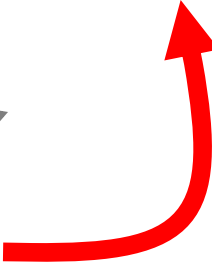
- Editiermodi bei Diagrammeditoren
- **DiaGen**
  - Konzept
  - Diagrammanalyse bei DiaGen-Editoren
- **Benutzerassistenz**
  - Möglichkeiten der Benutzerunterstützung
  - Diagramm- & Hypergraphvervollständigung
  - Automatische Generierung strukturierter Editieroperationen
- **Sketching**
  - Beispiele
  - Probleme & Einordnung
  - Erkennung & Analyse
- Zusammenfassung

# Erkennung & Analyse



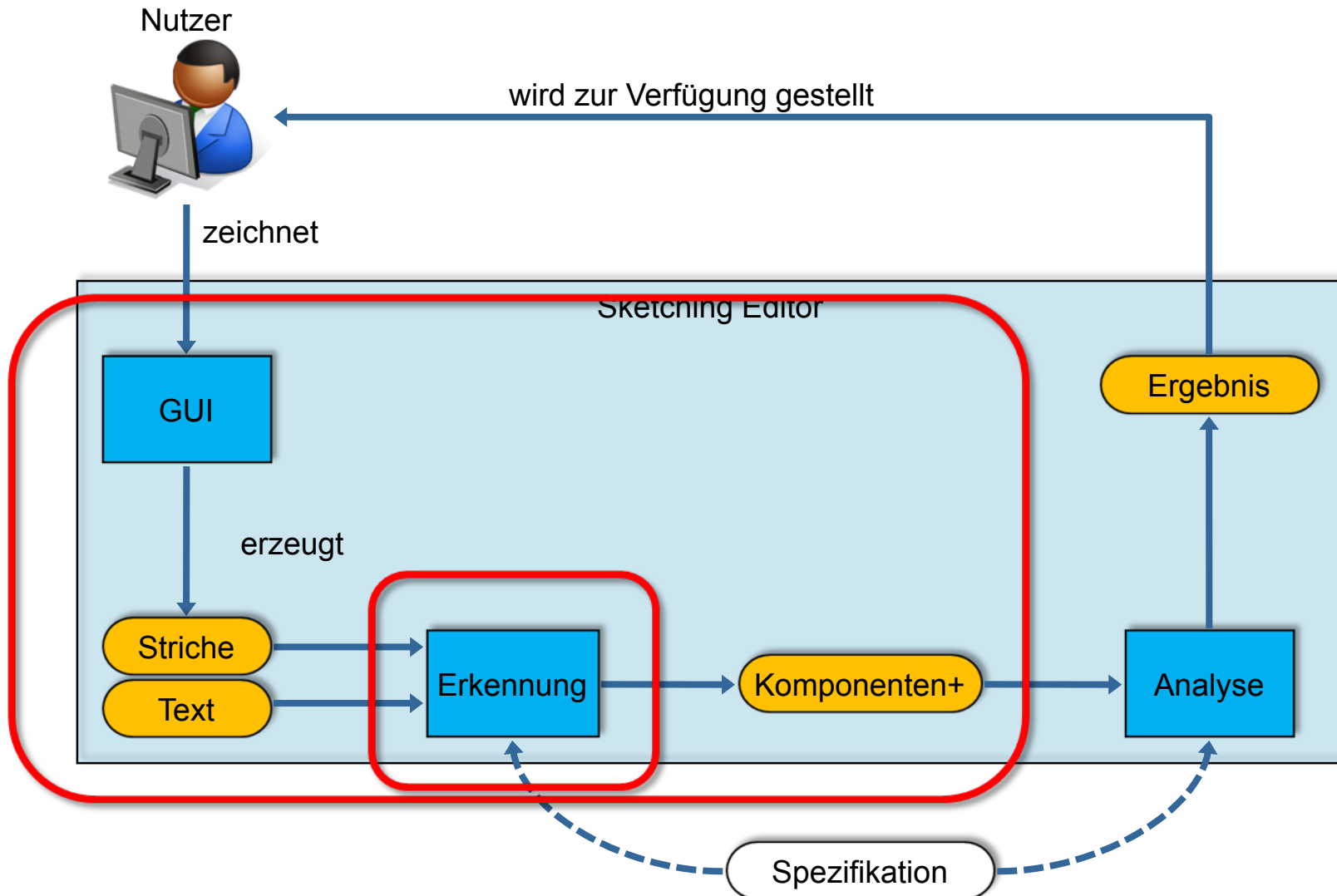
  
Erkennung von  
Komponenten



  
Analyse

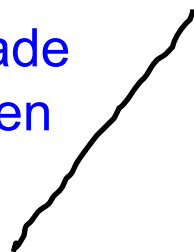


# Architektur Sketching-Editor

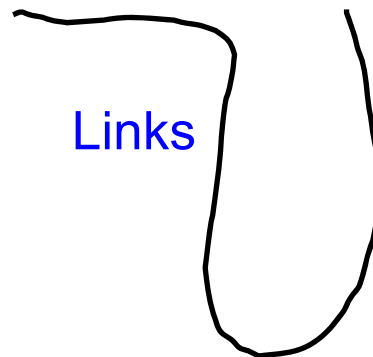


- Komponenten sind aus Primitiven zusammengesetzt
- Primitive sind fix und werden nicht spezifiziert
- Ein zusammenhängend gemalter Strich kann aus verschiedenen Primitiven zusammengesetzt sein
- Vier verschiedene Primitive:

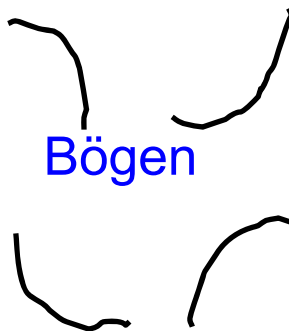
Gerade  
Linien



Links



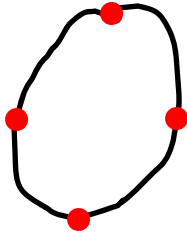
Bögen



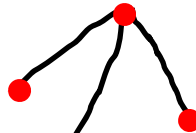
Hallo Welt  
Text

# Beispiele

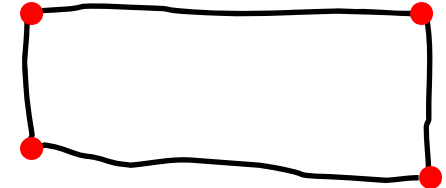
Vier Bögen



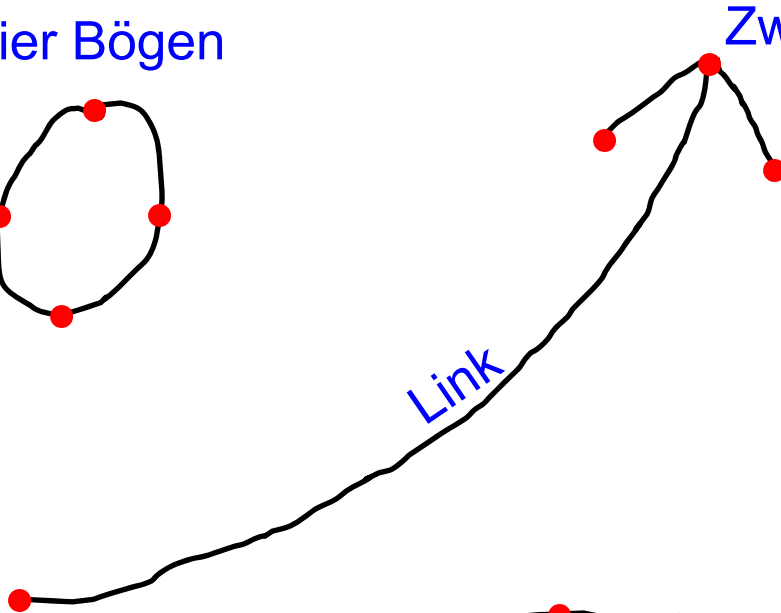
Zwei Linien



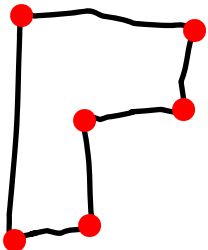
Vier Linien



Link



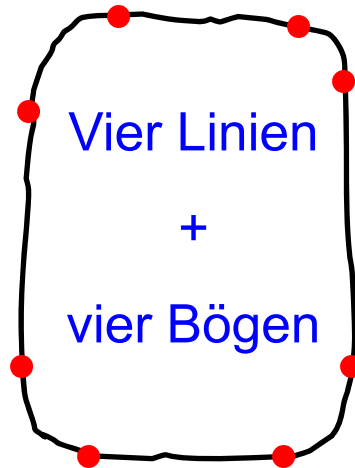
Sechs Linien



Vier Linien

+

vier Bögen

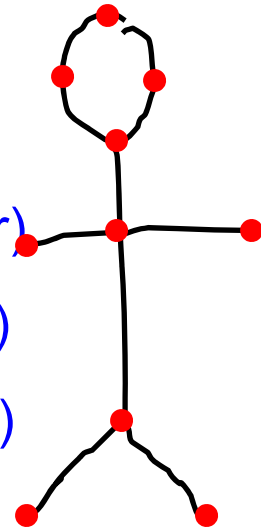


Vier Bögen (Kopf)

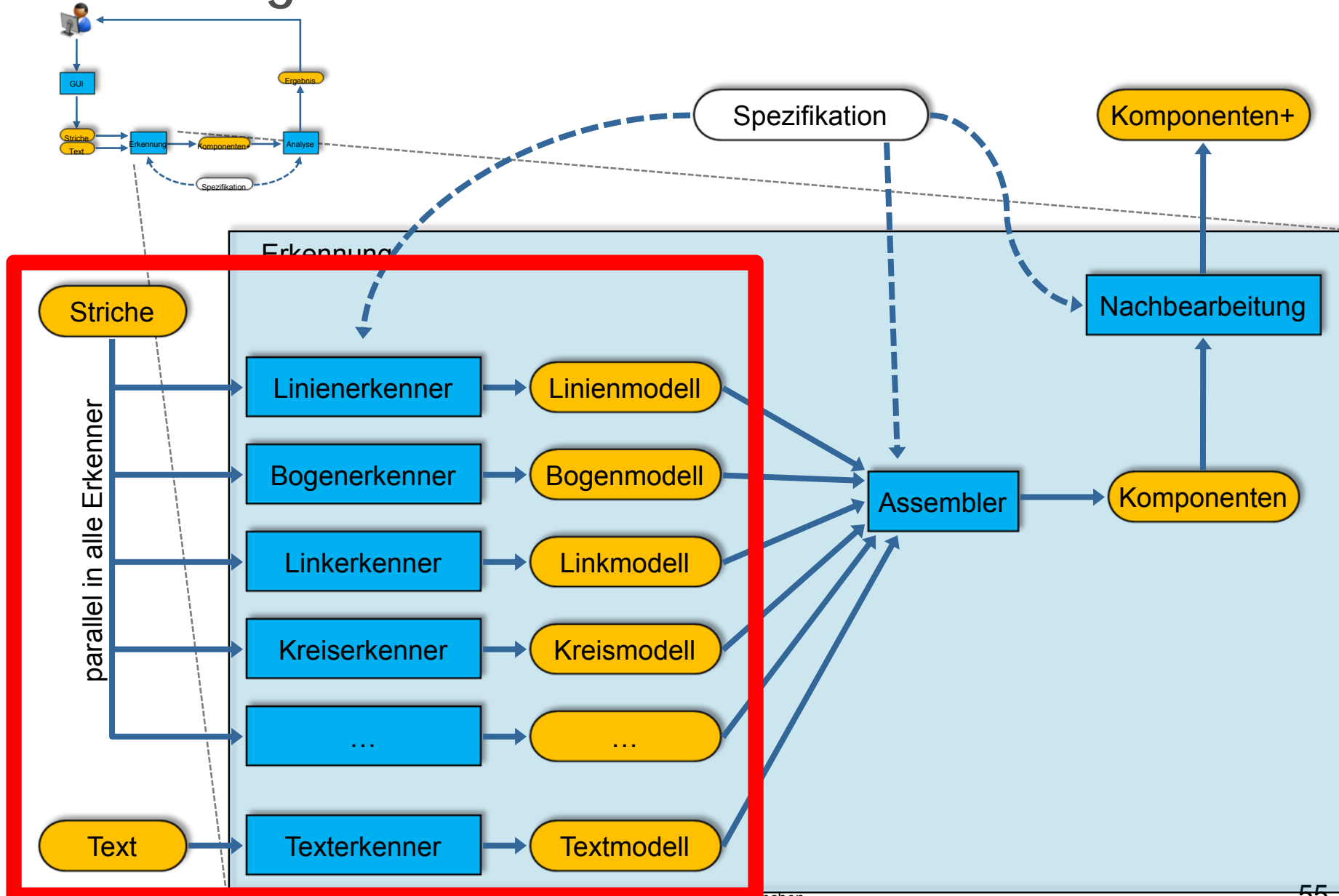
Zwei Linien (Körper)

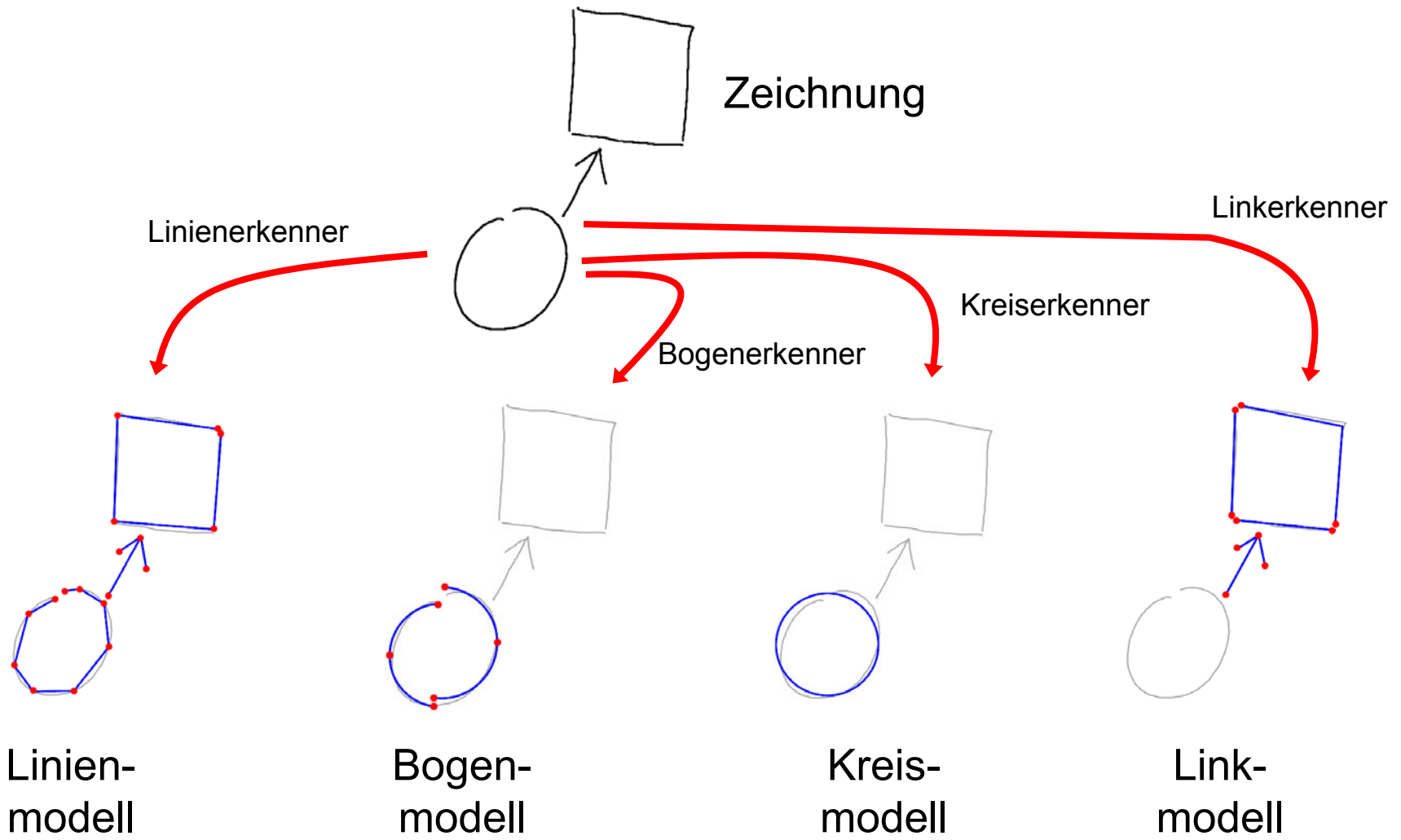
Zwei Linien (Arme)

Zwei Linien (Beine)

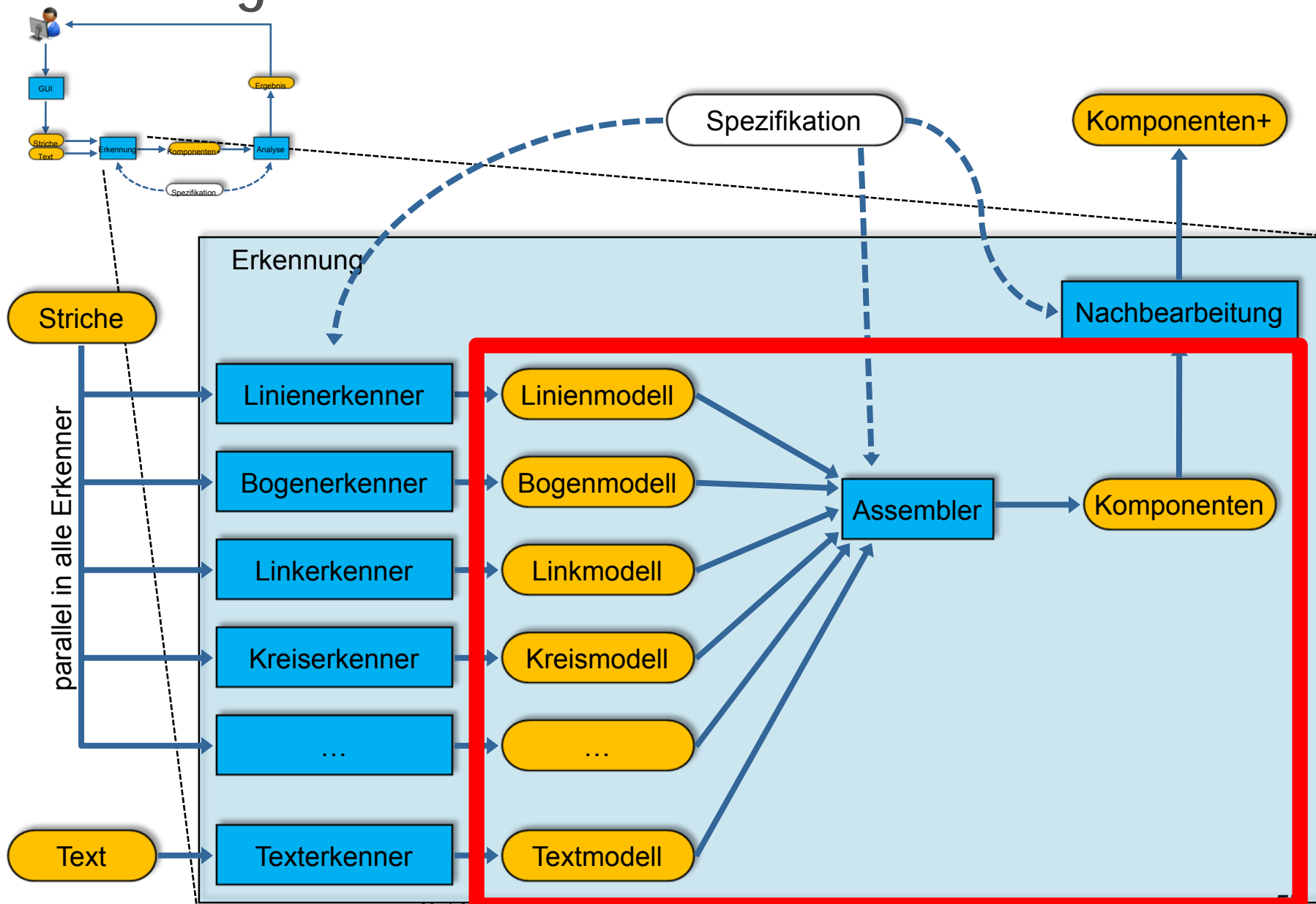


# Architektur Sketching-Editor - Erkennung

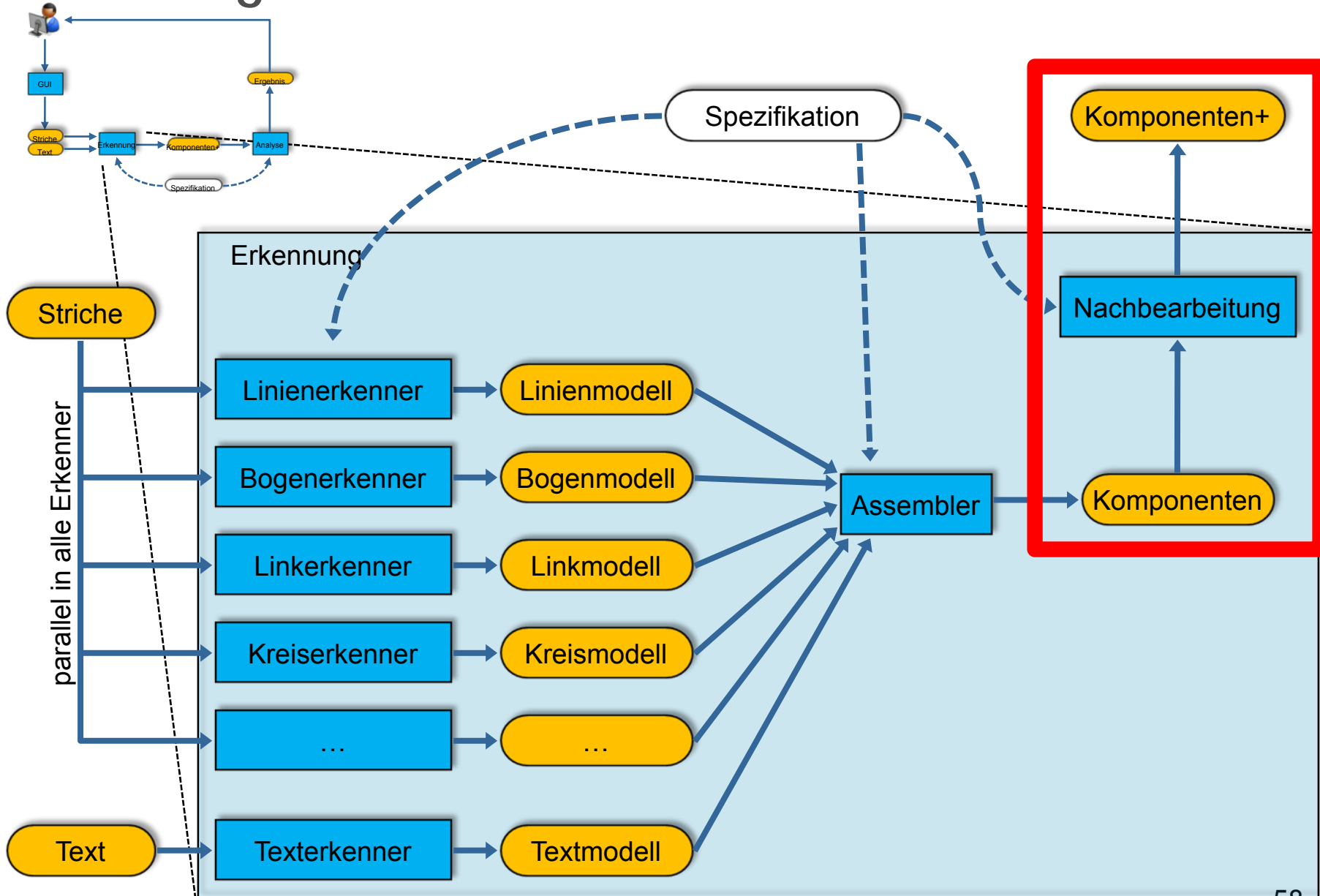




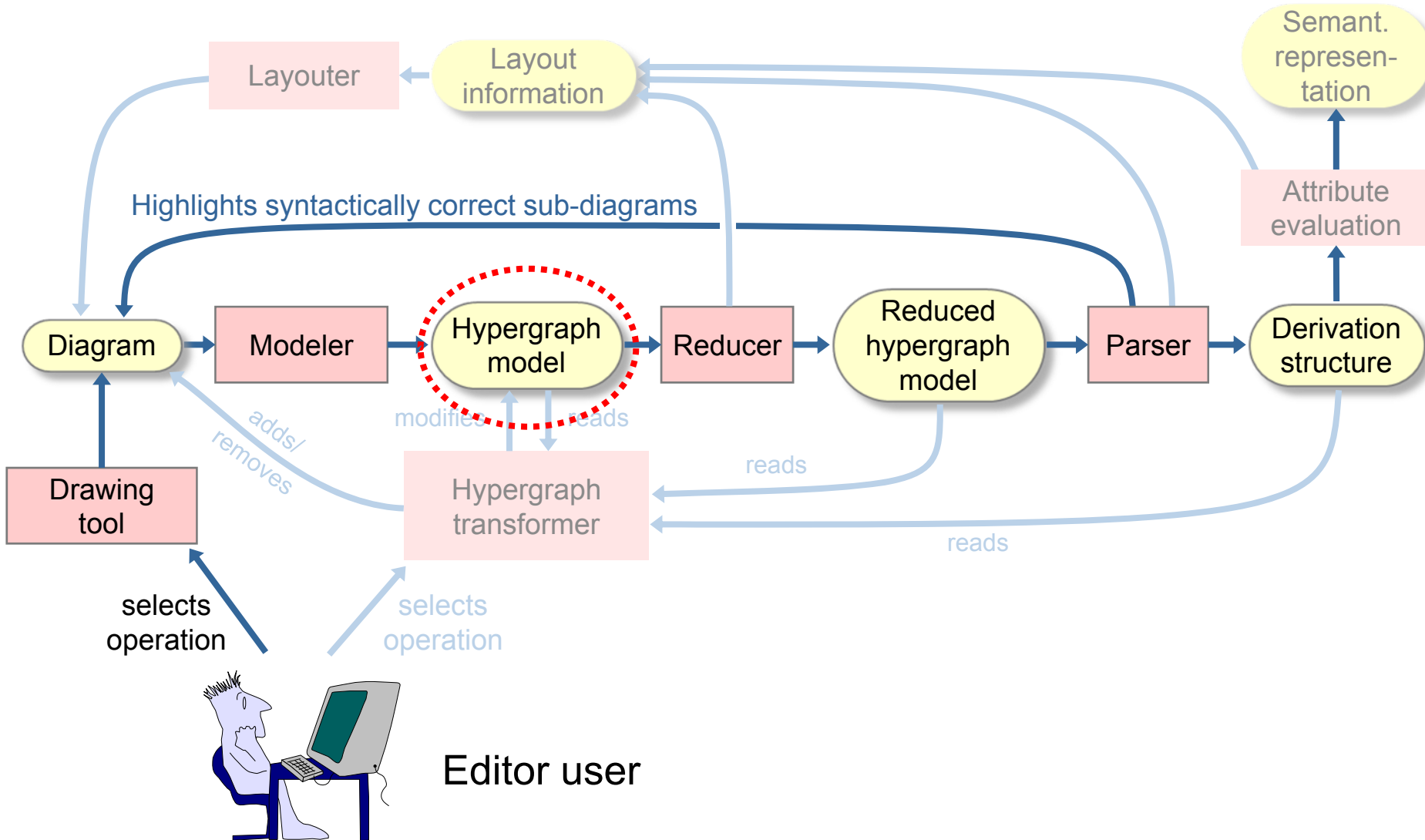
# Architektur Sketching-Editor - Erkennung



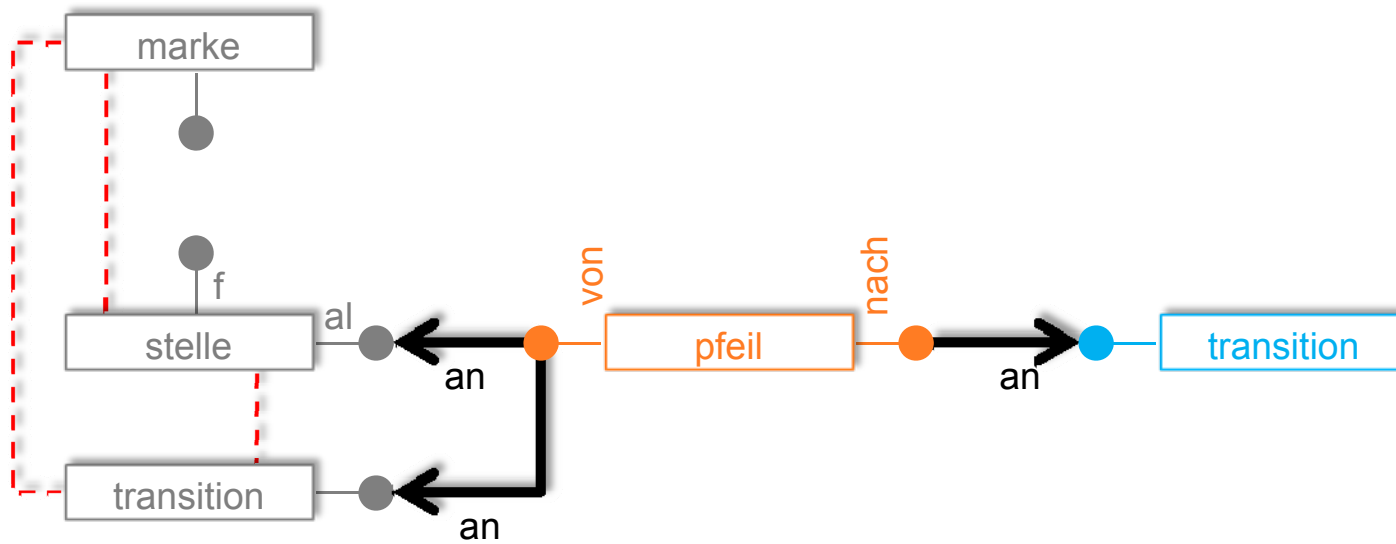
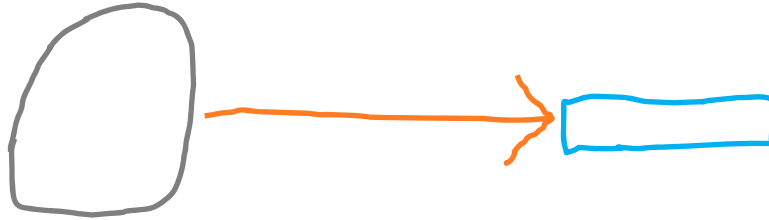
# Architektur Sketching-Editor - Erkennung



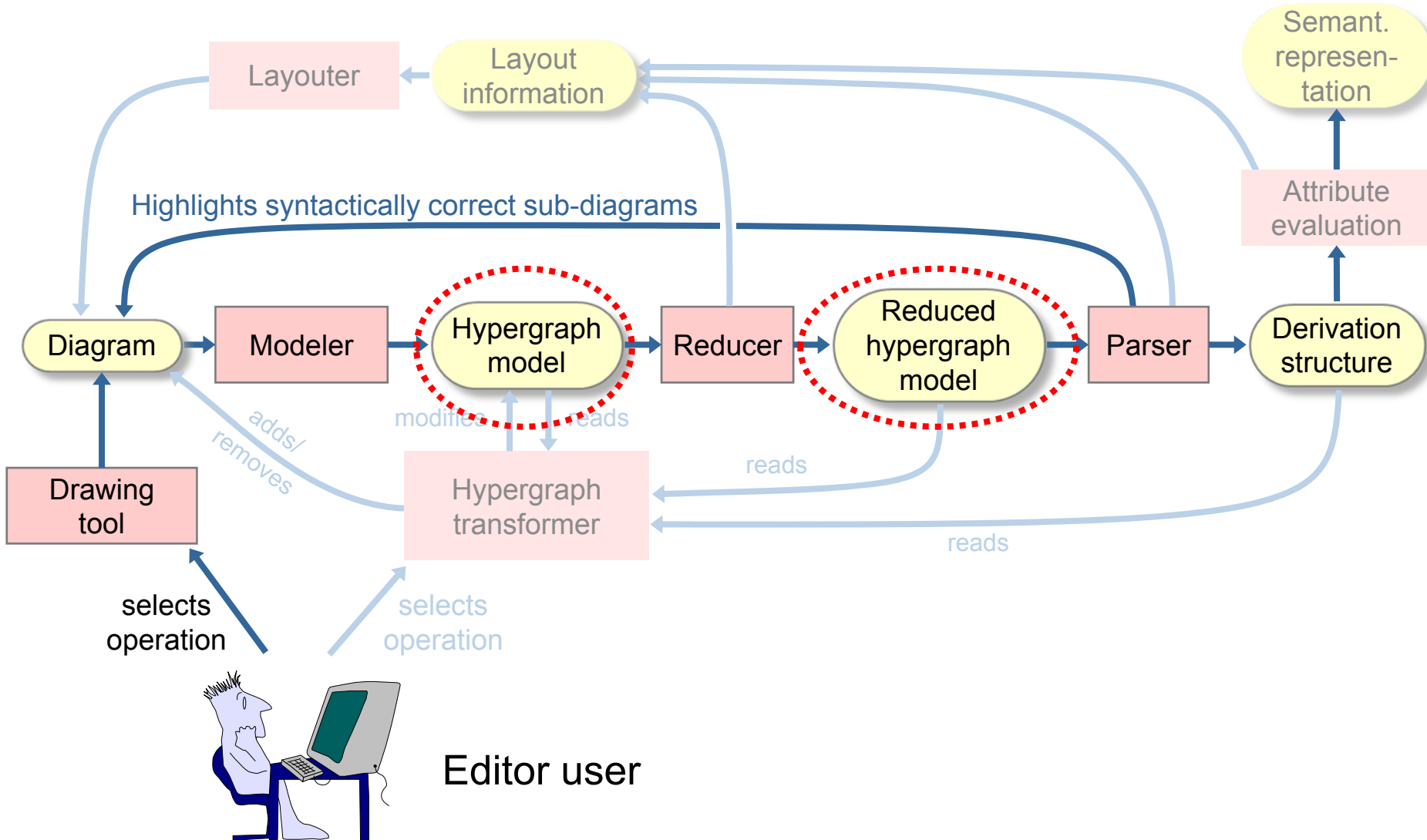
# DiaGen: Editorarchitektur



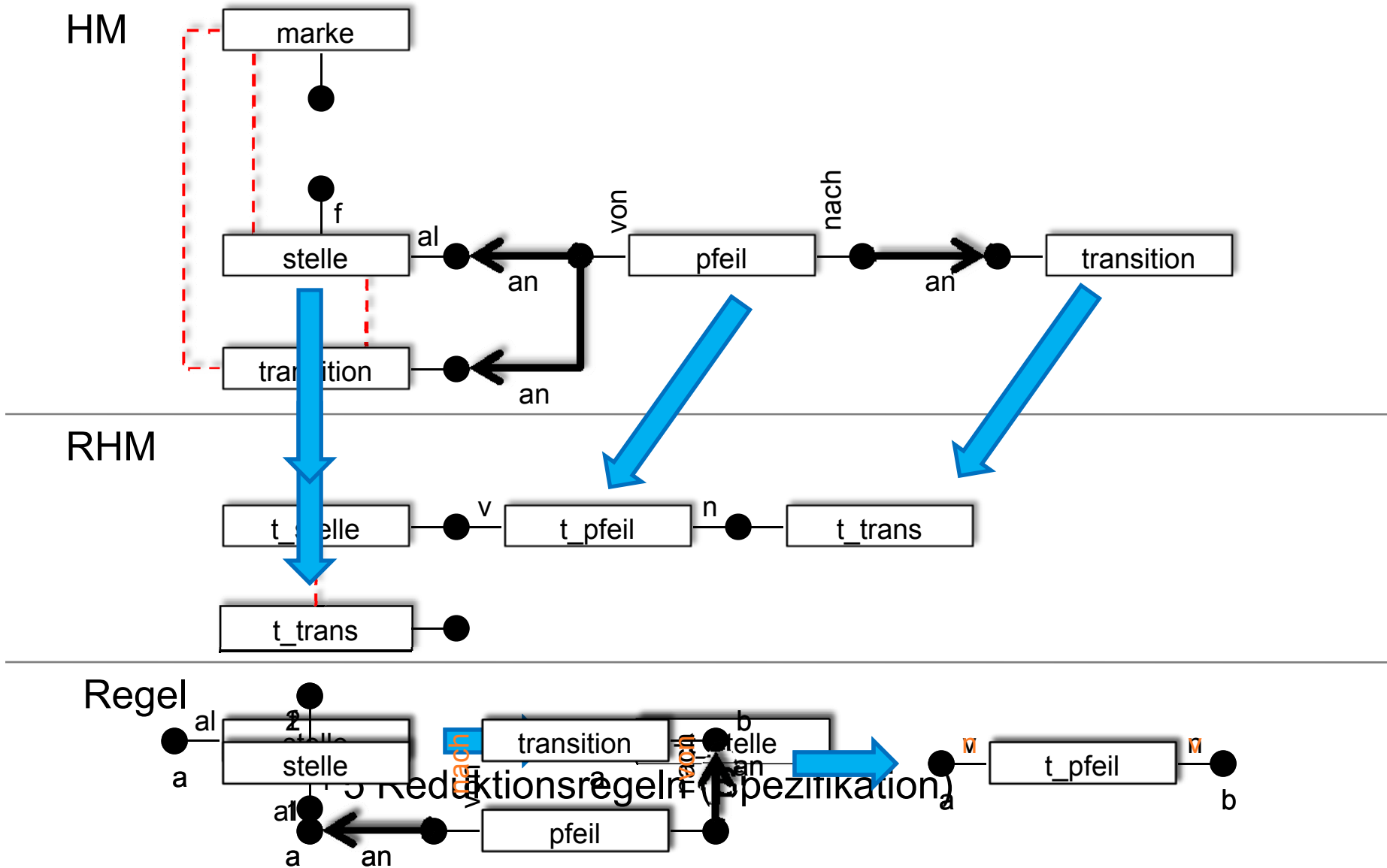
# Sketching: Hypergraph-Modelle



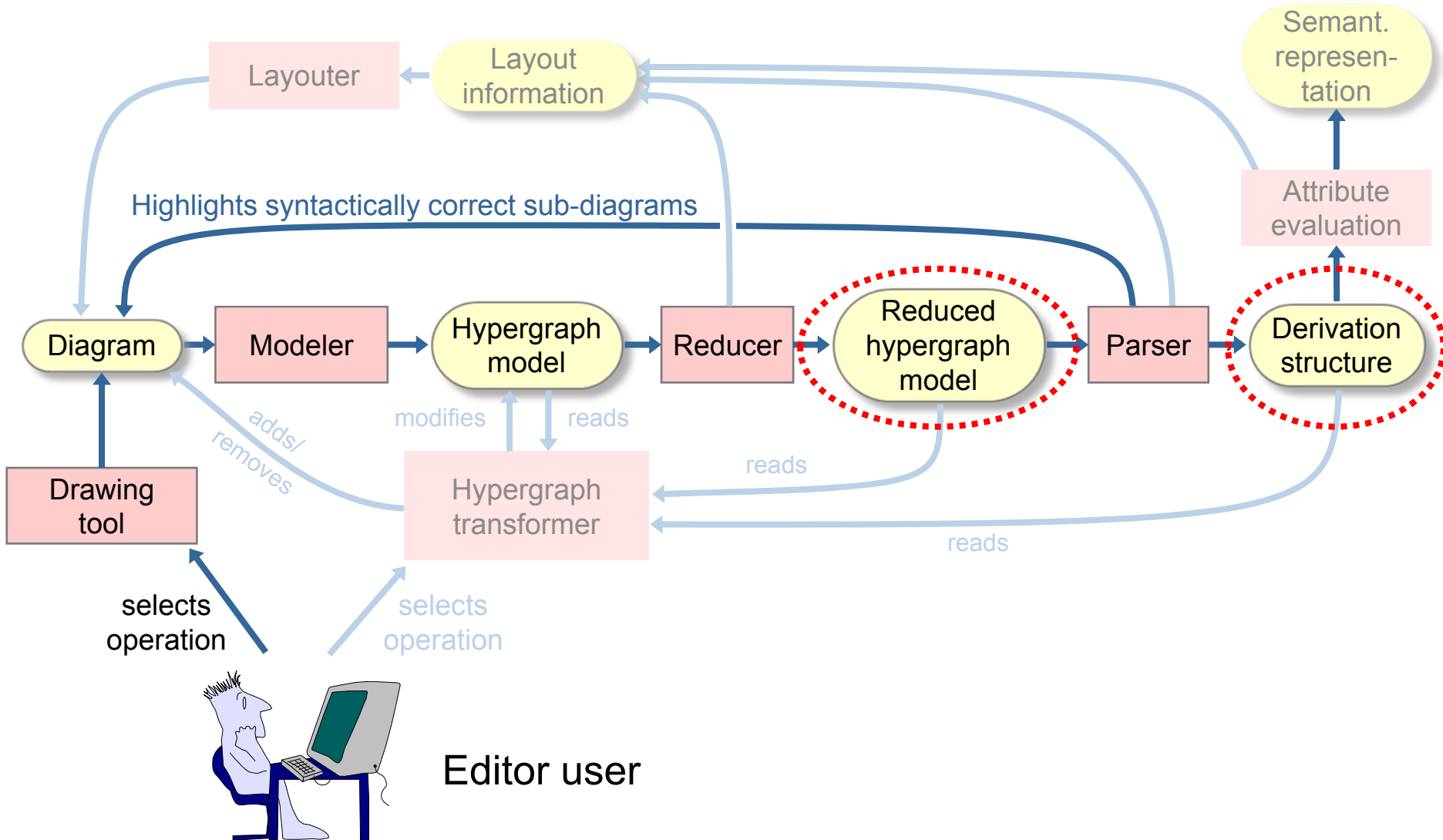
# DiaGen: Editorarchitektur



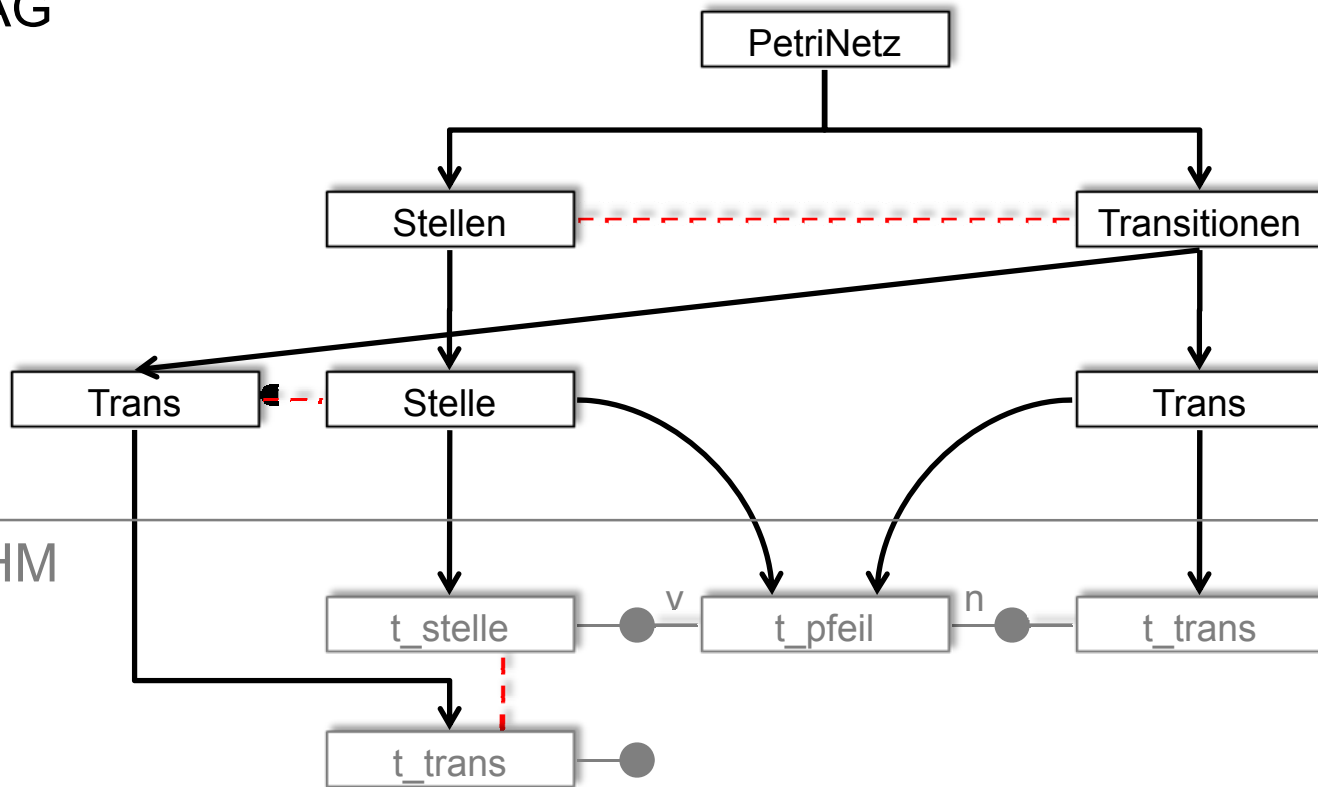
# Sketching: Reduzieren



# DiaGen: Editorarchitektur

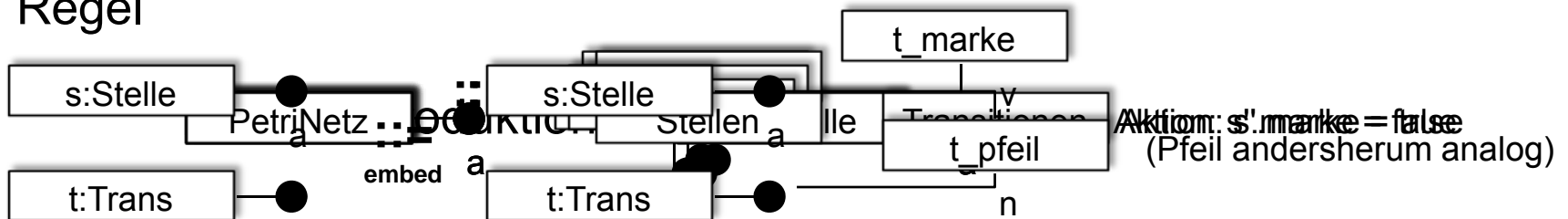


DAG



RHM

Regel



- Spezifikation & Generierung von **Diagrammeditoren**
  - Freihändiges & strukturiertes Editieren
- **Benutzerassistenz:**
  - Syntax-Unterstützung für Diagrammeditoren:
    - ▶ Diagrammvervollständigung/-korrektur
    - ▶ Generierung von Beispieldiagrammen
    - ▶ Generierung von Editieroperationen
    - ▶ generischer Ansatz, kein zusätzlicher Spezifikationsaufwand
  - Basiert auf Hypergraph-Vervollständigung
- **Sketching:**
  - Aufgrund Spezifikation vollständig generisch
  - Stellt nur wenig Anforderungen an Zeichenstil
  - Ausnutzung von Kontextinformationen