Diplomarbeit

Entwurf und Implementierung eines grafischen Editors für Aufbaustrukturpläne

Andreas K. Knöpfel
September 1995

Universität Kaiserslautern
Fachbereich Elektrotechnik
Lehrstuhl für Digitale Systeme
Prof. Dr.–Ing. S. Wendt

Diplomarbeit

Entwurf und Implementierung eines grafischen Editors für Aufbaustrukturpläne

Andreas K. Knöpfel
September 1995

Betreuer: Prof. Dr.–Ing. S. Wendt
Dipl.–Ing. Wolfram Kleis
Dipl.–Ing. Mathias Neumüller

Andreas K. Knöpfel
Wilhelm–Raabe–Str. 26 – App. 1113
67663 Kaiserslautern

Erklärung

Hiermit erkläre ich, die vorliegende Diplomarbeit ausschließlich mit Hilfe der angegebenen Quellen, sowie den Ratschlägen der Betreuer angefertigt zu haben.

(Ort, Datum) (Unterschrift)

Anhang

Einleitung

Auf der Suche nach allgemein anwendbaren, einheitlich durchgängigen Beschreibungsformen von Systemen hat der Lehrstuhl der Universität Kaiserslautern das SPIKES–Projekt ins Leben gerufen. SPIKES steht dabei als Abkürzung für 'Structure Plans for Improving Knowledge Transfer and Engineering of Systems' und umfaßt mittlerweile eine definierte Auswahl von Vorgehensweisen und Hilfsmitteln zur Beschreibung von Systemen. Steht man vor der Aufgabe komplexe Systeme verständlich beschreiben zu wollen, so wird dies in nahezu allen Fällen durch die grafische Darstellung der vorkommenden Strukturen innerhalb dieser Systeme mit Hilfe von Plänen, sogenannten Strukturplänen, wesentlich erleichtert. Zu den im Rahmen des SPIKES–Projektes verwendeten Plänen gehören Ablaufpläne (Petrinetze), Aufbaupläne (Blockdiagramme) und Entity–Relationship–Diagramme. Als Hilfsmittel zur Erstellung dieser Pläne soll eine Gruppe von grafischen Editoren zur Verfügung stehen, mit denen ein Anwender an einem Computerarbeitsplatz die entsprechenden Pläne bearbeiten kann. Hierbei soll es für jeden Plantyp einen entsprechenden Editortyp geben, der genau an die jeweiligen Anforderungen angepaßt ist. Außerdem soll eine automatische Auswertung der Planstruktur beispielsweise zur Überprüfung gewisser formaler Kriterien möglich sein.

Allen bisher bekannten Editoren zur Erstellung solcher Pläne ist gemeinsam, daß die formale Korrektheit der erstellbaren Pläne durch Vorgabe eines bestimmten Editionsstiles erreicht wird, der dem Benutzer mehr oder minder unfreiwillig aufgezwungen wird. Die Gefahr bei diesen Verfahren besteht darin, daß durch den vorgegebenen Editionsstil die Sichtweise des Planerstellenden bewußt oder unbewußt derart eingeschränkt wird, daß eventuelle bessere Darstellungen ein und derselben Struktur möglicherweise nicht erkannt werden. Im Gegensatz dazu erlauben die Editoren der SPIKES–Familie einen nahezu freien Editionsstil. Eine Überprüfung oder Auswertung der durch einen Plan beschriebenen Struktur findet erst auf expliziten Wunsch des Benutzers statt.

Thema dieser Diplomarbeit ist der Entwurf und die Implementierung eines Editors für Blockdiagramme, der dieser freien Editionsphilosophie entspricht. Als Basis für die Erstellung dieser Editoren diente Interleaf–V, eine Anwendung zur Erstellung von Dokumenten mit eingebetteten Grafiken, die über eine LISP–Schnittstelle erweitert werden kann, sowie eine darauf aufbauende bereits teilweise bestehende Erweiterung, die SPIKES–Basis, die auch von anderen Editoren der SPIKES–Familie genutzt werden kann.

Die vorliegende Arbeit läßt sich in zwei Teile gliedern:

Der erste Teil beschreibt Konzepte, die für alle Editoren der SPIKES–Familie Gültigkeit haben. Neben einer Modellierung des Editionskonzeptes gilt der Problematik bei der Bearbeitung übereinanderliegender Kanten und die Lösung dieser Problematik durch das Kantenstückkonzept der Hauptaugenmerk. Solche übereinanderliegenden Kanten können zum Beispiel in Petrinetzen, vereinzelt aber auch in Blockdiagrammen, vorkommen. Die Entwicklung und Implementierung dieses Konzeptes stellt den eigentlichen Kern dieser Diplomarbeit dar.

Der zweite Teil mit dem Titel ”Blockdiagramm–Editor ” ist ausschließlich den blockdiagramm–editor–spezifischen Aspekten gewidmet. Um eine konkrete Vorstellung von den Plänen zu gewinnen, die mit dem Blockdiagramm–Editor erstellt werden können, werden in diesem Teil zunächst die in Blockdiagrammen vorkommenden Symboltypen vorgestellt und kurz ihre Bedeutung erläutert. Daran anschließend wird die Realisierung dieser Symbole durch Grafikobjekte im Editorsystem und ihre Bearbeitung durch den Benutzer mit Hilfe des Editors geschildert. Auf allgemeingültige Editionskonzepte, die auch in anderen Editoren der SPIKES–Familie angewendet werden, wird in diesem Teil nicht mehr eingegangen. Naturgemäß fällt dieser Teil dadurch sehr viel implementierungsnäher und aber auch wesentlich kürzer als der allgemeine Teil aus.

Es wird in dieser Arbeit davon ausgegangen, daß dem Leser die verwendeten Plantypen (Blockdiagramme, Petrinetze und Entity–Relationship–Diagramme) geläufig sind. Dies ist umso wichtiger, da den Plänen in dieser Arbeit eine Doppelrolle zukommt. Zum einen stellen sie die Gebilde dar, deren Erstellung mit den Editoren erleichtert werden soll, zum anderen werden eben Pläne diesen Typs zur Beschreibung von Aufbau und Ablauf in dieser Arbeit verwendet.

Nicht erläutert wird in dieser Arbeit die Funktionsweise von Interleaf–V, solange diese nicht wesentlich die Implementierung der SPIKES–Basis–Konzepte beeinflußt, so daß einige Besonderheiten ohne Erläuterung der Interleaf–Eigenheiten nicht zu verstehen wären. Nähere Informationen zu Interleaf V findet der Leser unter [1], [2] und [3]. Ebenso ist es selbstverständlich, daß im Rahmen dieser Arbeit nicht sämtliche Aspekte der SPIKES–Basis detailliert dargestellt werden können. Auch hier sei auf die im Literaturverzeichnis aufgeführten weiterführenden Quellen verwiesen ([4], [7], [8]).

1 SPIKES–Editoren

Das erste Unterkapitel dieses allgemeinen Teiles dient dazu, dem Leser eine grobe Vorstellung über die Menge der möglichen Plantypen vermitteln. Es geht hierbei nicht um eine Beschreibung der Pläne im Hinblick auf ihre Interpretation. Vielmehr wird anhand eines allgemeinen Objektmodells, in dem alle möglichen Komponenten eines Planes und deren Beziehungen zueinander dargestellt sind, gezeigt, wie die Pläne aussehen, die mit Hilfe der SPIKES–Editoren erstellt werden können. Anschließend wird der Leser ausgehend von diesem Modell an das allgemeine SPIKES–Objekt–Modell herangeführt. Hierbei werden nacheinander alle nötigen Akteure zur Realisierung eines solchen Editorsystems eingeführt und deren Funktion kurz vorgestellt.

Vieles hiervon ist bereits in einigen Vorabspezifikationen (vgl. [3], [4] und [8]) behandelt worden und wird hier nochmals, wenn auch in gekürzter Form, dargestellt. Zum einen dient dies dem besseren Verständnis der nachfolgenden Kapitel, zum anderen berücksichtigt diese Darstellung einige inzwischen neu hinzugekommene Akteure, die durch spätere konzeptionelle Erweiterungen des Editor–Systems noch in der Implementierungsphase nötig geworden sind.

Das zweite Unterkapitel stellt das allgemeine Editionskonzept der SPIKES–Editoren vor, das die Basis aller Editionsschritte darstellt. In mehreren Schritten wird der Leser an Aufbau und Funktionsweise des SPIKES–Editor–Systems herangeführt.

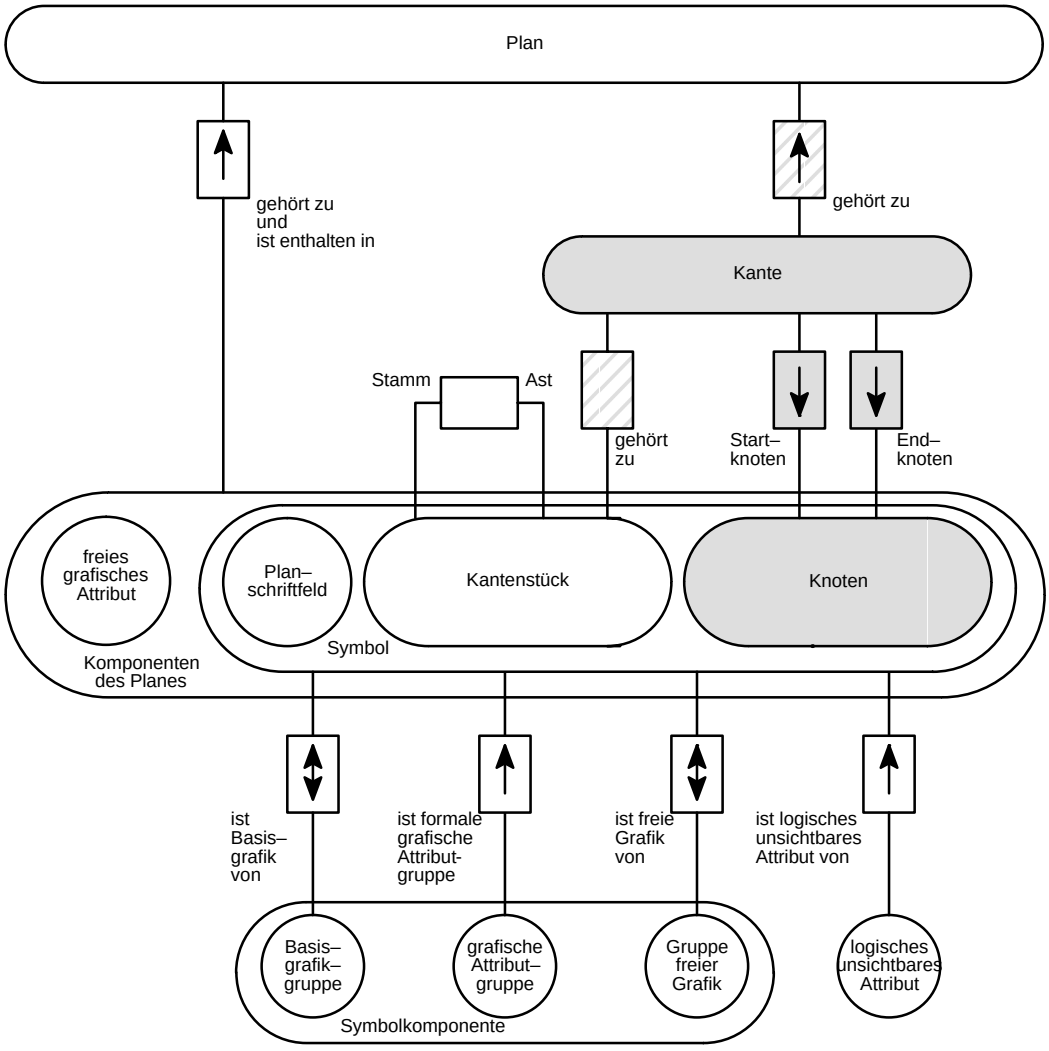
Das dritte Unterkapitel befaßt sich mit der Problematik der Behandlung von teilweise übereinanderliegenden Kanten, wie sie z.B. bei Kantenbäumen auftreten. Es werden verschiedene Konzepte als Lösungsansätze zu dieser Kantenbaumproblematik vorgestellt sowie deren Auswirkungen auf die Edition von Kanten. Entsprechend der gewünschten Editionsphilosophie wird die Auswahl des sogenannten Kantenstückkonzeptes begründet und dessen Integration in das Editor–System dargestellt.

Das letzte Kapitel dieses allgemeinen SPIKES–Teiles beschreibt die Aufgabenbereiche des Planeditors, die wesentlich von der Kantenbaumproblematik betroffen sind. Zu diesen gehören u.a. das Ausschneiden von Grafikobjekten aus einem Plan sowie das anschließende Einfügen dieser Grafikobjekte in denselben oder einen anderen Plan als auch das Aufheben von Editionsschritten. Letzteres ist dargestellt in dem sogenannten ”Undo–Konzept”.

1.1 Allgemeines SPIKES–Objektmodell

1.1.1 Die Datenstruktur in SPIKES–Plänen

Bei allen Plänen, die mit den Editoren der SPIKES–Familie erstellbar sein sollen, handelt es sich um Strukturpläne, das heißt um formale Grafiken, die Strukturen darstellen. Unter *formalen* Grafiken sollen hierbei all diejenigen grafischen Strukturen verstanden werden, zu denen es eine eindeutig definierte Interpretationsvorschrift gibt. Bild–1 zeigt ein Objektmodell, das für alle Strukturpläne, die mit Hilfe der SPIKES–Editoren erstellbar sein sollen, gültig ist[1].

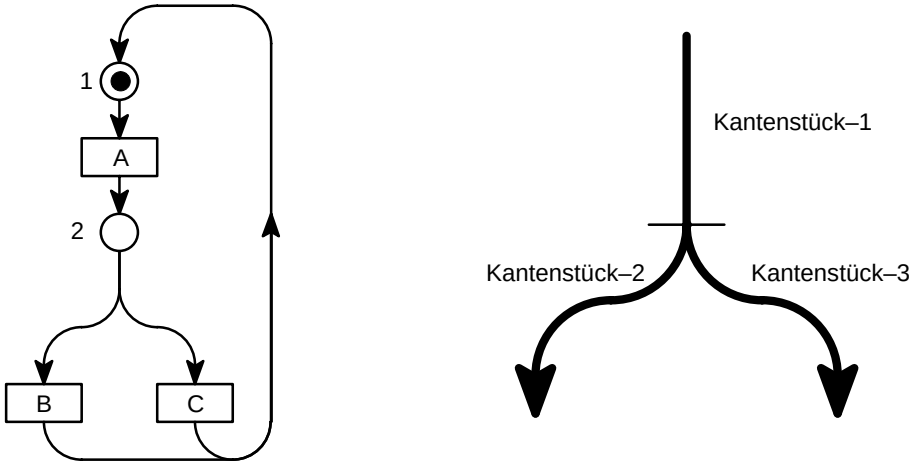


Bild–1: Objektmodell der SPIKES–Pläne

1. Die im allgemeinen SPIKES–Objektmodell entsprechend der Spezifikation nach [4] möglichen Anschlußgrafiken sind in diesem Modell nicht gezeigt und werden auch im folgenden in dieser Arbeit nicht berücksichtigt. Bei den Anschlußgrafiken handelt es sich um fest definierte Kontaktpunkte bestimmter Knotentypen zum Anschluß von Kanten, wie z.B. bei den Partitionsknoten in Entity–Realationship–Diagrammen. Da die Anschlußgrafiken einerseits für die in dieser Arbeit beschriebenen allgemeinen SPIKES–Konzepte nicht relevant sind und sie andererseits beim Blockdiagrammeditor überhaupt nicht zur Anwendung kommen, stellt diese reduzierte Sichtweise keine Einschränkung dar.

Definitionsgemäß besteht eine Struktur (vgl. [5]) aus einer oder mehreren Mengen von Objekten und einer oder mehreren Relationen, die diese Objekte zueinander in Beziehung setzen. In einem Strukturplan werden alle Objekte abgebildet auf flächenartige Symbole, die Knotensymbole oder kurz Knoten genannt werden. Einer Beziehung B aus einer der vorgegebenen Relationen zwischen zwei Objekten O_1 und O_2 entspricht ein linienartiges Kantensymbol, kurz Kante, das die Knoten zu O_1 und O_2 eindeutig miteinander verbindet. In Bild–1 wird dieser Zusammenhang dargestellt, durch die zwei grauen Entity–Knoten für Knoten und Kanten, die über die ebenfalls grau eingezeichneten Relationsknoten zueinander in Beziehung gesetzt werden. Ebenso spiegelt sich dieser Sachverhalt in der grafischen Struktur des obigen Planes wieder, da er selbst ein Strukturplan ist. Man sieht, daß jeder Kante genau ein Start– und ein Endknoten zugeordnet ist, wobei die Wahl, welcher Knoten Startknoten und welcher Endknoten ist, zunächst rein willkürlich sein darf. Eine Ausnahme stellen Enthaltenseinsrelationen dar, die grafisch auch direkt durch das Enthaltensein von Knoten ineinander zum Ausdruck gebracht werden können.

Alle Komponenten eines Planes werden bei SPIKES–Editoren durch abgeschlossene individuell erzeugbare und bearbeitbare Grafikobjekte dargestellt. Während jeder Knoten eine Komponente eines bestimmten Planes darstellt, ist dies bei den Kanten nicht der Fall. Ein Kante wird jeweils durch ein oder mehrere Kantenstücke repräsentiert. Hierzu sei als Beispiel das folgende Petrinetz (Bild–2) betrachtet.



Bild–2: Petrinetz als Beispiel eines Strukturplanes

Von Stelle 2 verläuft eine Kante zur Transition B. Eine weitere Kante verläuft von Stelle 2 zur Transition C. Beide Kanten kann man zerlegen in einen Teil, den beide Kanten gemeinsam haben und einen eigenständigen Teil. Die so gefundenen Teile werden *Kantenstücke* genannt. Das Entscheidende daran ist, daß mehrere Kanten ein *gemeinsames* Kantenstück haben können. Im Beispiel heißt dies, daß die linke Kante durch das gemeinsame Kantenstück 1 und Kantenstück 2 beschrieben wird und die rechte Kante durch das gemeinsame Kantenstück 1 und Kantenstück 3. Anstatt also jeweils eine Kante als abgeschlossenes Grafikobjekt zu realisieren, wurde bei den

SPIKES–Editoren entschieden, Kantenstücke als abgeschlossene Grafikobjekte zu realisieren. Die Gründe dafür werden in Kapitel 1.3 erläutert.

Knoten und Kantenstücke gehören zur Menge der Symbole eines Planes. Kennzeichnend für diese Symbole ist, daß sie die logische Information des Planes repräsentieren, das heißt, die Information, die formal aus der dargestellten Struktur hervorgeht. Ein weiterer Symboltyp, ist der des Planschriftfeldes. Ein solches Feld enthält den Namen des Plantyps, die Versionsnummer, sowie das Erstellungsdatum und den Namen des Planes und des Autors. Somit kann der Plan eindeutig identifiziert und an anderer Stelle auf ihn Bezug genommen werden.

Allen Symbolen in SPIKES–Plänen ist weiterhin gemeinsam, daß sie aus einer Basisgrafik, einer Reihe symboltypabhängiger, formaler, zumeist sichtbarer grafischer Attribute, sowie sonstiger freier Grafik bestehen können. Unter freier Grafik ist jegliche Art nichtformaler Grafik zu verstehen, sei es zur Verschönerung des Planes oder zu Erklärungszwecken. Auf sie wird im folgenden nicht näher eingegangen.

Die eigentliche Symbolgrafik enthält grundsätzlich eine Basisgrafikgruppe. Zusätzlich sind je nach Symboltyp mehrere formale Attributgrafikgruppen möglich. Zusammen bilden sie ein formales Symbol, für das entsprechend dem jeweiligen Plantyp eine allgemeingültige Interpretationsvorschrift existiert.

Eine Grafikgruppe ist ein Grafikobjekt, das sich dem Benutzer als abgeschlossene Einheit präsentiert und auch vom Grafikeditor als eine Einheit verwaltet wird, in der ein oder mehrere Grafikobjekte enthalten sein können (vgl. [3]). Diese enthaltenen Grafikobjekte können ihrerseits wieder Grafikgruppen sein oder aber unteilbare Grafikobjekte, wie z.B. Linien, Ellipsen oder Bögen bei Interleaf V.

Um die Trennung der Symbolgrafiken in Basisgrafikgruppe und Attributgrafikgruppen zu verstehen, betrachte man noch einmal das Bild–2. Das Petrinetz enthält zwei Stellen, dargestellt durch jeweils einen Kreis. Eine davon ist leer, die andere ist mit einer Marke belegt. Es macht keinen Sinn, eine leere Stelle und eine Stelle mit einer oder mehreren Marken als Stellvertreter unterschiedlicher Symboltypen aufzufassen. Dieses leitet sich unmittelbar aus der Definition eines Petrinetzes ab, nach der ein Petrinetz ein bipartiter Graph ist und es demnach nur zwei disjunkte Mengen von Knoten geben kann: Stellen und Transitionen. Elemente beider Mengen sollten durch Exemplare entsprechender eindeutiger Symbolklassen repräsentiert werden. Die Marken stellen demzufolge lediglich ein grafisches Attribut der Stellen dar. Dasselbe gilt in gleicher Weise auch für andere Graphen. Der Teil der Symbolgrafik, der die Mengenzugehörigkeit des Knotens oder der Kante bezüglich der Planstruktur definiert, wird zur Basisgrafik, die Erscheinung der Attribute zur Attributgrafik.

Neben diesem semantischen Grund zur Trennung von Basis– und Attributgrafik gibt es auch rein praktische Gründe. Dazu seien als Beispiel nochmals die Stellen von Petrinetzen betrachtet. Es ist erlaubt, daß eine Stelle beliebig viele Marken enthalten kann, solange dies ihre Kapazität nicht überschreitet. Würde man nicht zwischen Basisgrafik und Attributgrafik unterscheiden, so müßte es ein eigenes Symbol für leere Stellen und jeweils ein weiteres für Stellen mit einer, zwei

oder noch mehr Marken geben. Ungeachtet der Frage, bis zu welcher Anzahl von Marken diese Darstellungsform noch zweckmäßig ist, ist dieses Vorgehen auch deshalb nicht praktikabel, weil zur Abdeckung aller möglichen Fälle theoretisch unendlich viele Symbole definiert und zur Verfügung gestellt werden müßten.

Desweiteren erleichtert die Trennung von Basis– und Attributgrafiken die Einführung mehrerer ähnlicher Symboltypen. Man denke beispielsweise an gerichtete und ungerichtete Kanten. In diesem Fall ist es praktisch, als Basisgrafik für beide Kantentypen einen Linienzug vorzusehen und zusätzlich für gerichtete Kanten eine Attributgrafik für die Pfeilspitze zu definieren.
Letztlich ermöglicht diese Trennung eine unabhängige Edition von Basisgrafik und Attributgrafiken (siehe Kapitel 1.2), was die Komplexität der Werkzeuge zur Bearbeitung dieser Symbole (Editoren) erheblich reduziert.

Zusätzlich zu den sichtbaren gibt es unsichtbare Attribute. Bei diesen kann es sich um logische Attribute des Symbols, Verweise auf Stellen außerhalb des Planes oder sonstige Zusatzinformation handeln. Logische Attribute eines Symboles sind logische Planinformationen, die ausschließlich diesem Symbol zugeordnet sind. Man denke beispielsweise an die Flußzahl einer Kante innerhalb eines Petrinetzes. Solange nicht ausdrücklich anders angegeben, geht man bei Petrinetzkanten von einer Flußzahl gleich 1 aus. Auch wenn dies nirgendwo explizit im Plan erkennbar ist, so ist dieses Attribut dennoch eindeutig dem Symbol zugeordnet.
Bei Verweisen auf Stellen außerhalb des Planes könnte es sich um andere Dokumente oder um Implementierungshinweise (Programme, Code) handeln. Auf diese Art und Weise besteht beispielsweise die Möglichkeit der Unterstützung von Hyperlink–Dokumenten.

1.1.2 Symbolakteure

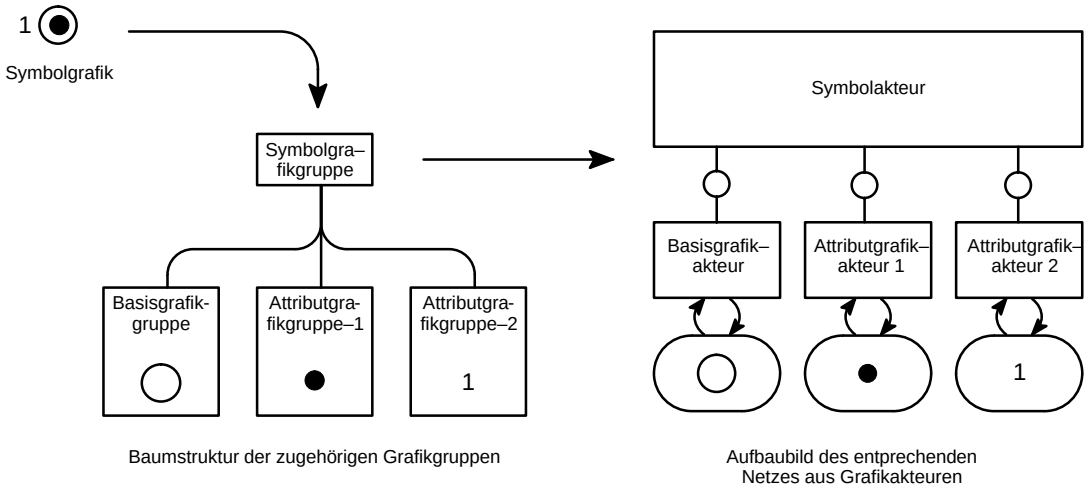
Bei grafischen Editoren für Pläne aus vielen einzelnen unterschiedlichen abgeschlossenen Grafikobjekten (im Gegensatz beispielsweise zu Pixelgrafik), ist es sinnvoll, die Planstruktur objektorientiert zu implementieren. Auf diese Art und Weise können alle zu einem Grafikobjekt gehörenden Informationen und Funktionen von anderen Objekten gekapselt einem verwaltenden Objektakteur zugewiesen werden[2].

Bei SPIKES–Editoren existiert zu jeder Grafik ein Objekt, das diese Grafik verwaltet. Diese Objekte werden im folgenden *Grafikakteure* genannt. Handelt es sich bei einem Grafik um eine Grafikgruppe, existiert hierzu ein Grafikgruppenakteur, dem die Grafikakteure der in der Gruppe enthaltenen Grafiken, die sogenannten Kinder, bekannt sind. Soll eine Grafikgruppe verändert oder gelöscht werden, so schickt man dem entsprechenden Grafikgruppenakteur eine Nachricht. Dieser wird den damit verbundenen Befehl entweder vollständig selbst ausführen oder gegebenenfalls die Botschaft an seine Kinder weiterleiten, die dann ihrerseits den (Teil–)Auftrag bear-

2. Bei der Wahl von Interleaf–V als Implementierungsplattform spielte dies eine wesentliche Rolle (vgl. [3]). Interleaf–V ist ein weitgehend objektorientiert programmiertes System zur Erstellung von Dokumenten, die Grafiken enthalten können. Es verfügt über eine LISP–Schnittstelle, über die bestehende Klassen erweitert, neue Klassen und Unterklassen definiert als auch neue Akteure erzeugt werden können. Dadurch wird das System beliebig erweiterbar.

beiten und/oder die Botschaft an eventuelle eigene Kinder weiter verschicken. Auf diesem Weg erscheint die Gruppe nach außen hin als eine Einheit.

Repräsentiert eine Grafikgruppe ein Symbol, so wird analog von *Symbolakteuren* die Rede sein. In Bild–3 ist die Implementierung einer Symbolgrafik durch einen Grafikgruppenakteur nach dem zuvor beschrieben Verfahren gezeigt. Der gezeigte Basisgrafikakteur sowie die Attributgrafikakteure stellen die Kinder des Symbolakteurs dar.



Bild–3: Implementierung einer Symbolgrafik durch Grafikakteure

Zusätzlich zu den Standardaufgaben normaler Grafikakteure, wie z.B. Farbe oder Liniendicke ändern usw., kommen den Symbolakteuren in SPIKES–Plänen weitere Aufgaben zu, wie die folgende Zusammenfassung zeigt:

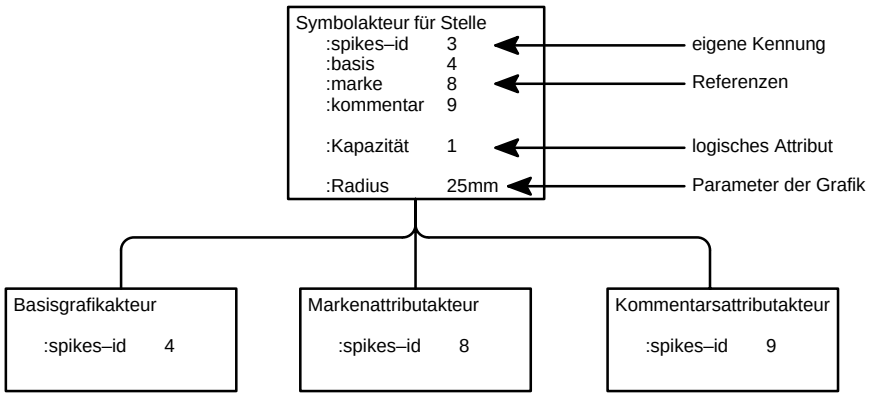
- Verwaltung der Symbolgruppe als grafische Einheit
- Zur Verfügungstellen von Operationsakteuren zur Bearbeitung der Symbolgruppe
- Verwaltung der logischen Attribute des Symbols
  (Operationsakteure zur Abfrage und Bearbeitung der logischen Attribute)
- Verwaltung von Referenzattributen zu anderen Grafikgruppen
  (Operationsakteure zum Setzen, Abfragen und Löschen von Referenzen)
- Bereitstellung sonstige Operationsakteure zur Realisierung anderer semantisch eng an das Symbol geknüpfter Aktionen (z.B. zur Ermittlung der grafischen Relationen zwischen den Symbolen zur automatischen Analsyse der Planstruktur usw.).

Die Bedeutung der Referenzattribute bedarf einer kurzen Erläuterung. Um die in Bild–1 gezeigten Relationen zu realisieren, werden zusätzliche unsichtbare Attribute eingeführt. Diese Attribute dienen als Verweise zwischen:

- Symbolakteuren und untergeordneten Basisgrafik– oder Attributgrafikakteuren
- unterschiedlichen Kantenstückakteuren innerhalb eines Kantenstückbaumes
- Knoten– und Kantenakteuren (nach einer Strukturanalyse)

Mit Hilfe dieser Verweise kann ein Symbolakteur aus der Vielzahl von Plankomponenten und Grafikakteuren für seine Basis– und Attributgrafiken die jeweiligen Relationspartner eindeutig identifizieren. Dies ist beispielsweise dann erforderlich, wenn eine bestimmte Attributgrafik bearbeitet werden soll oder auch bei der automatischen Planstrukturanalyse.

Um jeden dieser Akteure durch einen Verweis dauerhaft und eindeutig identifizieren zu können, benötigt jeder Akteur eine eindeutige dauerhafte Kennung, die im folgenden SPIKES–ID genannt wird. Um eine Relation zu realisieren, die ein Objekt A einem anderen Objekt B zuordnet, hinterlegt man bei B unter einem Schlüssel für A eine Kennung von A (siehe Bild–4). Man spricht von Referenzattributen. Wie dies im Speziellen mit Hinblick auf die Implementierungsplattform Interleaf V gelöst ist, ist bei den Aufgaben des Planakteurs im folgenden Abschnitt erläutert.



Bild–4: Zuordnung von Grafikobjekten durch Kennungen und Referenzattribute

1.1.3 Das SPIKES–Objekt–Modell

Bild–A1 zeigt das bis auf die Anschlußgrafiken, die in dieser Arbeit nicht behandelt werden[1.], vollständige SPIKES–Objekt–Modell, das aus dem zuvor vorgestellten Datenmodell für SPIKES–Pläne durch die Einbettung in das SPIKES–Editorsystem mit allen seinen Komponenten hervorgeht. Die Grafikobjekte aus Bild–1 sind hierbei durch die sie realisierenden Grafikakteure vertreten. Die Beziehungen der Grafikobjekte zueinander spiegeln sich dabei in den Beziehungen der Grafikakteure zueinander wieder. Lediglich die logischen Attribute, die entsprechend Bild–1 einem Symbol zugeordnet werden können, sind nicht mehr aufgeführt, da sie nun, wie in Kapitel 1.1.2 beschrieben, Teil des lokalen Speicherinhaltes eines Symbolakteurs sind.

Ein SPIKES–Plan stellt einen Grafikrahmen eines bestimmten Typs innerhalb einer beliebigen Folge aus Textzeichen, weiteren Grafikrahmen, Querverweisen usw. dar, die zusammen ein Dokument bilden. Jeder Grafikrahmen wird von einem Grafikrahmenakteur verwaltet, der die Aufgabe hat, ähnlich wie ein einfacher Grafikgruppenakteur, die ganze in ihm enthaltene Grafik, dargestellt durch zum Teil viele einzelne Grafikobjekte, nach außen hin als eine Einheit erscheinen zu lassen. Darüber hinaus verwaltet er eine Reihe von Attributen wie: Rahmengröße, Positionierung und Editionseinstellungen (Gitter, verwandte Maßeinheiten usw.). Ein Planakteur ist ein

erweiterter Grafikrahmenakteur, der für die Verwaltung eines SPIKES–Planes zuständig ist. In dieser Funktion muß ein Planakteur eine Reihe zusätzlicher Aufgaben erfüllen:

- Verwaltung der Planattribute: Plantyp, Planversion, Planname, Erstellungsdatum usw. Diese Attribute können mit Hilfe eines Formularakteurs bearbeitet werden.[3.]
- Versions– und Typkontrolle: Soll ein SPIKES–Plan eines bestimmten Typs und einer bestimmten Version (z.B. Typ: Blockdiagramm, Version 1.0) zur Bearbeitung geöffnet oder geladen werden, so muß die Editionsumgebung diesen Typ und diese Version unterstützen. Ist dies nicht der Fall, so wird der Anwender davon informiert und gegebenenfalls die Bearbeitung abgebrochen.
- Laden und Speichern: Soll ein Dokument abgespeichert werden, so bekommen alle Grafikrahmen innerhalb dieses Dokumentes eine Nachricht geschickt, die sie auffordert, ihre Daten in eine angegebene Dokumentdatei zu schreiben. SPIKES–Plan–Akteure müssen dafür sorgen, daß alle logischen Informationen des Planes, sowie die Information über die Klassenzugehörigkeit der jeweiligen Symbolverwalter mit abgespeichert werden. Beim Laden sind diese zusätzlichen Eigenschaften wiederherzustellen (vorausgesetzt, die Versionskontrolle war erfolgreich).

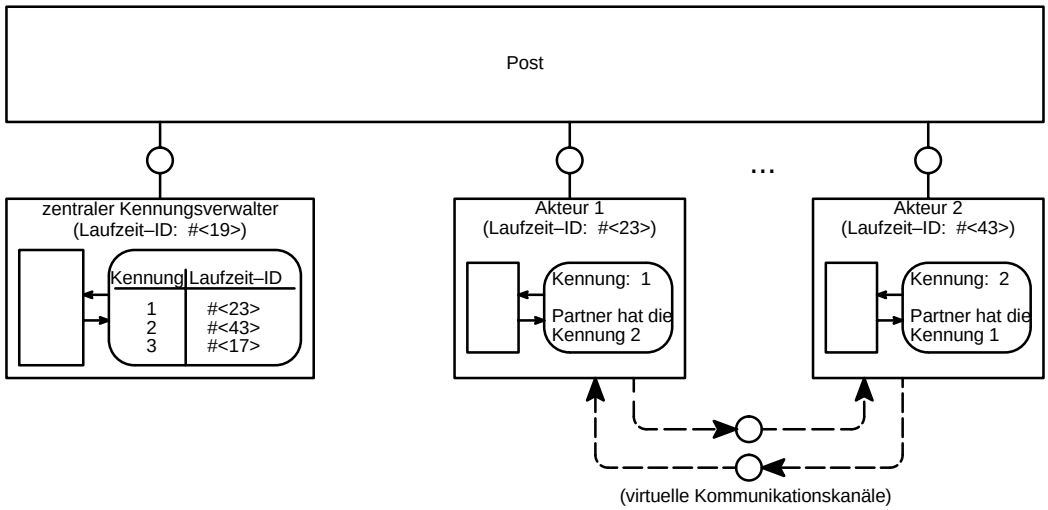
Wie bereits erwähnt verfügt jeder SPIKES–Grafikakteur (Symbolakteure, sowie deren Untergruppenakteure) über eine eindeutige dauerhafte Kennung, mit Hilfe derer dauerhafte Referenzen zwischen SPIKES–Grafikakteuren hergestellt werden können.

Nun verfügen alle Akteure in objektorientierten Systemen über eine ID, über die sie angesprochen werden können. Leider ist diese ID bei den meisten Systemen (so auch bei Interleaf V) laufzeitabhängig und absolut willkürlich. Das heißt, wird ein Dokument gespeichert, geschlossen und anschließend wieder geöffnet, werden alle dem Objekt zugehörigen Objekte wieder angelegt. Allerdings haben sie jetzt mit großer Sicherheit eine andere ID. In Interleaf–V kommt erschwerend hinzu, daß es einige Grafikoperationen gibt, bei denen ein Grafikobjekt durch eine Kopie ersetzt wird (wie z.B. bei der Untergruppenbearbeitung), obwohl logisch nur eine Änderung des ursprünglichen Grafikobjektes beabsichtet war. Obwohl es sich also logisch immer noch um ein und dasselbe Objekt handelt, hat sich durch die Bearbeitung scheinbar dessen Laufzeit–ID verändert. In welchen Fällen dies passiert, ist leider undokumentiert. Aufgrund dieser Tatsache sind diese IDs nicht zum Realisieren dauerhafter Referenzen geeignet.

Als Lösung bietet sich folgender Weg an: Jeder SPIKES–Akteur kann zusätzlich zu seiner Laufzeit–ID eine dauerhafte Kennung erhalten. Dazu muß ein zentraler Kennungsverwalter existieren.

Dieser Kennungsverwalter verfügt über eine lokale Tabelle, in der paarweise jeweils eine dauerhafte Kennung und die dazu gehörende aktuelle Laufzeit–ID abgelegt ist. Dieser Kennungsver-

3. Die Planattribute entsprechen sinngemäß dem Inhalt des Planschriftfeldes, welcher bislang noch nicht als Symboltyp unterstützt wird. Es wäre zum einen denkbar, daß der Planschriftfeldakteur, falls ein solcher im Plan existiert, die Werte direkt vom Plan übernimmt. Es könnte aber auch sinnvoll sein, die Planattribute einer formalen Auswertung vorzubehalten und den Inhalt des Planschriftfeldes frei gestaltbar zu lassen.



Bild–5: Der Kennungsverwalter
(vgl. [8])

walter muß allgemein bekannt sein. Um für einen SPIKES–Akteur eine dauerhafte Kennung zu erhalten, schickt man dem Kennungsverwalter eine Anforderungsbotschaft mit der Laufzeit–ID des SPIKES–Akteurs als Parameter. Der Kennungsverwalter bestimmt daraufhin eine einmalige Kennung, die der Interessent übermittelt bekommt. Ein Weg eindeutige Kennungen zu finden, besteht darin, einfach fortlaufende ganze Zahlen zu vergeben.
Um eine Referenz aufzubauen, wird unter einem Schlüssel für das zu referenzierende Objekt dessen ID hinterlegt, wie im vorausgehenden Kapitel beschrieben. Abweichend handelt es sich jetzt aber nicht mehr um die ID, unter der das Objekt dem ganzen System bekannt ist, sondern um eine übersetzungsbedürftige Kennung.

Beim Speichern des Dokumentes wird die eigene SPIKES–ID, sowie alle Referenzen zusammen mit allen übrigen Daten eines SPIKES–Grafik–Akteurs gesichert. Nach Schließen und anschließendem Öffnen des Planes werden alle Akteure neu angelegt. Dabei muß ebenfalls die Referenztabelle erneut angelegt werden[4].

Ein abschließendes Beispiel soll deutlich machen, wie sich dieser Unterschied beim Arbeiten mit Referenzen bemerkbar macht. Dazu betrachte man Bild–5. Akteur 1 hat in seinem lokalen Speicher eine Referenz auf Akteur 2 vermerkt und möchte diesem eine Nachricht schicken. Da Akteur 1 nicht die Adresse von Akteur 2 kennt, sondern nur dessen Kennung, schickt Akteur 1 zunächst ein Nachricht an den Kennungsverwalter mit der Bitte um Übersetzung der Kennung in die dazu gehörende Adresse. Erst nachdem Akteur 1 diese erhalten hat, kann er die eigentliche Nachricht an Akteur–2 versenden. Dieser Mehraufwand benötigt selbstverständlich Zeit und man sollte bei der Implementierung darauf achten, diesen so gering wie möglich zu halten, insbesondere, wenn starker Gebrauch von Referenzen gemacht wird.

4. Es ist durchaus zulässig, daß die Objekte nach dem erneuten Laden eine andere Kennung zugewiesen bekommen, als die, die ihnen vor dem Abspeichern zugewiesen war. Aber es ist sichergestellt, daß mit Hilfe der gespeicherten Informationen alle Referenzen wieder korrekt hergestellt werden können. Auf Probleme, die sich durch das Ausschneiden und Einfügen in andere SPIKES–Pläne und Dokumente ergeben und welche Konsequenzen dies für das ID–Konzept hat, wird in Kapitel 1.4.2.3 eingegangen.

Da der Planakteur, wie zuvor erwähnt, alle in ihm enthaltenen Grafikobjekte im Sinne eines Grafikgruppenakteurs verwaltet und ihm daher alle Laufzeit–IDs dieser Grafikobjekte bekannt sein müssen, ist es sinnvoll, den Kennungsverwalter, der ebenfalls diese Information benötigt, als Teil, d.h. Operationsakteur, des Planakteurs anzusehen bzw. zu realisieren.

Allen bisher eingeführten Objekten ist gemeinsam, daß es ihr eigentliches Wesen ist, Träger grafischer Information zu sein. Ein Dokument dient in erster Linie dazu, Menschen die in diesem Dokument enthaltenen Informationen nahe zu bringen.

Im Gegensatz dazu dienen die nachfolgend eingeführten Objekte dazu, ein solches Dokument zu erstellen, zu bearbeiten und zu prüfen (in Bild–A1 sind diese farbig eingezeichnet). Um einen schnellen Überblick über die Vielzahl dieser Objekte zu gewinnen, soll eine Einteilung in vier Gruppen vorgenommen werden, innerhalb derer die neuen Objekte vorgestellt werden:

1) Die erste Gruppe enthält die Akteure, die das Repertoire und Aussehen der zu erzeugenden Symbole innerhalb eines Planes vorgeben.

   Das Repertoire ist bestimmt durch die plantypspezifische Menge der möglichen Symboltypen. So sind in einem Petri–Netz beispielsweise nur Stellen, Transitionen und gerichtete Kantenstücke als Symbole zulässig. Soll ein Exemplar einer bestimmten *Symbolklasse* erzeugt werden, so muß diese Klasse innerhalb des Planes definiert sein. An den *Symbolklassenakteur*, der die jeweilige Klasse im System repräsentiert, muß dazu ein Auftrag zum Erzeugen eines Exemplares geschickt werden.

   Das Aussehen der Symbole ist im wesentlichen geprägt durch den jeweiligen Symboltyp. Trotzdem gibt es eine gewisse Anzahl von Größen, die sehr wohl das Aussehen, nicht aber die Interpretation der Symbole beeinflussen. Zu diesen Größen gehören u.a. Größe, Liniendicke, Füllung und Farbe einer Symbolkomponente, die individuell festlegbar sind. Damit der Anwender die Möglichkeit hat, für das Erzeugen von Symbolen bestimmte Vorgaben zu machen und nicht gezwungen ist, alle Symbole nachzubearbeiten, gibt es sogenannte Defaultwertverwalter, auch *Master*[5.] genannt. Von diesen werden die Voreinstellungen beim Erzeugen von Symbolen erfragt und das Aussehen entsprechend automatisch angepaßt.

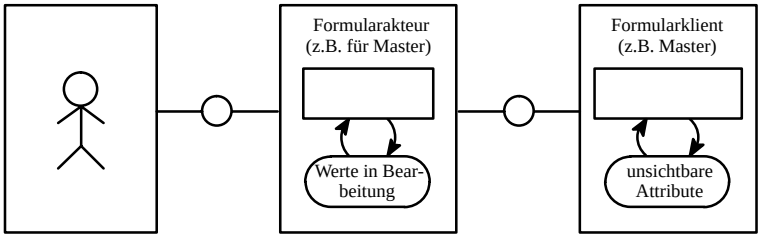
5. SPIKES–Masterobjekte dürfen nicht mit den Interleaf–Masterobjekten verwechselt werden. Zwar sind beide als Grafikgruppen implementiert, aber sie unterscheiden sich grundlegend in ihrer Anwendung. SPIKES–Masterobjekte verwalten nur einzelne abfragbare Werte, die eine Beschreibung des Aussehens liefern. Dem gegenüber sind Interleaf–Master–Objekte vollwertige unsichtbare Grafikgruppen, deren Inhalt als Vorlage zum Erzeugen von Kopien dient, die dann in das Dokument eingefügt werden. Die Entscheidung SPIKES–Masterobjekte als Interleaf–Grafikgruppen zu implementieren, ist dadurch begründet, daß sie als Grafikgruppen automatisch die Fähigkeit zum Speichern von (dadurch dauerhaften) Datenattributen erben (vgl. [3]). Dies bewirkt, daß alle Voreinstellungen über das Speichern hinaus wirksam bleiben, was auf jeden Fall wünschenswert ist.

Ursprünglich (siehe [4]) war geplant gewesen, für jede Symbolklasse innerhalb eines Planes einen eigenen zuständigen Defaultwertverwalter zu haben. Dieses an sich sehr flexible Konzept wurde aufgegeben, weil es sich gezeigt hat, daß es für den Anwender im Falle von Plänen mit einer Reihe verwandter Symboltypen eher die Arbeit vergrößert denn erleichtert. Als Beispiel seien die Kantenstücke von Blockdiagrammen betrachtet. In Blockdiagrammen sind drei Klassen von Kanten(stücken) zu unterscheiden. Es gibt gerichtete, ungerichtete und modifizierende Kanten. Höchstwahrscheinlich möchte der Anwender für alle Exemplare dieser drei Klassen die gleiche Liniendicke vorgeben. In diesem Falle wäre es höchst ärgerlich, nacheinander für alle drei Symbolklassen diese Einstellung vornehmen zu müssen, anstatt dies einmal zentral erledigen zu können. Aus diesem Grund wurde zu Lasten der Voreinstellungsvielfalt und zu Gunsten der Anwenderfreundlichkeit entschieden, es zuzulassen, daß ein Defaultwertverwalter die Voreinstellungen für mehrere Symbolklassen verwaltet.

Um für unterschiedliche Pläne unterschiedliche Voreinstellungen zu ermöglichen, müssen jedem Plan eigene Defaultwertverwalter zugeordnet sein. Diese Verwalter werden ihrerseits unter symbolischen Namen vom Planakteur verwaltet, die ebenfalls allen möglichen Interessenten für die jeweiligen Defaultwerte bekannt sind. Um eine Voreinstellung zu erfragen oder zu ändern, muß der Interessent oder Auftraggeber beim Planakteur unter dem symbolischen Namen die eigentliche Adresse (Laufzeit–ID) des betreffenden Defaultwertverwalters erfragen. Erst dann kann er mit dem Master direkt kommunizieren.

2) Bei den Defaultwerten handelt es sich, ebenso wie bei den logischen Attributen des Planes oder der Symbolakteure um unsichtbare Attribute. Um diese unsichtbaren Attribute bearbeiten zu können, sind spezielle Akteure nötig, die *Formularakteure* oder auch *Eigenschaftenblatt*(–akteure)[6]. genannt werden sollen. Wie in Bild–6 gezeigt ist, stellt der Formularakteur dabei das Benutzerinterface zur Beabeitung der gewünschtenWerte des Formularklienten zur Verfügung. Für den Benutzer stellt sich dieses Interface in Form eines sogenannten Eigenschaftenblattes dar. Ein solches Eigenschaftenblatt besteht aus einer Anzahl von Feldern, in denen änderbare Werte eingetragen sind. Die Art und Anzahl der Felder ist vom Typ des Eigenschaftenblattes abhängig.
Beim Öffnen eines Eigenschaftenblattes erfragt der Formularakteur die Werte aller so bearbeitbaren Attribute bei dem entsprechenden SPIKES–Akteur und stellt diese

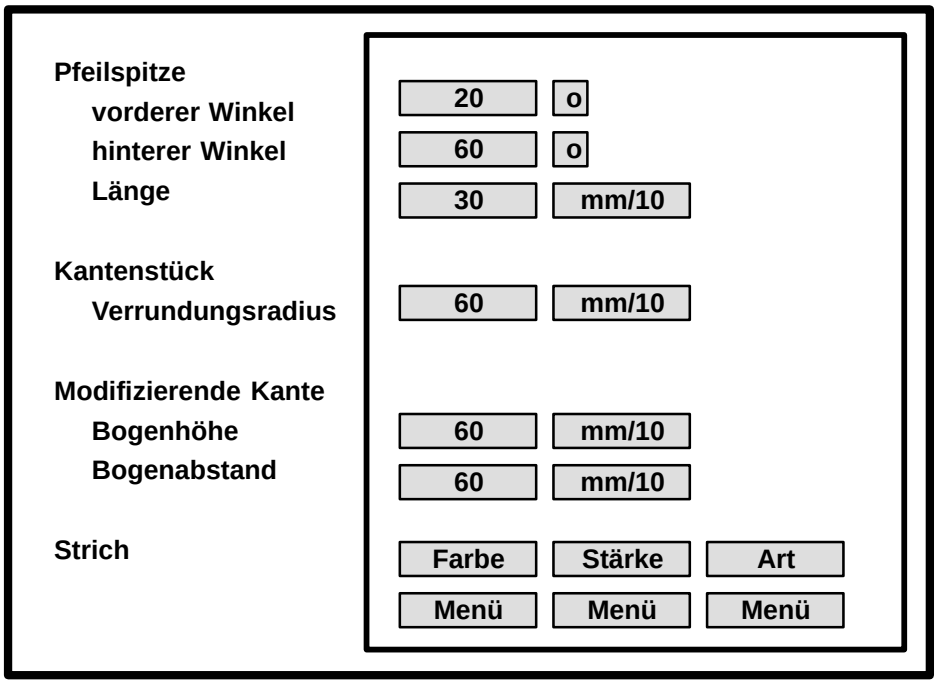
6. Interleaf V stellt bereits Mechanismen zur Erzeugung solcher Formularakteure unter dem Namen '*property sheets*' zu Verfügung. Dabei können mehrere Eigenschaftenblätter (prop–submenu's) als eine Einheit (prop–menu) verwaltet werden. Öffnet der Anwender ein Formular zur Bearbeitung, erscheint ein Fenster, an dessen oberem Rand ihm eine Auswahl möglicher Eigenschaftenblätter angezeigt wird. Das jeweilige aktuell zu bearbeitende Eigenschaftenblatt erscheint darunter. Die Implementierung der SPIKES–Formularakteure nutzt diese Mechanismen aus (vgl. Anhang C).



Bild–6: Formularakteur und Klient

in einem Fenster dar. Anschließend kann der Benutzer diese Werte bearbeiten, wobei der Formularakteur gewisse Grenzen bezüglich des Wertebereiches vorgeben kann. Nach Abschluß aller Änderungen können die neuen Werte wahlweise durch Schließen des Eigenschaftenblattes an den SPIKES–Akteur zurückgegeben werden oder aber durch Abbrechen der Bearbeitung verworfen werden.

Damit ein Formularakteur weiß, welche Attribute bearbeitbar sind, existieren verschiedene Typen dieser Akteure, von denen jeweils einer eindeutig einem bestimmten Typ von Symbol–Akteur, Defaultverwalter oder SPIKES–Plan zugeordnet ist.



Bild–7: Beispiel eines Eigenschaftenblattes

In Bild–7 ist gezeigt, wie das Eigenschaftenblatt für den Defaultwertverwalter für Kantenstücke des Blockdiagrammeditors aussieht. Die änderbaren Felder sind grau dargestellt, die Texte links daneben dienen zur Identifizierung der hinter den Feldern verborgenen Parameter.

3) Die größte Gruppe neuer Akteure in Bild–A1 umfaßt alle Akteure, die der direkten grafischen Bearbeitung eines Planes dienen. Auf das Zusammenspiel dieser Akteure wird in den folgenden Kapiteln sehr ausführlich eingegangen, so daß hier nur eine kurze Zusammenfassung gegeben werden soll.

Das Repertoire der möglichen Benutzeraktionen durch Menüauswahl oder Tastatur–eingaben innerhalb eines Grafikrahmens und damit innerhalb eines SPIKES–Planes ist durch den für dieses Dokument zuständigen *Dokumenteneditionsakteur* bestimmt (vgl. bereichszuständiger Editor in Bild–A4, Kapitel 1.2). Fordert der Benutzer innerhalb eines Grafikrahmens durch Drücken der entsprechenden Maustaste ein Menü an, so wird der bereichszuständige Akteur[7], in diesem Falle also der diesem Grafikrahmen zugeordnete Dokumenteditionsakteur, das dem jeweiligen Editionszustand entsprechende Menü anbieten. Ein solches Menu besteht aus einem bestimmten Repertoire von Einträgen, von denen jeweils einer durch Bewegen der Maus anwählbar ist. Mit jedem dieser Einträge ist eine Aktion verknüpft, mit der der Anwender entweder neue Grafiken erzeugen, bestehende Grafiken des Planes bearbeiten, Formulare zur Bearbeitung öffnen oder seine Editionsumgebung einrichten kann.

Neben den Symbolklassenakteuren zum Erzeugen von Symbolgrafiken und den ebenfalls schon erwähnten Formularakteuren sind für die eigentliche Edition von Grafiken weitere Akteure nötig. Zum einen gibt es eine Reihe von Basisoperationen zum Erzeugen und Bearbeiten von Basiskomponenten, wie z.B. Linien, Bögen usw. Diese Basisoperationen sind durch die Operationsakteure des Dokumenteditionsakteurs bereits vorgegeben[8].

Weiterhin existiert zum Bearbeiten von Symbolkomponenten je eines bestimmten Typs ein spezieller *Editionsakteur.* Dieser sorgt entsprechend dem jeweiligen *Menüauswahlverwalter* dafür, daß bei der Bearbeitung einer Symbolkomponente dem Benutzer ein Repertoire an Editionsschritten zur Verfügung steht, mit dem genau solche Grafiken erzeugt werden können, die mit dem entsprechenden Symbol-

7. Möglicherweise sind mehrere Dokumente geöffnet, die auf dem Bildschirm verschiedene Bereiche in Anspruch nehmen. Jedem dieser Bildschirmbereiche ist dann ein unterschiedlicher Dokumenteditionsakteur zugeordnet. Aus diesem Grunde ist von *Bereichszuständigen* die Rede. Bei Interleaf geht dieses Konzept so weit, daß es innerhalb eines Dokumentes mehrere verschiedene Bereichszuständige gibt. So existiert außer dem Dokumenteditionsakteur (kurz Dokumenteditor) noch der Komponenteneditor für die Komponentenspalte usw. Weiterhin gibt es Bereichszuständige für den Schreibtisch, die einzelnen Schubladen, die Eigenschaftenblätter, so daß jedem Bildschirmausschnitt innerhalb einer INTERLEAF–Anwendung eindeutig ein Bereichszuständiger zugeordnet werden kann. Es muß also einen unsichtbaren Akteur geben, der die Benutzereingaben an den jeweils richtigen Bereichszuständigen weiterleitet. Dieser Akteur soll, sofern er überhaupt erwähnt wird, *Steuerakteur* (vgl. [3]) genannt werden.

8. Im Zusammenhang mit Interleaf V ist hier die Menge aller standardmäßig vorgegebenen Grafikfunktionen zum interaktiven Bearbeiten von Grafikkomponenten durch den Benutzer gemeint. Daß es hierfür keinen separaten Grafikeditionsakteur gibt, ist implementierungsbedingt. In Interleaf VI existiert ein solcher Akteur.

typ verträglich sind.
So darf es innerhalb der Bearbeitung der Basisgrafik eines Kantenstücksymbols, welche durch verrundete Linienzüge dargestellt wird, nicht möglich sein, Kreise, Punktgrafiken oder gar Symbolkomponenten zu erzeugen.
Zum andern sind die Symbolkomponenten zum Teil, sehr komplex und daher mit einfachen Basisoperationen nur umständlich zu erstellen oder die Symbolkomponenten innerhalb eines Symboles müssen in bestimmter Art und Weise zueinander ausgerichtet sein. In beiden Fällen kann die Edition für den Anwender durch geeignete Funktionalität des Editionsakteurs erheblich erleichtert werden.

Zuletzt gibt es eine Reihe von Aktionen, die allgemein zur Bearbeitung von Symbolen benötigt werden. Dazu zählen Ausschneiden und Einfügen, Bewegen, Kopieren usw. Im Zusammenhang mit der besonderen Struktur von SPIKES–Plänen gibt es hierbei einige Besonderheiten im Zusammenhang mit der Kantenbaumproblematik und der Implementierung dauerhafter Referenzen, auf die später ausführlich eingegangen wird. Zuständig für diese Aktionen sind der Planeditor(–Akteur), der Cut&Paste–Akteur, sowie der Kantenstückbaumakteur.

4) Die letzte Gruppe von Akteuren ist für die syntaktische Korrektheit und logische Auswertung des Planes verantwortlich (vgl. [7]). Zum gegenwärtigen Stand ist ein Syntaxprüfer vorgesehen, der nach Aufforderung durch den Benutzer dazu veranlaßt werden kann, die durch die Grafik des Planes beschriebene Planstruktur auf Fehler hin zu untersuchen. Werden Fehler gefunden, so soll ein interaktives Verfahren zur Beseitigung dieser Fehler unterstützt werden. Auf eine ständige Überwachung des Planes wurde bewußt verzichtet, da dies einen bestimmten Editionsstil vorgegeben hätte und damit der Benutzer in seinen Editionsfreiheiten eingeschränkt worden wäre. Ebenfalls in diesem Zusammenhang zu erwähnen sind die Kantenstückakteure, die während der logischen Auswertung des Planes eine Zuordnung von Start– und Endknoten zu den entsprechenden Kanten realisieren. Während der Edition eines Planes sind keine Kantenakteure vorhanden. In dieser Phase existieren nur Kantenstücke, bei denen eine Zuordnung von Start– und Endknoten keinen Sinn ergibt.

1.2 Editionskonzept aus Benutzer– und Systemsicht

In diesem Kapitel wird in mehreren Schritten der Aufbau eines SPIKES–Editor–Systems hergeleitet. Ausgehend von der Sicht auf die Editionsumgebung, die ein Anwender bei der Arbeit mit einem SPIKES–Editor haben soll, wird überlegt, welche Aktionen innerhalb des Editorsystems nötig sind, welche Akteure dazu gebraucht werden und wie diese miteinander kommunizieren müssen.

Einleitend wird zunächst eine sehr einfache Sichtweise betrachtet, bei der es nur einschrittige Benutzeraktionen auf oberster Planebene gibt. Dieses einfache Modell wird danach erweitert, um die Bearbeitung von Untergruppen, insbesondere Symbolkomponenten, zuzulassen. Basierend auf diesem Ansatz wird daraus der allgemeine Aufbau eines SPIKES–Editors hergeleitet. Dieses dritte Modell ist geprägt durch Realisierungsüberlegungen, die sich aus der Verwendung von Interleaf als Implementierungsplattform ergeben.

1.2.1 Einschrittige Sichtweise

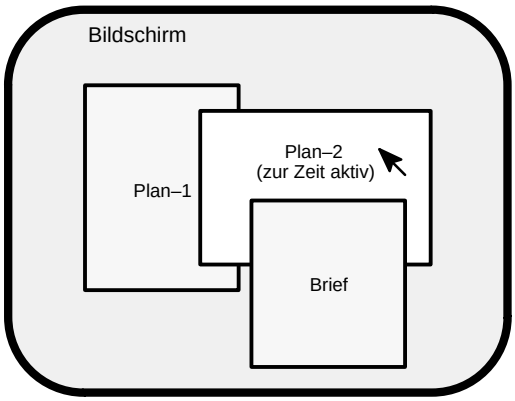
Die Bearbeitung aller SPIKES–Pläne läßt sich aus Sicht eines Anwenders immer in der in Bild–A2 beschriebenen Form darstellen. Zunächst muß der Anwender das Dokument öffnen, in dem sich der zu bearbeitende Plan befindet, oder in dem er einen neuen Plan erzeugen will. Darauf, sowie auf das Laden und Speichern von Dokumenten soll in dieser Arbeit nicht näher eingegangen werden.

Nach dem Öffnen des Planes kann der Benutzer mittels Maus und Tastatur Änderungen am Plan vornehmen, die auf dem Bildschirm dargestellt werden. Durch Drücken der Menütaste der Maus erscheint auf dem Bildschirm ein Menü, das ein bestimmtes Repertoire an Aktionen beschreibt, unter denen der Benutzer auswählen kann. Dieses Repertoire ist abhängig vom Plantyp und damit dem Editortyp, sowie der Menge der ausgewählten Plankomponenten. Zum Beispiel kann der Benutzer nur dann Plankomponenten erzeugen, wenn nichts ausgewählt ist, oder er kann nur dann logische Attribute bearbeiten, wenn er zuvor die gewünschte Plankomponente ausgewählt hat. Alles in allem stellt sich die Bearbeitung des Planes als eine Folge von einzelnen Editionsschritten dar, die durch das Schließen des Planes beendet wird.

Zunächst interessieren dabei weniger die konkreten Auswirkungen eines bestimmten Typs von Editionsschritten als vielmehr das allgemeine System, das nötig ist, um eine solche Edition zu ermöglichen. Dazu sollen alle möglichen Benutzeraktionen im folgenden als einschrittige, das heißt, unteilbare Aktionen auf oberster Planebene betrachtet werden, auch wenn sich Aktionen wie Bearbeiten oder Erzeugen komplexer Symbolkomponenten selbst wieder in eine Folge von Unteraktionen zerlegen lassen.

Bild–A3 zeigt hierzu eine Abbildung der Abläufe aus Benutzersicht auf die korrespondierenden Aktionen innerhalb des Editorsystem. Die diesen Ablauf realisierenden Akteure sind in Bild–A4 gezeigt und werden im folgenden näher beschrieben.

Solange der Benutzer nichts unternimmt, ist das ganze Editorsystem in Ruhe und wartet auf ein Eingabeereignis. Ein solches Ereignis kann sein, daß der Benutzer die Maus bewegt , eine Maus–taste drückt oder die Tastatur betätigt. Der grafische Ein–/Ausgabedienst fängt diese Ereignisse auf und leitet sie an den *Steuerakteur* weiter. Dessen Aufgabe ist es, den Zuständigen, das heißt, den Empfänger dieser Ereignisse, zu ermitteln. Mehrere potentiell Zuständige kann es beispielsweise geben, wenn es, wie in Bild–8 gezeigt, nebeneinander mehrere geöffnete Dokumente gibt.



Bild–8: Bildschirm mit mehreren Fenstern für unterschiedliche Dokumente

Jedem dieser Dokumente ist ein eigener Bildschirmbereich zugeordnet. Nun gilt für eine Reihe grafischer Benutzeroberflächen die Regel, daß derjenige der Empfänger besagter Eingabeereignisse ist, in dessen Bildschirmausschnitt sich gerade der Mauszeiger befindet. Dies heißt, daß es einen Steuerakteur geben muß, der anhand der Mauszeigerposition und dem Ereignistyp den *Bereichszuständigen* für die Eingabeereignisse ermittelt. Die Abhängigkeit vom Ereignistyp hängt mit der Modellvorstellung zusammen, daß es innerhalb eines Grafikrahmens mehrere Bereichszuständige geben soll. So soll beim Drücken der Menütaste der bereichszuständige Editor eine Nachricht gesandt bekommen, beim Auswählen mit der Auswahltaste und beim interaktiven Bewegen von Grafiken mit der Maus, sollen hingegen die jeweiligen Grafikakteure Nachrichten gesandt bekommen.

Desweiteren übernimmt der Steuerakteur Filter– und Übersetzungsaufgaben. So kann es sinnvoll sein, die Bewegungsfreiheit des Mauszeigers einzuschränken. Beispielsweise sollte es dem Benutzer beim interaktiven Bewegen einer Grafik nicht erlaubt sein, den Grafikrahmen zu verlassen. Zu den Übersetzungsaufgaben gehören die Übersetzung der eingehenden Tastaturcodes in den aktuellen Zeichensatz mittels einer Tastaturtabelle oder es wäre eine Umbelegung der Maustasten denkbar.

Angenommen der Benutzer hat einen SPIKES–Plan seiner Wahl geöffnet und drückt nun in diesem die Menütaste. Der Steuerakteur wird daraufhin eine Menüanforderung an den bereichszu-

ständigen SPIKES–Editor senden. Innerhalb des Editors wird die Nachricht an den Menüauswähler weitergeleitet. Dieser Menüauswähler stellt zunächst fest, welche Plankomponenten ausgewählt, das heißt zur Bearbeitung markiert sind. Abhängig von dieser *Selektionsmenge* aktiviert er den für diesen Fall vorgesehenen *Menüakteur*. Dieser stellt auf dem Bildschirm ein Menü dar und wartet auf die Wahl des Benutzers. Zuständig für diese Aktion ist innerhalb des Menüakteurs der *Anforderungsannahmeakteur*. Dieser liest bei Anstoß des Menüakteurs aus einem lokalen Speicher die Menübeschreibung, welche in Listenform nacheinander die Definition aller Menüeinträge enthält, unter denen der Benutzer auswählen kann. Jeder Eintrag besteht aus einer Zeichenfolge, die auf dem Bildschirm für diesen Menüpunkt dargestellt wird und einem Verweis auf den dahinter verborgenen Auftrag. Zusätzlich kann der Anforderungsannahmeakteur auf einen Speicher zugreifen, in dem die *Menühistory* abgelegt ist. Bei einem Menüaufruf wird zuerst mittels der Menübeschreibung das Menü auf dem Bildschirm dargestellt. Anschließend wird eine Vorauswahl mit Hilfe der Menühistory getroffen. Diese Auswahl entspricht der Menüauswahl beim letzten Aufruf und erleichtert insbesondere bei großen Menübäumen das Auffinden eines Punktes bei sich wiederholenden Aktionen. Hat der Benutzer seine Wahl getroffen, wird diese als neue Vorgabe in der Menühistory vermerkt. Mittels des Auftragsverweises wird der dieser Aktion zugeordnete *Auftragszusteller* identifiziert und angestoßen. Dieser leitet den Editionsschritt durch Versenden einer Nachricht an die betroffenen Akteure ein und hat darüber hinaus keine weiteren Aufgaben. Als Empfänger kommen dabei Akteure von Plankomponenten, Klassenakteure von Plankomponenten und der *Editionsakteur* des SPIKES–Editors in Frage. Auf das Repertoire an möglichen Aktionen soll im folgenden kurz eingegangen werden.

Um eine Plankomponente zu erzeugen, schickt der durch die Benutzerwahl angestoßene Auftragszusteller dem Klassenakteur für die gewünschte Plankomponente eine Nachricht zum Erzeugen eines Exemplars dieser Klasse. Entsprechend dem der Klasse typischen Exemplarbauplan erzeugt der Klassenakteur daraufhin einen Plankomponentenakteur inklusive der typischen Symbolgrafik und initialisiert das neue Objekt entsprechend der aktuellen Defaultwerte[9.]. Der Plan enthält nun eine weitere Plankomponente. Daß dies ein mehrschrittiger Editionsprozeß sein kann, innerhalb dessen weitere Komponenten erzeugt und gelöscht werden können, spielt in diesem Zusammenhang keine Rolle.

Ebenso wie die Erzeugung wird die Bearbeitung einer Plankomponente eingeleitet. Der durch die Benutzerwahl angestoßene Auftragszusteller ermittelt die betroffenen Plankomponentenakteure und schickt diesen die entsprechende Botschaft. In der Regel werden dies Plankomponenten sein, die der Benutzer zuvor ausgewählt hat. Daß die dazugehörenden Akteure den Nachrichtentyp zur Bearbeitung verstehen, ist dadurch sichergestellt, daß der Menüauswähler entsprechend der Selektionsmenge nur solche Menüakteure anstoßen kann, deren Aufträge von den ausgewählten Akteuren ausgeführt werden können. Will man beispielsweise eine Plankom-

9.. In der hier gewählten Betrachtungsweise soll zur Vereinfachung zunächst der Klassenakteur die Verwaltung der Defaultwerte übernehmen. Tatsächlich sind bei SPIKES–Editoren die eindeutig einem Plan zugeordneten Master für diese Ausgabe verantwortlich, da, wie in Kapitel 1.1.3 beschrieben, nur durch die Einführung eigener plangebundener Objekte eine individuelle Defaultwertverwaltung für jeden Plan möglich ist.

ponente bewegen, so wird man dem zuständigen Akteur eine entsprechende Nachricht schicken, dieser wird beim Steuerakteur die Animation anmelden und solange, bis der Benutzer durch Drücken einer Maustaste die Aktion beendet, kann die Komponente frei innerhalb des Grafikrahmens bewegt werden[10].

Neben diesen beschriebenen Editionsoperationen gibt es andere, bei denen es nicht sinnvoll oder gar unmöglich ist, sie als Teil eines Plankomponentenakteurs oder eines Klassenakteurs für Plankomponenten anzusehen. Diese Aktionen fallen in den Zuständigkeitsbereich des *Editionsakteurs*. Dazu gehören komplexe Operationen, die mehrere Plankomponenten gleichzeitig betreffen und deren Beziehungen untereinander berücksichtigt werden müssen. Man denke an Syntaxprüfung, sowie alle Aufgaben, die im Zusammenhang mit dem Aufheben von Editionsschritten stehen (Undo–Akteur). Beim Bearbeiten oder Erzeugen komplexer Plankomponenten ist es denkbar, daß es Operationsakteure gibt, die dem Benutzer Editionshilfen geben (dazu im folgenden mehr). Diese Akteure müssen ihrerseits dann mit den Akteuren für Plankomponenten und deren Klassen kommunizieren.

Nach Abschluß eines jeden Editionsschrittes, der eine der oben angedeuteten Aktionen zur Folge hat, kehrt der Editor wieder in seinen Ruhezustand zurück und wartet auf das nächste Eingabeereignis. Das Wesen des Editors als ereignisgetriebenes System wird hier deutlich sichtbar.

1.2.2 Dialogzustände und Editoren

Bislang konnte der Benutzer nur einschrittige Editionen auf oberster Planebene vornehmen. Gerade das Erzeugen und Bearbeiten komplexer Plankomponenten stellt sich häufig als mehrschrittiger Prozeß dar. Jede Plankomponente ist eine Grafikgruppe, die aus mehreren Grafikgruppen (”Plankomponenten–Komponenten”) bestehen kann, für die wiederum dasselbe gilt usw. Beim Bearbeiten von Plankomponenten will der Benutzer häufig nicht die ganze Grafikgruppe, sondern nur eine Komponente davon bearbeiten. Bild–A5 gibt eine Vorstellung vom prinzipiellen Arbeiten mit Grafikgruppen. Auf oberster Planebene sind alle Plankomponenten als Einheit auswählbar und bearbeitbar. Will der Benutzer nur Teile einer Plankomponente ändern, so gelangt er in einen neuen Editionszustand. Auf dieser Ebene kann er nur Komponenten innerhalb dieser Plankomponente erzeugen oder bearbeiten. Die übrigen Komponenten oberhalb dieser Bearbeitungsebene können in diesem Zustand nicht ausgewählt oder bearbeitet werden. Hierarchisch kann man auf diesem Wege immer tiefer gehen, bis man auf Komponenten stößt, die keine zusammengesetzten Gruppen sind, sondern unteilbare Basisgrafiken darstellen. Wird eine Plankomponente erzeugt, muß diese Schritt für Schritt aus solchen unteilbaren Basisgrafiken, auch *Grafikprimitiven* genannt, zusammengesetzt werden. Wird eine Plankomponente bearbeitet, sind es letztlich solche Basisgrafiken, die bearbeitet, erzeugt oder gelöscht werden. In der bisherigen

10.. Bei vielen (und fast allen interaktiven) Grafikoperationen verließen die Entwickler von Interleaf–V die objektorientierte Sichtweise. Vielmehr gibt es eine Bibliothek von Grafikfunktionen, die aufgerufen werden können, um Änderungen an den selektierten Grafikobjekten auszuführen.

Sichtweise geschah dies automatisch in einem einzigen Schritt, jetzt soll untersucht werden, wie sich das Editormodell durch diese neue Sicht verändert.

Zuerst soll betrachtet werden, wie sich dies auf das zur Verfügung zu stellenden Editionsrepertoire auswirkt. Dürfte auf allen Ebenen dasselbe Repertoire zur Verfügung gestellt werden, könnten auf jeder Ebene Exemplare von allen Grafikklassen erzeugt werden, der Benutzer hätte immer die gleiche Funktionalität des Editors. Dies darf natürlich nicht sein. So kann ein Symbol nur aus einer definierten Anzahl Komponenten eines bestimmten Typs bestehen. Zum Beispiel darf die Grafikgruppe für ein Speichersymbol keine Grafikgruppe für ein weiteres Speichersymbol enthalten. Es ist hierbei wichtig zu unterscheiden zwischen dem Enthaltensein von Grafikgruppen ineinander und dem grafisch sichtbaren bzw. logischen Enthaltensein von Knoten und Kanten, wobei letzteres zulässig ist. Will man nicht nur einschrittige Editionen zulassen, was die Auswahl der möglichen Symbolausprägungen erheblich einschränken würde, so muß sich das Repertoire der zulässigen Aktionen beim Bearbeiten einer Untergruppe von dem Repertoire auf oberster Planebene unterscheiden. Unterschiedliches Verhalten eines System auf die gleiche Eingabe kann aber nur bedeuten, daß das System unterschiedliche Zustände besitzen muß. Bei dem betrachteten Editorsystem spricht man daher in Bezug auf die Zustände, die für den Dialog mit dem Benutzer relevant sind, von unterschiedlichen *Dialogzuständen* oder auch *Editionszuständen*. Jeder Dialogzustand ist dabei gekennzeichnet durch die Menge der möglichen Aufträge, die man an das System in diesem Zustand senden kann und durch die Folgezustände in die man aus diesem Zustand gelangen kann. Stellt man alle Dialogzustände und deren Übergänge in einer Grafik dar, erhält man den Dialoggraphen des Editorsystems. Als Beispiel kann der in Bild–A31 gezeigte Dialoggraph des Blockdiagrammeditors dienen. Ausgehend von dem Grundzustand ”Bearbeitung auf Plankomponentenebene” kann man je nach Auswahlmenge eine Aktion aus einem der im Anhang H gezeigten Menübäume für diesen Editionszustand auswählen. Einige dieser Aktionen führen in andere Dialogzustände mit anderen Menüs, wie zum Beispiel die ”Bearbeitung modifizierender Kanten”. Durch Anwahl des Menüpunktes ”Schließen” gelangt der Benutzer wieder in den vorausgegangenen Dialogzustand. Aus der relativ geringen Zahlen von Dialogzuständen ist leicht ersichtlich, daß nicht für die Bearbeitung jeder Symbolgrafik und der jeweiligen Untergruppen ein eigener Dialogzustand eingeführt wurde. Sinnvoll ist dies nur dort, wo Attribute oder die Basisgrafik eines Symboles in einer Folge aus mehreren Editionsschritten bearbeitbar sein sollen oder dürfen. Geht es nur darum ein Attribut an einer symbolspezifischen Stelle zu erzeugen, ist dazu kein separater Dialogzustand zum mühsamen Positionieren nötig, sondern dies kann in einer einschrittigen Aktion durch eine entsprechende Aufforderung an den Symbolakteur erfolgen, der das Attribut automatisch richtig plaziert (z.B. Markierung einer Stelle in einem Petrinetz).

Ein naheliegender Ansatz zur Realisierung eines Editors mit unterschiedlichen Dialogzuständen besteht darin, im Menüauswahlverwalter die komplette Information des Dialoggraphen abzulegen, sowie einen Speicher für den aktuellen Dialogzustand vorzusehen. Damit könnte der Menüauswahlverwalter nicht nur selektionsmengen– sondern auch dialogzustandsabhängig

agieren. Die Funktionalität des Editionsakteurs müßte entsprechend erweitert werden. Im Hinblick auf Überschaubarkeit, Änderbarkeit und Austauschbarkeit von Komponenten innerhalb des Editors ist dieser Weg jedoch denkbar ungünstig. Anstatt einen alleskönnenden Rieseneditor zu realisieren, ist es besser, für jeden geplanten Dialogzustand einen eigenen kleinen Editor zu implementieren und bei einem Wechsel des Dialogzustandes den Editor zu wechseln. Der Ablauf läßt sich wie in Bild–A6 gezeigt darstellen. Zusätzlich zu dem Zuständigkeitsbereich für Toplevelaktionen gibt es eine Reihe weiterer Zuständigkeitsbereiche für Editoren, die für das Bearbeiten spezieller Untergruppen zuständig sind. Wie in Bild–A3, das eine einschrittige Sichtweise der Edition von Plänen ohne Untergruppenbearbeitung zeigt, wird auch nach der Erweiterung dieses Modells um die interaktive mehrschrittige Untergruppenbearbeitung weiterhin eine Ereignisschleife durchlaufen, in der nach Ermitteln des Bereichszuständigen dieser auf das registrierte Ereignis reagiert. Je nach Bearbeitungsebene, z.B. Plankomponentenebene oder Linienzugedition eines Kantenstückes, werden nun jedoch zusätzliche Zuständigkeitsbereiche für denselben Bildschirmbereich unterschieden. Jeder dieser Zuständigkeitsbereiche beschreibt das Verhalten des Editorsystems auf eine Menüanforderung in einem anderen Dialogzustand. Es existiert in jedem Zuständigkeitsbereich eine fett eingezeichnete Stelle, von denen im Ruhezustand des Systems, das heißt für den Fall des Wartens auf ein Eingabeereignis, genau eine Stelle mit einer sogenannten Marke belegt ist, die den aktuellen Dialogzustand kennzeichnet. Im Falle einer Menüanforderung kann so nur in denjenigen Zuständigkeitsbereich verzweigt werden, der das Editionsrepertoire des aktuellen Dialogzustandes beschreibt. Nach Abschluß der Ereignisbehandlung ist wieder eine dieser fett eingezeichneten Stellen markiert. Die jeweilige Ereignisbehandlung kann dabei als Zustandsübergang vom aktuellen in den folgenden Dialogzustand interpretiert werden. Wird eine Untergruppenbearbeitung gestartet, so wird durch Markenfluß über die Zuständigkeitsgrenze des Editors der Editor gewechselt, ebenso wird bei Beendigung einer Untergruppenbearbeitung wieder eine Marke zurückgegeben und somit die Kontrolle an den ursprünglichen Editor zurückgegeben.

Bild–A7 zeigt den prinzipiellen Aufbau eines solchen Systems mit der Möglichkeit zur Untergruppenbearbeitung. Es handelt sich hierbei um eine Erweiterung des einschrittigen Systemmodells um die zur Realisierung und Bearbeitung von Untergruppen zusätzlich benötigten Akteure. Da einige Plankomponenten, hierbei insbesondere alle Symbole sich aus einer Reihe von Grafikkomponenten zusammensetzen, die im Falle von Grafikgruppen ebenfalls aus weiteren Grafikkomponenten bestehen können, ist für jedes dieser Grafikobjekte ein eigener Akteur erforderlich. Es ergibt sich eine Hierarchie von Grafikakteuren, wie sie schon in Kapitel 1.1.2, Bild–3 dargestellt ist. Am unteren Ende dieser Hierarchie findet man die Akteure aller Grafikprimitiven, die letztlich das Aussehen eines Grafikobjektes bestimmen. Diese Struktur spiegelt sich in der Hierarchie der korrespondierenden Klassenakteure wieder, von denen jeweils einer für das Erzeugen von Grafikobjekten eines bestimmten Typs verantwortlich ist. Klassenakteure für Plankomponenten können Plankomponenten (Akteur und Symbolgrafik) erzeugen, wobei sie den Plan verändern. Dazu müssen sie entsprechend einem klassenspezifischen Exemplarbauplan einzelne

Komponenten der Symbolgrafiken erzeugen, wozu sie den entsprechenden Klassenakteuren für diese tiefere Ebene die jeweiligen Aufträge geben müssen. Ein anderer Weg besteht darin, interaktiv mit dem Benutzer mit Hilfe eines speziellen Editors diese Komponenten zu definieren. Dazu existiert eine Reihe geeigneter Komponenteneditoren, die in ihrer Aufbaustruktur dem Topleveleditor gleichen. Diese Hierarchie läßt sich nach unten fortsetzen, bis man auf unteilbare Basisgrafiken stößt.

Natürlich ist dieser streng hierarchische Ansatz in Bezug auf die Hierarchie der Klassenakteure stark idealisiert ist. Würde man dies wirklich so realisieren, so müßte es beispielsweise getrennte Klassenakteure für Linien der ersten Gruppenebene, der zweiten Gruppenebene usw. geben, was ziemlich unsinnig ist. Trotzdem mag die prinzipielle Vorstellung zunächst hilfreich sein, wenn man statt dessen an einen Klassenakteur für Kantenstücke auf oberster Ebene, darunter an Klassenakteure für Pfeilspitzen und verrundete Linienzüge und auf unterster Ebene an solche für Linien und Bögen denkt. Ebenfalls ist es keineswegs so, daß es für jede Gruppenebene Editoren geben muß, wie bereits zuvor erwähnt. Selektiert man beispielsweise eine Plankomponente und wählt den Menüpunkt ”Bearbeite Basisgrafik” (vgl. Bild–A31) aus, so wird eine Ebene, die der Symbolgruppe, übersprungen. In diesem Falle würde demzufolge von Plankomponentenebene direkt ein Editor zur Bearbeitung der zweiten Untergruppenebene aufgerufen werden. Will man dies mit der streng hierarchischen Sichtweise in Einklang bringen, so bedeutet dies, daß der Editor einer zu überspringenden Ebene lediglich dazu dient, den Editionsauftrag an den Akteur der zu bearbeitenden Untergruppe der nächsten Grafikebene weiterzuleiten. Tatsächlich gibt es bei den existierenden SPIKES–Editoren in der Regel nur zwei Editionsebenen und je nach Symboltyp mindestens drei oder mehr Grafikebenen, jedoch ist dieser Unterschied in der folgenden Beschreibung nicht von Bedeutung.

Weiterhin muß es zur Realisierung eines Editors, der Untergruppenbearbeitung unterstützt, einen Mechanismus geben, der bei der Bearbeitung einer Untergruppe dafür sorgt, daß die Komponenten der darüber liegenden Ebenen nicht bearbeitet oder selektiert werden können. Beim *Öffnen einer Untergruppe* müssen alle Komponenten innerhalb dieser Gruppe einzeln anwählbar werden (insofern dies nicht explizit verboten ist) und alle Komponenten darüber für die Bearbeitung unerreichbar werden. Beim *Schließen einer Untergruppe* muß dies wieder rückgängig gemacht werden. In diesem Modell ist dafür der *Auswahl– und Restmengenverwalter* zuständig. Wie dies im einzelnen realisiert werden kann, wird in [3] beschrieben.

Bevor der Ablauf einer solchen mehrschrittigen Aktion und die Kommunikation dabei unter den dazu nötigen Akteuren geschildert werden kann, ist es nötig, die zwei unterschiedlichen Editionskonzepte vorzustellen, die bei SPIKES–Editoren zum Einsatz kommen:

- Das erste Konzept beschreibt Editionen derart, wie sie bislang dargestellt wurden. Der Benutzer öffnet eine Grafikgruppe und erzeugt, bearbeitet und löscht Komponenten. Dies tut er solange, bis die Grafikgruppe exakt das gewünschte Aussehen hat. Danach kann die Gruppe geschlossen werden. Das Aussehen bleibt bis zur nächsten

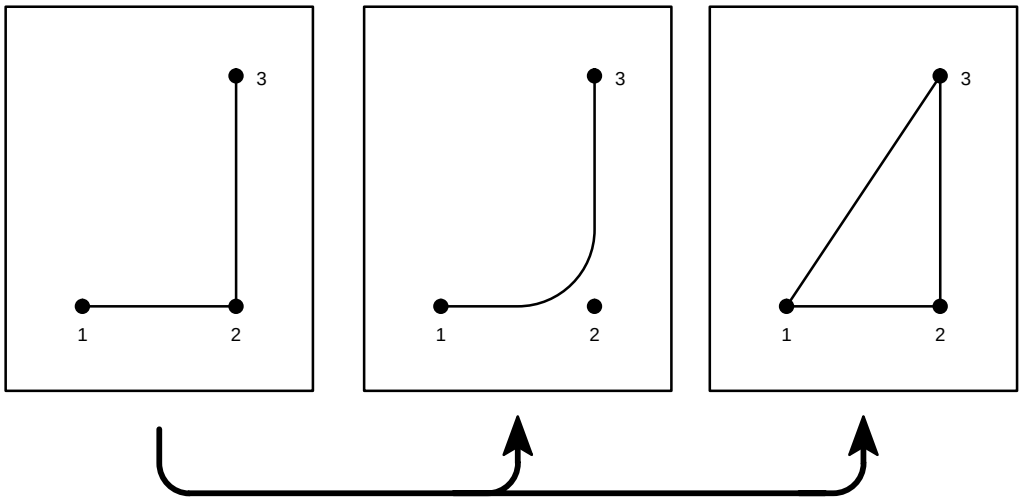
Bearbeitung durch den Benutzer unverändert. Der Akteur für die ganze Gruppe bleibt dabei im Zusammenhang mit der Bearbeitung bis auf das anfängliche Anstoßen der Edition passiv.

- Das zweite Konzept geht von der Idee aus, daß bei einem bestimmten Komponententyp, das Aussehen nicht beliebig definierbar ist. Dadurch ist die Grafik einer solchen Komponente in der Regel nur innerhalb gewisser durch wenige Parameter eindeutig beschreibbarer Grenzen variierbar. Es reicht daher, diese Parameter vorzugeben und die Komponente darauf basierend zu erstellen. In diesem Sinne werden beim Bearbeiten einer Komponente zunächst nur deren definierende Parameter editiert, sei es grafisch oder auf textuellem Wege, und nach Editionsabschluß gibt der Editor diese Parameter als Ergebnis an seinen Auftraggeber, den Komponentenakteur (oder einen Klassenakteur beim Erzeugen), zurück. Erst dieser kreiert daraus das neue Aussehen der Komponentengrafik. Bei grafischer Edition erfolgt die Definition der Parameter dabei mittels Hilfsgrafiken, wobei diese so gewählt sind, daß der Benutzer eine klare Vorstellung davon hat, wie die spätere Grafik aussehen wird.

  Bild–9 zeigt ein einfaches Beispiel zur Erläuterung dieses zweiten Konzeptes, welches auch deutlich macht, wie flexibel dieses Konzept ist. Um den verrundeten Linienzug in der Mitte (beispielsweise die Basisgrafik eines Kantenstückes) zu erstellen, reicht es, die Punktfolge 1, 2, 3 und den Verrundungsradius vorzugeben. Nimmt man den Verrundungsradius als gegebenen Defaultparameter an, so reicht es, mit einem sehr einfachen Editor den links dargestellten Linienzug einzugeben, der diese Punkte definiert. Ein solches Verfahren hat gegenüber dem ersten Konzept mehrere Vorteile. Für den Benutzer wird der Zeichenaufwand geringer. Auch bedeutet ein solcher Editor aufgrund der geringeren zu unterstützenden Funktionalität für die Entwickler weniger Aufwand. Letztlich kann ein solcher Editor nicht nur zur Bearbeitung von Komponenten eines bestimmten Typs, sondern für mehrere unterschiedliche Typen eingesetzt werden. In obigem Beispiel könnten mit der Punktfolge ebenfalls geschlossene Polygone, wie ganz rechts in Bild–9 gezeigt, oder Stützpunkte für Splinekurven definiert werden.

  Zudem gibt es spezielle Klassen von Plankomponenten, deren Exemplare ohnehin in der Lage sein müssen, ihre Symbolgrafik selbstständig zu erstellen. In diesem Zusammenhang sei z.B. an die automatische Ausrichtung und Anpassung von modifizierenden Kanten in Blockdiagrammen auf Knoten gedacht. Es wäre paradox, den Benutzer diese Grafiken mühsam Schritt für Schritt erstellen zu lassen, wenn der Grafikakteur bei Vorgabe einiger Parameter selbst dazu in der Lage ist.

Vergleicht man beide Konzepte, so besteht beim zweiten Konzept sicherlich ein Nachteil darin, daß der Benutzer es natürlich am liebsten hat, wenn sich die bearbeitete Grafik und das Endergeb-



Bild–9: Edition komplexer Grafiken durch Bearbeitung einfacherer Stellvertretergrafiken

nis gleichen. Das führt dazu, daß viele Benutzer den ersten Ansatz bevorzugen. Umgekehrt gibt es jedoch eine Reihe komplexer Editionen, die ohne vereinfachte Darstellung in der Interaktion mit dem Benutzer zu langsam ausfallen würden. Für solche komplexen Editionen wird bei SPIKES–Editoren das zweite Konzept angewendet. Das erste Konzept findet seine Anwendung bei der Bearbeitung von freien grafischen Attributen und Textattributen.

Was genau beim mehrschrittigen Erzeugen und Bearbeiten passiert, soll Bild–A8 erläutern. Auf die Besonderheiten des jeweiligen Konzeptes wird dabei an entsprechender Stelle hingewiesen.

Zu Beginn der Betrachtung befindet sich das Editorsystem im Dialogzustand ”Bearbeitung auf Plankomponentenebene”, also in dem Zustand, in dem sich das System nach Öffnen eines Planes befindet. Der bereichszuständige Editor für den Grafikrahmen ist der Topleveleditor. Angenommen der Benutzer wollte eine Plankomponente erzeugen und hat bereits mittels des Menüakteurs eine Botschaft an den zuständigen Klassenakteur gesandt. Abhängig von der jeweiligen Klasse wird die Plankomponente daraufhin einfach einschrittig oder in einem mehrschrittigen Editionsprozeß erzeugt. Im zweiten Fall handelt es sich um Plankomponenten, deren Basisgrafik vom Benutzer unmittelbar nach bzw. in einigen Fällen direkt vor dem Erzeugen bestimmt werden kann.

Dazu ist dem Klassenakteur ein geeigneter Editor bekannt, hier Komponenteneditor genannt, dessen Editionsakteur er eine Startnachricht schickt. Dabei übergibt er seinen eigenen Namen und den Typ der Abschlußbotschaft. Zusätzlich können dem Editionsakteur weitere Parameter als Editionsvorgabe übergeben werden. Damit der Editionsakteur das spätere Ergebnis korrekt an seinen Auftraggeber schicken kann, notiert er zuerst seinen Auftraggeber, in diesem Fall also

den Klassenakteur, sowie die Abschlußbotschaft und die vorgegebenen Editionsparameter[11.]. Anschließend muß er sich beim Steuerakteur anmelden, wodurch alle Eingabeereignisse, die sonst an den Topleveleditor geschickt würden, nun an den Komponenteneditor geschickt werden. Unabhängig vom Editionskonzept ist es nötig, daß eine Untergruppe geöffnet wird, um die übrigen Plankomponenten vor einer versehentlichen Bearbeitung zu schützen. Im Falle des ersten Editionskonzeptes ist dies naturgemäß der Fall, da eben die zu definierende oder zu bearbeitende Untergruppe geöffnet werden muß. Im Falle des zweiten Editionskonzeptes dient die Gruppe dazu, die Hilfsgrafik aufzunehmen, auf deren Basis nach Editionsabschluß die neuen grafischen Parameter bestimmt werden. Es ist Aufgabe des Initialisierungsakteurs eines Komponenteneditors, basierend auf einer gegebenenfalls bereits vorgegebenen grafischen Parametern diese Hilfsgrafik zu erstellen. Die Untergruppe hat in diesem Fall reine Schutzfunktion, um eine abgeschlossene Edition zu gewährleisten und wird anschließend nach Bestimmung der Parameter wieder gelöscht. Im Falle des Erzeugens kann es je nach Editortyp erlaubt sein, daß kein einziger Parameter vorgegeben werden muß und so der Benutzer weitgehende Freiheit bei der Erstellung der Grafik hat. Ist der Auftraggeber jedoch eine schon bestehende Komponente, so wird diese genau die sie beschreibenden Parameter übergeben, so daß die daraus erstellte Hilfsgrafik eben diese Komponente beschreibt.

Die nun folgende eigentliche Edition dieser Grafik ist völlig vom Typ des jeweiligen Editors abhängig. Zwar wird es zwischen den einzelnen Editoren gewisse Gemeinsamkeiten geben, wie möglicherweise das Bewegen oder Dehnen von einzelnen Grafikobjekten oder das Aufheben von Editionsschritten, aber im wesentlichen handelt es sich um sehr spezielle Editoren, die auf die Erstellung und Bearbeitung von Grafikkomponenten eines oder weniger Typen optimiert wurden. In den meisten Fällen ist das Repertoire und die Anzahl der erstellbaren Hilfsgrafiken vorgeben oder begrenzt. Müssen die Grafikobjekte in einer bestimmten grafischen Relation zu einander positioniert werden, kann der Editor dies durch geeignete Automatismen unterstützen.

Beendet wird die Edition bei allen Editoren dadurch, daß der Benutzer den dafür vorgesehenen Eintrag ”Schließen” aus dem Menü auswählt. Der betreffende Menüverwalter des Komponenteneditors wird daraufhin den *Abschlußakteur* anstoßen. Je nach Editortyp ermittelt dieser selbst die Parameter für die zu erzeugende oder in Bearbeitung befindliche Komponente und verwirft anschließend die ganze Hilfsgrafik oder er beendet lediglich die Untergruppenbearbeitung, schließt also die Grafikgruppe und überläßt die Auswertung dem jeweiligen Auftraggeber. Bevor

11.. Es ist in Interleaf–V nicht möglich das Ergebnis einer Interaktion mit dem Benutzer zur Bearbeitung von Grafik direkt als Rückgabewert der die Interaktion einleitenden Funktion zu erhalten. Es ist allerdings möglich, wie in [7] dargestellt wird, durch das die Interaktion beendende Ereignis die Abwicklung einer weiteren Prozedur zur Auswertung des Interaktionsergebnisses einzuleiten. Nur innerhalb einer solchen Folgeprozedur können dem Editionsauftraggeber die Editionsergebnisse in Form einer expliziten Nachricht übermittelt werden. Es ist daher nötig, daß vor Beginn der Interaktion Adressat und Botschaftstyp für die Abschlußbotschaft feststehen. Dadurch, daß bei SPIKES–Editoren diese explizit von dem Editionsauftraggeber an den jeweiligen Editor übergeben werden, ist es möglich, anstelle des wirklichen Auftraggebers einen anderen Akteur als Auftraggeber anzugeben, so daß der Adressat des Ergebnisses frei wählbar ist. Im Hinblick auf eine mögliche gezielte Ausnutzung dieses Mechanismus durch folgende Editorgenerationen wurde dessen Darstellung bewußt in die hier gezeigte Sichtweise des Editorsystem aufgenommen.

der Komponenteneditor die Editionsergebnisse an seinen Auftraggeber schicken kann, muß er sich zuvor beim Steuerakteur abmelden und somit die Kontrolle an den Topleveleditor zurückgeben.

Nachdem der Auftraggeber die Ergebnisse der Edition erhalten hat, ist in der Regel eine Auswertung bzw. Nachbearbeitung dieser Ergebnisse erforderlich. Handelte es sich um eine Edition entsprechend dem zweiten Konzept, so wird aus den im vorangegangen Editionsprozeß bestimmten grafischen Parametern die gewünschte Grafik erstellt. Beim Erzeugen von Grafikkomponenten sind dabei zusätzlich die jeweiligen plan– und klassenspezifischen Defaultwerte, wie zum Beispiel Linienstärke und Farbe zu berücksichtigen. Bei der Bearbeitung einer Grafikkomponente können diese Werte entsprechend der ursprünglichen Grafik übernommen werden. In der Regel wird man jedoch vor der Bearbeitung der grafischen Parameter einer Grafikkomponente mit Hilfe einer Hilfsgrafik die zu bearbeitende Grafikkomponente aus dem Plan entfernen. Die ursprüngliche Komponente wird somit scheinbar durch die Hilfsgrafik ersetzt. Um nach der Bearbeitung trotzdem eine Anpassung der neuen Grafik gemäß den zu erhaltenden ursprünglichen Parametern vornehmen zu können, darf die ursprüngliche Grafik nicht gelöscht werden, ohne zuvor die relevanten Parameter zu sichern. Neben diesen Anpassungsschritten sind darüber hinaus gegebenenfalls zusätzliche Attributgrafiken zu erzeugen (wie z.B. die Pfeilspitze bei einem gerichteten Kantenstück nach einer Linienzugedition).

Neben der Möglichkeit, eine Plankomponente gemäß dem oben beschriebenen Konzept schrittweise mittels einer Hilfsgrafik zu definieren, besteht natürlich auch die Möglichkeit, zunächst eine vollwertige Plankomponente einschrittig (basierend auf entsprechenden Defaultwerten) zu erzeugen und unmittelbar anschließend eine Bearbeitung dieser Komponente zu starten. In diesem Fall würde der anstoßende Auftragszusteller zunächst den entsprechenden Klassenakteur beauftragen, eine Plankomponente zu erzeugen und diese anschließend beauftragen, sich bearbeiten zu lassen.

Bei der Edition von Grafik entsprechend dem ersten Konzept entfallen diese Schritte. Die Bearbeitung bereits bestehender Plankomponenten unterscheidet sich hierbei allerdings nur geringfügig. Zu Beginn der Edition wird der Editor die Grafikgruppe der zu bearbeitenden Komponente öffnen und anschließend wird der Benutzer direkt deren Inhalt bearbeiten. Als Aufrufparameter muß in diesem Falle also ein Verweis auf die Gruppe übergeben werden.

1.2.3 Implementierungsnahe Sicht

Bedingt durch die Implementierung der SPIKES–Editoren als Aufsatz auf das Dokumentverarbeitungssystem Interleaf V, ist es nicht sinnvoll bzw. an einigen Stellen unmöglich das vorgestellte allgemeine Systemmodell direkt umzusetzen. Bild–A9 zeigt ein implementierungsnahes Aufbaumodell des SPIKES–Editorsystems. Aus Gründen der Übersichtlichkeit ist die hierarchische Ordnung innerhalb von Plänen nicht mehr dargestellt. Dadurch bedingt wird ebenso nicht

mehr ausdrücklich zwischen Klassenakteuren für Plankomponenten und solchen unterer Ebenen unterschieden.

Desweiteren fällt auf, daß der Topleveleditor, sowie die Komponenteneditoren nicht mehr als Einheit, sondern zerlegt in Editionsakteure und Menüauswahlverwalter dargestellt und auch realisiert wurden. Der Editionsakteur des Topleveleditors besteht nach dieser Definition aus den bereits erwähnten Akteuren Planeditor[12.], Syntaxprüfer, sowie dem Kantenstückbaumakteur, wobei letztere zunächst als Komponenten des Planeditors betrachtet werden sollen.

Ein praktischer Grund für die Zerlegung von Editionsakteuren und Menüauswahlverwalter besteht in der Möglichkeit, bei Verwendung desselben Editionsakteurs mehrere unterschiedliche plantypspezifische Menüakteure zu definieren. Sicherlich wäre dies auch möglich durch Definition entsprechender Unterklassen einer übergeordneten Editorklasse, die diesen Editionsakteur allgemein definiert. Darüber hinaus gibt es jedoch einen weiteren zwingenden Grund, der mit der Rolle des Dokumenten–Editors bei der Bearbeitung von Grafikrahmen zusammenhängt.

Zuvor soll jedoch von Bild–A9 ausgehend zunächst näher auf das Problem der Umschaltung zwischen verschiedenen Bereichszuständigen bei einem Wechsel des Dialogzustandes eingegangen werden. Bislang ist dies so gelöst, daß sich ein neuer Editor beim Steuerakteur anmelden muß, um Adressat der nächsten Benutzeraufträge zu werden, und sich nach Beendigung der Edition wieder abmelden muß. Der Steuerakteur von Interleaf V stellt diese Funktionalität jedoch nicht zur Verfügung, so daß hierfür ein anderer Weg gefunden werden muß.

Betrachtet man zunächst die Reihenfolge, in der die Editoren aufgerufen werden (Bild–A6), so erkennt man, daß es eine Aufrufschichtung gibt. Ausgehend vom Topleveleditor wird ein Editor einer niedrigeren Ebene aufgerufen, der wiederum einen tiefer gelegenen Editor aufrufen kann. In umgekehrter Reihenfolge wird die Zuständigkeit beim Abschließen der jeweiligen Untergruppenbearbeitung wieder nach oben gegeben. Es ist daher sinnvoll, einen *Dialogzustandsverwalter* einzuführen, der einen Stapel verwaltet, auf dem diese Aufrufreihenfolge, das heißt, die Dialogzustandsinformation, abgelegt ist. Der oberste Eintrag auf diesem Stapel verweist dabei auf den aktuellen Dialogzustand und spezifiziert somit den Editor, der zur Zeit für die Bearbeitung des Planes zuständig ist.

Um in einen neuen Dialogzustand zu gelangen, muß dem Dialogzustandsverwalter eine entsprechende Nachricht geschickt werden, die den neuen Dialogzustand oder den korrespondierenden Editor spezifiziert. Diese Information muß auf dem Dialogzustandsstapel als oberstes Element abgelegt werden und ein entsprechender Editor als Bereichszuständiger aktiviert werden. Um wieder in den ursprünglichen Zustand zu gelangen, beauftragt man den Dialogzustandsverwalter den obersten Eintrag vom Stapel zu entfernen. Entsprechend der nun obersten Stapelinformation muß der ursprünglichen Editor wieder zum Bereichszuständigen für den Grafikrahmen werden.

12.. Der Name Planeditor ist im Kontext mit der allgemeinen Verwendung des Begriffs Editor, der hier eine Einheit aus Editionsakteur und Menüauswahlverwalter beschreibt sicherlich etwas unglücklich gewählt, aber da sich der Begriff ”Planeditor” im Rahmen des SPIKES–Projektes so eingebürgert hat, soll er auch im folgenden verwendet werden.

Bevor die Implementierung des Editorwechsels erklärt werden kann, sind einige Bemerkungen zur Rolle des *Dokument–Editors* in Interleaf V nötig. Jedem Dokument ist genau ein solcher Editor zugeordnet (vgl. Bild–A1 – Dokumenteditionsakteur). Dieser Akteur ist zuständig für alle Menüanforderungen innerhalb der Kopfzeile eines Dokumentes, sowie innerhalb aller Grafik–rahmen desselben Dokuments, wodurch der Begriff Editor bei der hier verwendeten Nutzung etwas irritieren mag. Der Dokumenteditor wird automatisch beim Öffnen eines Dokuments miterzeugt und kann später nicht mehr ausgetauscht werden (siehe [3])[13.].

Setzt man dies als gegeben voraus, muß man einen Trick anwenden, um einen Editorwechsel zu erzielen. Man nutzt die Tatsache aus, daß in Interleaf–V alle wesentlichen Akteure als Objekte im Sinne der objektorientierten Programmierung realisiert sind. Als solche gehören sie einer bestimmten Klasse an, die ihre Attribute und Funktionalität definiert. Desweiteren besteht die Möglichkeit neue Unterklassen zu definieren und die Klasse bereits bestehender Objekte zu verändern (siehe [2]). Dies kann man ausnutzen, um neue Unterklassen der Klasse zu definieren, der der Dokumenteditor angehört (*doc–editor–class*). Dabei wird es für jeden Editortyp genau eine Unterklasse geben. In dieser wird die Menüauswahlverwalterbeschreibung des Dokumenteditors entsprechend der jeweiligen Editorklasse umdefiniert (neue *dg–popup–method*).

Findet auf übergeordneter Betrachtungsebene ein Wechsel des aktuellen Editors gegen einen Editor anderen Typs statt, so bedeutet dies auf der Implementierungsebene, daß nicht das Editor–Objekt ausgetauscht, sondern stattdessen die Klassenzugehörigkeit des aktuellen Editors entsprechend auf die gewünschte Editor–Klasse gesetzt wird. Eine zweckmäßige Vorstellung dabei ist die, daß es nebeneinander eine Reihe von editortypspezifischen Menüauswahlverwaltern gibt, die im Multiplexverfahren realisiert werden. Wird an den bereichszuständigen Dokumenteditor eine Botschaft geschickt, so wird das Objektsystem von Interleaf entsprechend der Klasse des Dokumenteditorobjektes und des Botschafttyps den entsprechenden Akteur anstoßen. Die dafür zuständige Instanz im Objekt–System tritt dabei im Aufbaubild als der ”vom Dialogzustand abhängige Auswähler” in Erscheinung. Vom Dialogzustand ist dieser Auswähler insofern implizit abhängig, als der Dialogzustandsverwalter bei einem Editorwechsel die Klasse des Dokumenteditors wechselt. In der Implementierung ist daher auch die Information auf dem Dialogzustandsstack identisch mit den Verweisen auf die jeweilige Editorklasse[14.].

Dieses Verfahren birgt auch den Grund in sich, warum es eine Trennung zwischen Menüauswahlverwaltern und Editionsakteuren geben muß. Der Wechsel eines Objekts von einer Klasse A zu

13.. Es gibt zwar einen Weg durch Setzen eines Dokumentattributes (der property :doc–editor) quasi einen neuen Editor vor den alten zu setzen, es handelt sich hierbei allerdings um ein undokumentiertes Merkmal von Interleaf V, das erst nach der Implementierung entdeckt wurde. Inwieweit dessen Ausnutzung möglicherweise zu Problemen führen kann bzw. welche Vorbedingungen erfüllt sein müssen, ist aufgrund der fehlenden Dokumentation ebenfalls unbekannt. Eine Ausnutzung dieser Möglichkeit wäre daher nicht nur aufwendig, da vieles umgestellt werden müßte, sondern auch überaus bedenklich.

14.. Zusätzlich zu diesem Stack ist ein zweiter Stack zum Verwalten der im jeweiligen Dialogzustand verwendeten Tastaturtabellen nötig. So muß es möglich sein, in einigen Dialogzuständen die Tastatur völlig zu sperren, das heißt eine leere Tastaturtabelle vorzugeben. Würde dies nicht geschehen, wäre es dem Benutzer möglich, auch in Grafikgruppen, in denen dies verboten ist, Texte einzugeben.

einer Klasse B kann als Strukturvarianz gedeutet werden. Die Teile des Objektes, die bei A definiert werden, bei B aber fehlen müssen sind nach dem Klassenwechsel nicht mehr definiert. Wird die Klasse jetzt zurück gewechselt, sind diese Teile zwar wieder definiert, aber im Falle, daß es sich bei diesen Teilen um Speicher gehandelt hat, kann nicht davon ausgegangen werden, daß der Inhalt des Speichers dem Ausgangszustand entspricht. Es ist sogar sehr wahrscheinlich, daß dem nicht so ist, da in der Vorstellung das ursprüngliche Objekt zerstört, durch ein neues Objekt vom Typ B ersetzt wurde, das anschließend wieder zerstört wurde und zuletzt wieder durch ein Objekt vom Typ A ersetzt wurde[15.]. Während diese Vorstellung beim Dokument–Editor nicht schädlich ist, da keine Information über die einzelnen Dialogzustände hinweg gerettet werden muß (die Menühistory darf bei einem Wechsels des Dialogzustandes verfallen), muß dies bei den Editionsakteuren nicht der Fall sein. Sollen die Editionsakteure dauerhaft Daten während der gesamten Bearbeitung des Planes speichern können, wie z.B. der Planeditor die Undoinformation (vgl. Kapitel 1.2.4), dürfen sie nicht als Teil des Dokument–Editors, sondern müssen als eigenständige Objekte realisiert werden.

Zu den weiteren Änderungen im implementierungsnahen Aufbaumodell zählt die Sonderstellung des Planeditors unter den Editionsakteuren. Der Planeditor ist als Editionsakteur auf oberster Planebene der einzige Editionsakteur, der bei jeder Bearbeitung des Planes benötigt wird. Wie bereits erwähnt, fällt in seinen Zuständigkeitsbereich unter anderem die Aufgabe, Editionsschritte rückgängig zu machen (Undo)[16.]. Da es sicher keinen Grund gibt, diese Möglichkeit dem Benutzer nur auf oberster Planebene zur Verfügung zu stellen, müssen auch Untergruppeneditoren diese Funktionalität anbieten. Anstatt nun für jeden Editionsakteur einen eigenen Undo–Akteur vorzusehen, ist es sinnvoll, das idealisierte Konzept der hierarchischen Editoren zu lokkern und die Untergruppeneditoren den Planeditor mitbenutzen zu lassen.

Im Gegensatz zum Planeditor, der durch diese Nutzung dauerhaft während der ganzen Bearbeitung eines Planes zur Verfügung stehen muß, ist dies bei den übrigen Editionsakteuren nicht der Fall. Diese werden nur für die Dauer einer ihrem Typ entsprechenden Untergruppenedition benötigt. Aus diesem Grunde ist es sinnvoll, diese Editionsakteure nur dann zu erzeugen, wenn sie auch benötigt werden und danach wieder zu löschen. In einem solchen Falle wird der zukünftige Auftraggeber eines Editionsakteurs, zum Beispiel der Akteur einer Symbolgrafik, nachdem er einen Anforderungsauftrag zur Bearbeitung erhalten hat, zunächst einen Editor geeigneten Typs erzeugen lassen. Anschließend wird, wie bereits beschrieben, die Edition gestartet. Hierbei wird der Initialisierungsakteur allerdings nicht, wie beim allgemeinen Modell beschrieben, sich beim Steuerakteur anmelden, sondern gemäß vorangehender Überlegung den Dialogzustandsverwalter zum Ändern der Klasse des bereichszuständigen Dokumenteditors auffordern, wozu er ihm

15.. Bei der Realisierung der SPIKES–Objekte (das Interleaf–V Objekt–System wurde innerhalb des SPIKES–Projektes erweitert) bleiben die Daten bei Aktionen wie oben beschrieben zwar bestehen, aber dies muß nicht in allen objektorientierten System so sein.

16.. Zwar verfügt Interleaf–V bereits über Methoden zum Rückgängigmachen von Editionsschritten, aber diese Funktionalität reicht im Zusammenhang mit SPIKES–Editoren nicht aus.

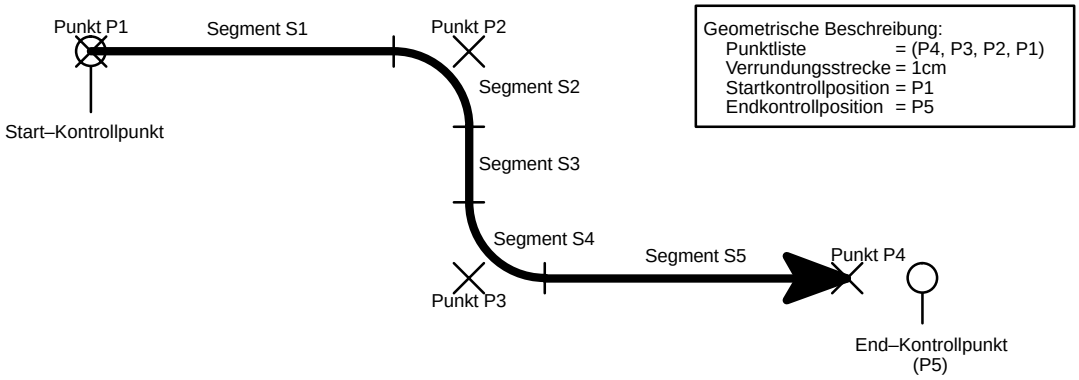
einen Verweis auf die entsprechende Klasse übergeben muß[17]. Nach Abschluß der Untergruppenbearbeitung und Zurücksetzen der Klassenzugehörigkeit des Dokumenteditors wird der Editionsakteur nicht mehr benötigt und kann aus dem System entfernt werden. Dies erledigt wiederum der Auftraggeber. Da es durch dieses Konzept nicht nötig ist, alle Editoren zentral zu verwalten, wird die leichte Austauschbarkeit der Editoren erhöht.

17.. Bei der Implementierung der SPIKES–Editoren ist für den Dialogzustandsverwalter kein eigenes Objekt vorgesehen. Stattdessen ist er als Erweiterung des Dokument–Editors realisiert. Aus diesem Grunde werden alle Aufforderungen zu einem Wechsel des Dialogzustandes an den Dokumenteneditor selbst gesandt.

1.2.5 Beispiel: Das Erzeugen und Bearbeiten von Kantenstücken

Bevor im folgenden Kapitel auf die Hintergründe des Kantenstückkonzeptes eingegangen wird, soll zunächst das abstrakte Editionsmodell an dem konkreten Beispiel des Erzeugens und Bearbeitens von Kantenstücken verdeutlicht werden.

Kantenstücke sind Plankomponenten, deren Basisgrafikgruppe aus einem verrundeten Linienzug besteht. Dieser ist aus einer alternierenden Folge von Linienstücken und Kreisabschnittsbögen, den *Segmenten* des Linienzuges, aufgebaut. Per Konvention gilt, daß das erste und letzte Segment grundsätzlich eine Linie sein muß. Ebenso muß zwischen allen aufeinanderfolgenden Bögen eine Linie existieren. Diese Linien dürfen allerdings beliebig kurz sein.

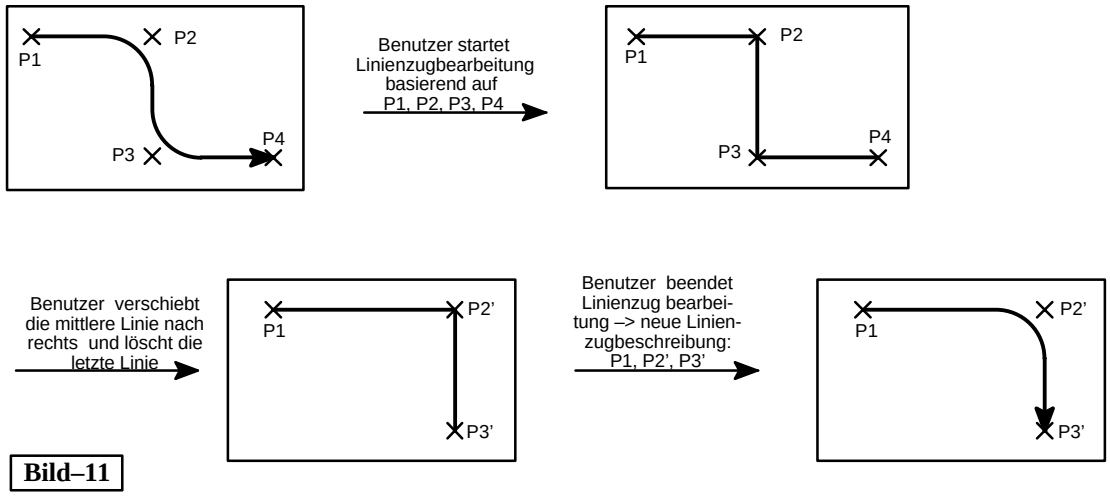


Bild–10: Aufbau eines (gerichteten) Kantenstückes

Der Linienzug wird durch den Verrundungsparameter und eine Folge von Punkten beschrieben, die Anfangs– und Endpunkt sowie die Eckpunkte des Linienzuges definieren. Die Eckpunkte liegen dabei wegen der Verrundung nicht auf dem Linienzug selbst. Von Eckpunkten zu reden macht jedoch trotzdem Sinn, da ein solcher Punkt genau den Schnittpunkt der beiden Geraden definiert, die durch das vorangehende und nachfolgende Linienstück beschrieben werden. Da es bei SPIKES–Editoren vorgesehen ist, daß Kantenstücke auf den Rand eines Knotens hin angezogen werden können, ist es möglich, daß sich dadurch Anfangs– und/oder Endpunkt des Linienzuges und damit auch der Punktfolge ändern. Um eine logische Auswertung des Planes durch eine vorangehende grafische Analyse des Planes zu ermöglichen, soll es dem Benutzer aber nur innerhalb eines vorgegebenen Rasters möglich sein, die Punkte zu plazieren oder Plankomponenten zu positionieren, um so eindeutige grafische Kontakte herzustellen. Da es durch die Anziehung des Kantenstückes auf einen Knoten mit beliebigem Rand möglich ist, daß ein Endpunkt des Linienzuges aus dem Raster gerät, wurde für jedes Ende des Linienzuges in Form eines unsichtbaren Punktes, der grundsätzlich im Raster liegt, ein zusätzliches Grafikattribut eingeführt. Diese Punkte werden als Start– und Endkontrollpunkt (*start–control* und *fin–control*) bezeichnet. Im Gegensatz zu diesen handelt es sich bei der zuvor beschriebenen Punktfolge nicht um eigene Gra-

fikobjekte, sondern lediglich um eine geometrische Beschreibung, die als Attribut eines Kantenstückakteurs in dessen lokalem Speicher abgelegt ist.

Bild–A10 stellt ein Petrinetz dar, in dem nebeneinander der Ablauf zum Erzeugen und zum Bearbeiten eines solchen Kantenstückes dargestellt ist. Im folgenden soll zunächst das Bearbeiten eines Kantenstückes betrachtet werden, wobei das folgende Bild–11 die Bearbeitung illustriert. Gegeben ist ein Kantenstück bestehend aus dem durch die Punkte P1 bis P4 definierten verrundeten Linienzug und einer Pfeilspitze.



Bild–11

Hat der Benutzer das Kantenstück ausgewählt und den entsprechenden Menüeintrag zum Bearbeiten eines Kantenstückes ausgewählt, wird der zuständige Kantenstückakteur angestoßen. Dieser wird, um die Bearbeitung gegebenenfalls rückgängig zu machen, den Planakteur veranlassen die nötige Undo–Information zu vermerken (siehe Kapitel 1.4.1 – Undo–Konzept). Danach wird das Kantenstück gegebenenfalls aus einer aus mehreren Kantenstücken bestehenden Baumstruktur ausgegliedert, um anschließend von dieser Struktur losgelöst behandelt werden zu können (siehe Kapitel 1.3 – Kantenstückkonzept). Diese beiden Punkte sind an dieser Stelle zunächst nur der Vollständigkeit halber aufgeführt und werden in den entsprechenden Kapiteln detailliert behandelt.

Bei der Bearbeitung oder dem Erzeugen eines Linienstückes wird nicht direkt der verrundete Linienzug bearbeitet, sondern statt dessen eine Hilfsgrafik in Form eines nicht verrundeten Linienzuges, dessen Eckpunkte die Folge von Punkten definiert, die das Kantenstück beschreiben. Der ursprüngliche verrundete Linienzug, sowie direkt an den Linienzug geknüpfte Attributgrafiken des Kantenstückes, wie zum Beispiel die Pfeilspitze, werden daher vor der Bearbeitung vom Kantenstückakteur gelöscht. Für die eigentliche Edition des Kantenstückes ist ein spezieller Editionsakteur für Linienzüge erforderlich, der zu diesem Zweck vom Kantenstückakteur erzeugt wird. Der Editionsakteur erhält nun vom Kantenstückakteur den Auftrag, basierend auf der bisherigen Punktliste des Kantenstückes eine Bearbeitung derselben durch den Benutzer zu ermöglichen.

Die Funktionsweise des Editionsakteurs für Linienzüge ist in Bild–A11 gezeigt. Nachdem der Editionsakteur einen Editionsauftrag erhalten hat, notiert er Auftraggeber und Abschlußbotschaft um die Punktliste nach der Bearbeitung zurückzusenden. Danach beauftragt der Editionsakteur den Dialogzustandsverwalter einen Wechsel in den Dialogzustand für Linienzüge vorzubereiten. Um abgeschlossen von der übrigen Grafik des Planes die Hilfsgrafik bearbeiten zu können, wird diese in einer eigens zu diesem Zweck erzeugten Grafikgruppe erstellt. Wurde eine leere Punktliste übergeben, was nur beim Erzeugen von Kantenstücken vorkommen kann, definiert die dem Mauszeiger am nächsten gelegene Rasterposition den Startpunkt eine ersten Linie deren Endpunkt direkt anschließend durch den Benutzer interaktiv festgelegt werden muß. In allen anderen Fällen wird entsprechend der übergebenen Punktliste eine Reihe von Hilfslinien erstellt.

Im Beispiel besteht dieser Linienzug aus drei Linien, die die Punkte P1 bis P4 miteinander verbinden. Nach dem Erstellen der Hilfsgrafik befindet sich das Editorsystem im Dialogzustand ”Linienzugbearbeitung” und der Benutzer hat nun die Möglichkeit, diesen Linienzug zu bearbeiten. Dabei kann er neue Linienstücke erzeugen, bestehende Linienstücke stauchen, dehnen, löschen oder bewegen, sowie Anfang und Ende vertauschen, was eine Richtungsumkehr der Kante nach Abschluß der Edition bewirkt. Wird die Edition auf Wunsch des Benutzer beendet, bestimmt der Abschlußakteur des Editionsakteurs durch Auswertung der Hilfslinien die neue Punktfolge. Ist diese Punktfolge zulässig, das heißt, wird durch diese Punktfolge mindestens eine Linie mit einer Länge größer Null definiert, stellt der Editionsakteur den ursprünglichen Dialogzustand wieder her und sendet die ermittelte Punktfolge mit der Abschlußbotschaft an seinen Auftraggeber zurück. Andernfalls kann der Benutzer den Dialogzustand ”Linienzugbearbeitung” solange nicht verlassen, bis durch Bearbeitung der Hilfsgrafik eine zulässige Punktfolge definiert wurde.

War der Auftraggeber ein Kantenstückakteur, so wird dieser die Punktliste notieren und anschließend versuchen, aufgrund der ermittelten Punktfolge (im Beispiel P1, P2’ und P3’) und dem Verrundungsparameter einen neuen verrundeten Linienzug zu erstellen. Wenn dies gelingt, wird er prüfen, ob eine Anziehung des Linienzuges auf berührte Knotensymbole erforderlich ist und diese gegebenenfalls durchführen. Die grafischen Parameter des Linienzuges wie Liniendicke und Farbe sind gemäß den ursprünglichen Werten anzupassen. Zusätzlich sind die typspezifischen vom Linienzug abhängigen, nötigen Attributgruppen neu zu erzeugen. Für alle Kantenstücke gleichermaßen sind hierbei lediglich die Kontrollpunkte definiert. Handelt es sich um ein gerichtetes Kantenstück, kann zusätzlich eine Pfeilspitze am Endpunkt erforderlich sein, wohingegen ungerichtete Kantenstücke nicht über eine solche Spitze verfügen.

Bei der Implementierung gibt es hierzu eine gemeinsame Oberklasse für SPIKES–Kantenstücke, die alle Attribute und Funktionen beschreibt, die für alle Kantenstücke gemeinsam definiert sind. Dazu gehört die Erstellung des Linienzuges basierend auf der Punktliste, sowie das Anlegen von Kontrollpunkten. Für speziellere Kantenstücke, wie zum Beispiel gerichtete Kantenstücke, sind Unterklassenakteure definiert, die diese Eigenarten berücksichtigen und beispielsweise dafür

sorgen, daß zusätzlich eine Pfeilspitze erzeugt wird. Die Ausnutzung dieses Klassenkonzeptes spielt bei den unterstützten Symboltypen innerhalb der SPIKES–Editoren–Familie eine große Rolle.

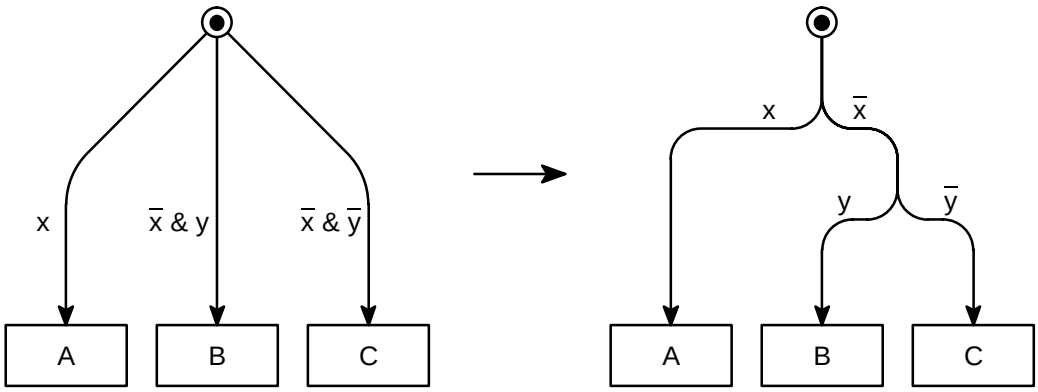
Konnte basierend auf der geometrischen Beschreibung kein verrundeter Linienzug erstellt werden, wird die Edition erneut gestartet. Dabei wird die Punktliste übergeben, die den Linienzug bis zu der Position beschreibt, bis zu der eine Verrundung durchgeführt werden konnte. Liegt kein Fehler vor, ist die Bearbeitung hiermit abgeschlossen. In jedem Fall wird der Editionsakteur gelöscht, so daß eine erneute Edition einfach durch einen wiederholten Bearbeitungsauftrag des Kantenstückakteurs an sich selbst eingeleitet werden kann.

Das Erzeugen eines Kantenstückes ähnelt in vielen Punkten dem Bearbeiten. Das Erzeugen wird eingeleitet durch einen Auftrag an den Klassenakteur, der für den gewünschten Kantenstücktyp verantwortlich ist. Da beim Erzeugen noch keine Punktliste definiert ist, übergibt der Klassenakteur dem Editionsakteur keinerlei Punkte, sondern überläßt diesem das Definieren eines ersten Startpunktes, wie zuvor beschrieben. Nach Abschluß der Linienzugedition, die sich bis auf das Erstellen der ersten Linie nicht vom Bearbeiten einer bereits bestehenden Punktfolge unterscheidet, wird der Klassenakteur einen Kantenstückakteur erzeugen, der gemäß den planspezifischen Defaultwerten zu initialisieren ist. Dem neu erzeugten Kantenstückakteur wird daraufhin die ermittelte Punktliste übergeben, der daraus wie beim Abschluß der Bearbeitung eines bereits bestehenden Kantenstückes eine neue Kantenstückgrafik erzeugt.

1.3 Das Kantenstückkonzept

1.3.1 Kantenbäume

Wie eingangs beschrieben, dienen Strukturpläne dazu, Beziehungen innerhalb einer Menge von Objekten darzustellen. Die Objekte werden als flächenhafte Knoten dargestellt. Die Beziehungen unter diesen Objekten werden durch Kanten dargestellt, das heißt, durch verrundete Linienzüge, die diese Knoten miteinander verbinden. Eine Kante verläuft hierbei eindeutig von einem Start– zu einem Endknoten. Logisch betrachtet ist eine Kante ein unteilbares Objekt, daß eindeutig identifiziert werden kann. Bei der grafischen Darstellung kann es sinnvoll sein, Kanten, die auf denselben Knoten hin zulaufen und auch sonst semantisch verträglich sind, ab einem gewissen Punkt gemeinsam darzustellen. Aus mehreren unabhängigen Kanten wird grafisch gesehen ein zusammenhängendes Gebilde, ein *Kantenbaum.*



Bild–12: Entstehungsbeispiel eines Kantenbaumes

Wie bei jedem Baum gibt es auch bei einem Kantenbaum eine *Wurzel.* Als solche wird das erste gemeinsame Stück, das die Kanten ausgehend von dem gemeinsamen Knoten beschreiben, bezeichnet. Daran anschließend verzweigt der Baum immer weiter, bis man letztlich zu nicht mehr verzweigenden Abschnitten gelangt. Diese sollen als *Blätter* bezeichnet werden. Ausgehend von einer Verzweigung wird man den wurzelseitigen Abschnitt als *Stamm* und die blattseitigen Abschnitte als *Äste* bezeichnen. Danach ist die Wurzel als Sonderfall eines Stammes und ein Blatt als Sonderfall eines Astes anzusehen. Der Teil eines Baumes, der ausgehend von einem beliebigen Stamm bis zu den darauf wachsenden Blättern definiert ist, soll als *Teilbaum* bezeichnet werden.
Logisch verändert sich durch die Darstellung als Baumgrafik nichts. Das Gebilde beschreibt dieselbe Struktur mit derselben Anzahl von Kanten wie zuvor.

Es gibt zwei Gründe Kantenbäume zu verwenden. Zum einen reduziert es erheblich den Zeichenaufwand, wenn man nicht für jede Kante einen eigenen kompletten Linienzug vom Start– zum Endknoten zeichnen muß, sondern, wenn man an einem beliebigen Zwischenpunkt im Baum anknüpfen kann. Zum anderen erhöht es die Übersichtlichkeit eines Planes einfach dadurch, daß weniger Grafik in einem Plan benötigt wird, um dasselbe darzustellen. Die Gefahr, durch viele eng nebeneinander verlaufende Linien verwirrt zu werden, wird minimiert. Darüber hinaus besteht die Möglichkeit, durch Zusammenlegen von Kanten mit gemeinsamen Attributen die Planstruktur übersichtlicher darzustellen. Im Plan äußert sich dies darin, daß gemeinsame Abschnitte mit den gemeinsamen Attributen beschrieben werden können. Ein Attribut, das einem bestimmten Ast im Baum zugeordnet ist, gilt für alle durch diesen Ast abschnittsweise definierten Kanten. In Bild–12 gilt so z.B. die Schaltbedingung, daß x unwahr ist, sowohl für die auf Transition B, als auch für die auf C hinlaufende Kante.

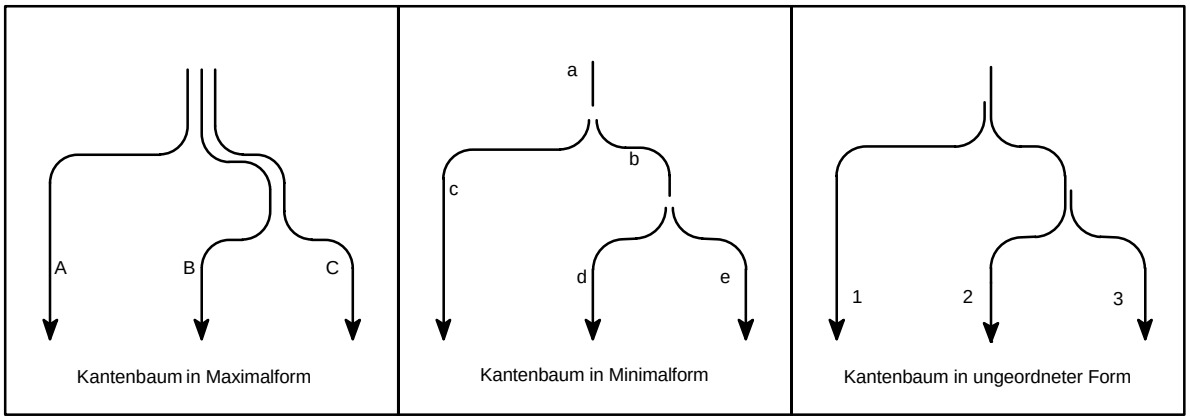
Entsprechend der den SPIKES–Editoren zugrunde liegenden Editionsphilosophie hat man sich daher entschieden, die Erstellung solcher Kantenbäume zugunsten der Benutzerfreiheit zu unterstützen. Aufgrund der erlaubten Attributierung von Kanten aber auch aus zeichnerischen Gründen sind dadurch, wie im folgenden gezeigt wird, eine Reihe von Problemen zu lösen, die bei den bislang bestehenden Editoren entweder ignoriert werden oder durch schlichtes Verbieten der Erstellung von Kantenbäumen umgangen werden.

1.3.2 Realisierungsmöglichkeiten für Kantenbäume

Um einen beliebigen Kantenbaum grafisch zu realisieren, existieren beliebig viele Wege, wenn man davon ausgeht, daß eine Kante durch eine beliebige Anzahl, mindestens aber ein Kantenstück, beschrieben werden soll. Ein solches Kantenstück besteht dabei, wie schon zuvor beschrieben, aus einem verrundeten Linienzug und gegebenenfalls zusätzlichen grafischen und logischen Attributen. Es sollen im folgenden drei Realisierungsformen von Kantenbäumen unterschieden werden.

- Kantenbäume in Maximalform
- Kantenbäume in Minimalform
- Kantenbäume in ungeordneter Form

So zeigt Bild–13 dreimal den gleichen Kantenbaum, jedoch entsprechend obiger Einteilung unterschiedlich realisiert.

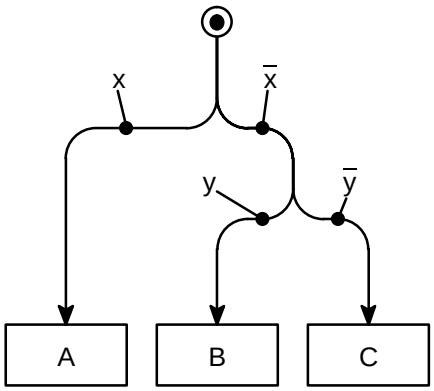


Bild–13: Realisierungsformen von Kantenbäumen

Kantenbäume in Maximalform:

In der *Maximalform* entspricht jeder Kante genau ein Kantenstück. Die Bäume erhält man durch Übereinanderlegen der Kantenstücke. Die logische Auswertung der Pläne ist in dieser Form problemlos, da Anfangs– und Endknoten einer Kante eindeutig durch Start– und Endpunkt des Linienzuges des entsprechenden Kantenstückes identifiziert werden können. Ebenso sind alle Attribute einer Kante durch die Menge aller Attribute des einen Kantenstückes spezifiziert. Dies bewirkt, daß grafische, logisch unverzichtbare Attribute, die einem gemeinsamen Stamm aus mehreren Kanten zugeordnet sein sollen, ebenso wie die Linienzüge, übereinander liegend realisiert sein müssen. Wäre dies nicht so, würde abhängig davon welche Kante der Benutzer ausschneiden oder bearbeiten würde, das gemeinsame Attribut mit betroffen oder nicht. Da dem Benutzer andererseits nicht zugemutet werden kann, dieselbe Edition für alle betroffenen Kantenstücke zu wiederholen, muß es einen Mechanismus geben, der diese grafische Redundanz der

Attribute automatisch herstellt und damit dafür sorgt, daß der Benutzer den Eindruck hat, als würde er nur ein Attribut, nämlich das des jeweiligen Stammes ändern. Weit schwerwiegender jedoch ist die Frage, wie entschieden werden soll, ob ein Attribut dem gemeinsamen Stamm oder dem sich anschließenden Ast zugeordnet werden soll. Sicherlich gibt es Fälle, wonach eine überwiegende Mehrheit von Menschen sich für ein und dieselbe Zuordnung entschließen würde, aber problemlos könnten Zweifelsfälle kreiert werden, in denen dies nicht möglich ist. Es ist daher nicht sinnvoll mit hohem Aufwand nach formalen Beschreibungen zu suchen, die letztlich das Problem doch nicht zufriedenstellend lösen können. Man müßte die Attribute daher nicht lediglich einem Kantenstück, sondern einer ganz bestimmten Position auf diesem Kantenstück zuordnen (vgl. Bild–14). Die Edition dieser Attribute müßte dies entsprechend unterstützen (vgl [8]).



Bild–14: Explizite Zuordnung von Kantenstückattributen zu Linienzugpositionen

Wird der Linienzug danach erneut durch den Benutzer bearbeitet, ist es möglich, daß dadurch diese Zuordnung wiederholt werden muß, da die ursprünglichen Zuordnungspositionen der Attribute nicht mehr auf dem neuen Linienzug liegen müssen. Gegen eine automatische Neuzuordnung spricht auch in diesem Falle die Vielzahl mehrdeutiger Situationen, die denkbar sind.

Probleme ergeben sich beim Auswählen von Kantenstücken durch die Wahl von Interleaf–V als Implementierungsplattform. Drückt der Benutzer die Auswahltaste der Maus in einer Situation, in der sich der Mauszeiger über einem Stamm aus mehreren übereinanderliegenden Kantensymbolen befindet, sollen sicherlich alle diese Kantensymbole ausgewählt werden und nicht etwa willkürlich nur irgendein Kantensymbol. Damit der Benutzer erkennen kann, welche Plankomponenten ausgewählt sind, und welche nicht, müssen sich ausgewählte von nicht ausgewählten Symbolen grafisch unterscheiden. Bei Interleaf–V ist dies so gelöst, daß ausgewählte Grafikobjekte in bestimmen Intervallen automatisch blinken. Leider gibt es hier das Problem, daß dies nur dann der Fall ist, wenn eine ungerade Anzahl von Grafikobjekten übereinander liegt. Im Falle maximaler Kantenbäume heißt dies, daß der Benutzer Stämme aus 2, 4, 6 usw. Kantensymbolen nicht als ausgewählt erkennen kann, sondern bestenfalls am Blinken anschließender Äste auf die Auswahlmenge schließen kann. Möglichkeiten, dieses Verhalten zu korrigieren, gibt es nicht.

Darüber hinaus ist ein direktes Erstellen von Kantenbäumen in Maximalform durch entsprechendes Erzeugen der Kantenstücke durch den Benutzer gerade bei großen Kantenbäumen schwierig.

Dies liegt zum einen daran, daß der gesamte gemeinsame Stamm nachgezeichnet werden muß, zum anderen daran, daß das ganze Kantenstück von der Wurzel bis zum Blatt einfach zu groß sein kann, um als Ganzes sinnvoll auf dem Bildschirm dargestellt zu werden. Um ein durchgängiges Arbeiten entsprechend diesem Konzept zu ermöglichen, muß es Mechanismen geben, die die Erstellung von Kantenbäumen in Maximalform unterstützen bzw. die Erstellung von anderen Bäumen verhindern.

Kantenbäume in Minimalform:

Bei Kantenbäumen in *Minimalform* kann man eine Reihe dieser Probleme umgehen. Bei dieser Form wird jeweils ein Ast des Kantenbaumes durch genau ein Kantenstück dargestellt. Aus diesem Grunde wird das Minimalformkonzept auch als *Kantenstückkonzept* bezeichnet. Eine Kante ist dabei eindeutig definiert durch eine lückenlose Aneinanderreihung von einzelnen Kantenstücken. Umgekehrt definiert ein Kantenstück als Repräsentant eines Astes denselben Abschnitt mehrerer Kanten, wenn es sich bei dem Kantenstück um den Stamm mehrerer anderer Äste handelt. Die Attribute einer Kante sind dabei gleich der Summe aller Attribute der die Kante definierenden Kantenstücke. Alle Probleme durch das Auswählen und Bearbeiten übereinander liegender Grafikgruppen treten beim Arbeiten mit Kantenbäumen in Minimalform nicht auf. Will der Benutzer eine oder mehrere Kanten löschen, so löscht er einfach die betroffenen Äste. Ebenso einfach ist es, einzelne Äste besonders durch Farbe oder Strichstärke hervorzuheben. Zum Bearbeiten ganzer Unterbäume sind spezielle Mechanismen vorzusehen, die entsprechend alle Kantenstücke des Unterbaumes berücksichtigen. Auch bei diesem Konzept muß sichergestellt werden, daß keine Bäume anderer Form, also nur Minimalformbäume, im Plan auftreten können.

Kantenbäume in ungeordneter Form:

Alle anderen Realisierungen eines Kantenbaumes sollen unter dem Begriff *ungeordnete Formen* zusammengefaßt werden. Durch grafische Analyse der einzelnen Linienzüge ist es möglich, die einzelnen hierdurch definierten Kanten, sowie deren Start– und Endknoten zu bestimmen. Für die Zuordnung der grafischen logisch unverzichtbaren Attribute gilt das gleiche, wie für die Kantenbäume in Maximalform. Anders als bei diesen können direkte Operationen wie Ausschneiden oder Bewegen beim Arbeiten mit Kantenbäumen ungeordneter Form jedoch zu zum Teil überraschenden Ergebnissen führen. Möchte beispielsweise ein Benutzer in dem in Bild–13 rechts dargestellten Kantenbaum die mittlere Kante löschen, so wird er höchstwahrscheinlich Kantenstück 2 auswählen und ausschneiden und dadurch das sicher ungewollte Ergebnis erzielen, daß die rechte und die linke Kante ”irgendwie in der Luft hängen” würden und nachbearbeitet werden müßten. Ein Verfahren, das diese Nacharbeiten automatisch ausführen würde, würde bei effektiver Realisierung letztlich wieder auf eine Maximal– oder Minimalform hinauslaufen. Wäre die Maximalform direkt verwendet worden, hätte es gereicht das Kantenstück B auszuschneiden.

Wäre die Minimalform verwendet worden, so hätte Kantenstück d ausgeschnitten und die Kantenstücke b und e automatisch zu einem Kantenstück verschmolzen werden müssen.

Entscheidung für das Minimalformkonzept:

Bei der Realisierung der SPIKES–Editoren wurde das Minimalformkonzept ausgewählt. Mehrere Gründe gaben dafür den Ausschlag. Die Arbeit mit ungeordneten Kantenbäumen scheidet von vorneweg aus, da hier nahezu jede nachträglich Edition der Bäume einen verhältnismäßig großen Aufwand darstellt. Ein Vergleich der beiden verbleibenden Konzepte bezüglich der zu erwartenden Editionsgeschwindigkeit ist kaum möglich. Bei beiden Verfahren soll der Benutzer beliebige Kanteneditionen vornehmen dürfen, so daß danach Schritte nötig sind, um die jeweilige Baumform zu erhalten bzw. gegebenenfalls planweit wiederherzustellen. Bei der Maximalform kommen darüber hinaus die Sonderbehandlungen im Zusammenhang mit der Bearbeitung der redundanten Grafiken hinzu. Wieviel Zeit dies im einzelnen benötigt, dürfte im wesentlichen vom Geschick bei der Implementierung abhängig sein.

Den Ausschlag für das Minimalformkonzept gab letztlich die einfache klare Zuordnung von Ästen und Kantenstücken, die die für den Benutzer umständliche Arbeit des festen Definierens von Zugehörigkeitspositionen für frei positionierbare Grafikattribute vermeidet. Ein positiver Nebeneffekt dieses Konzeptes liegt in der Tatsache begründet, daß es bei Kantenbäumen in Minimalform keinerlei grafische Redundanz gibt und somit Speicherplatz eingespart werden kann.

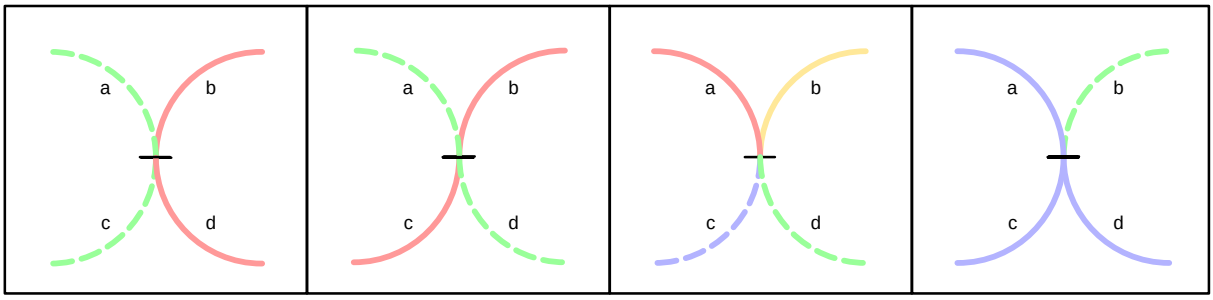
Auswirkungen des Minimalformkonzeptes:

Der Benutzer soll durch die Entscheidung für das Minimalformkonzept in seinen Möglichkeiten nicht eingeschränkt werden. Es soll ihm möglich sein, an beliebigen Stellen des Planes beliebig geformte Kantenstücke zu erzeugen, ganze Kantenstückbäume oder Teilbäume zu kopieren, auszuscheiden, einzufügen oder zu bewegen. Da durch solche Freiheiten selbstverständlich auch Bäume entstehen können, die nicht der Minimalform entsprechen, muß als Konsequenz davon nach jedem dieser Editionsschritte dafür gesorgt werden, daß alle betroffenen Kantenstückbäume wieder in Minimalform gebracht werden. Die Problematik läßt sich dabei auf zwei grundsätzliche Aufgaben zurückführen:

- Das Einfügen von Kantenstücken oder ganzen Kantenstückbäumen in die bestehende Kantenstückbaumstruktur des Planes (siehe Kapitel 1.3.3)
- Das Herauslösen von Kantenstücken oder Teilbäumen aus bestehenden Kantenstückbäumen (siehe Kapitel 1.3.4)

Zusätzlich ist eine explizite Beschreibung der zu einem bestimmten Editionszeitpunkt existierenden Kantenstückbaumstruktur des Planes nötig. Allein die grafische Analyse des Planes ist nicht

immer ausreichend. So kann es durch einen unglücklichen Editionsschritt zu Situationen kommen, die nicht eindeutig interpretierbar sind.



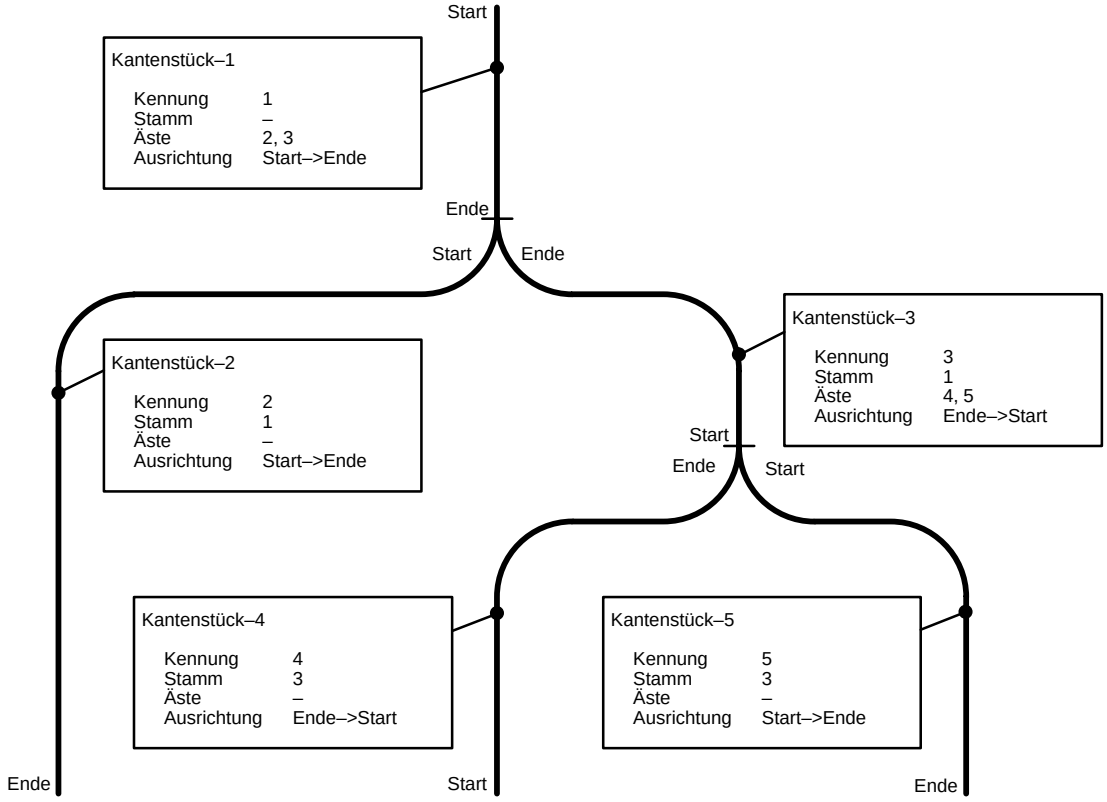
Kantenstücke, die zu einer Kante oder einem Baum gehören, sind in derselben Farbe dargestellt

Bild–15: mehrdeutige Kantenstücksituationen

Bild–15 zeigt ein solches Beispiel, in dem vier Kantenstücke in einem Punkt aufeinandertreffen. Eine Vielzahl von Interpretationsmöglichkeiten existiert, von denen sicherlich nur die beiden ersten syntaktisch korrekt sind, insofern jeweils zwei Kantenstücke zu einem zusammengefaßt werden können, so daß zwei sich berührende bzw. kreuzende Kanten entstehen, wie in den beiden linken Situationen in Bild–15. Vielleicht aber sollen drei der Kantenstücke einen Baum beschreiben und durch einen nachträglichen unglücklichen Editionsschritt ist die Situation mehrdeutig geworden, wie dies in der rechten Situation dargestellt ist. Auf keinen Fall kann es gewollt sein, daß durch solche Schritte, aus zuvor eindeutigen Kantenstückbaumstrukturen aufgrund einer willkürlichen erneuten Auswertung völlig andere Ergebnisse entstehen. Im Beispiel heißt dies, daß es nicht gewollt sein kann, daß aus einem bisherigen Kantenstückbaum aus drei Kantenstükken und einem unglücklich positionierten neu hinzugekommenen vierten Kantenstück durch Zusammenfassen von jeweils zwei Kantenstücken zu einem Kantenstück einer der beiden ersten Fälle wird. Solche Fehlinterpretationen können vermieden werden, wenn irgendwo in jedem Plan die bestehende Baumstrukturinformation aller Kantenstücke abgelegt ist. Immer wenn Kantenstücke eingefügt oder herausgetrennt werden müssen, muß dies in Einklang mit der bisher bestehenden Strukturinformation geschehen, die dabei den Änderungen gegebenenfalls angepaßt werden muß. Im rechten Situationsbild des Beispieles heißt dies, daß der durch die Kantenstücke a, c und d definierte Baum in dieser Strukturinformation vermerkt ist. Das später hinzukommende Kantenstück b läßt sich nicht in diesen Baum eingliedern und bleibt isoliert. Darüber hinaus ermöglicht diese Information eine schnellere Analyse der beschriebenen Planstruktur, sei es beim Einfügen von Kantenstücken oder auch bei der syntaktischen Auswertung des Planes.

Ein einfacher und effektiver Weg diese Information zu hinterlegen, besteht darin, bei jedem Kantenstück Referenzen auf den eventuellen *Vorgänger*, das heißt den Stamm (*trunk*), sowie alle *Nachfolger*, das heißt die Äste (*branches*), zu vermerken. Die Referenzen werden mit Kennungen entsprechend dem in Kapitel 1.1.2 beschrieben Konzept der SPIKES–ID's realisiert. Zusätzlich ist es nötig zu vermerken, welches Ende eines Kantenstückes ast– bzw. wurzelseitig liegt. Hierzu ist ein *Wachstumsattribut* definiert (in Bild–16 Ausrichtung (*grow–direction*) genannt), das

angibt, ob der Startpunkt des Kantenstückes wurzelseitig liegt, es also vom Start– zum Endpunkt wächst, oder ob der Endpunkt des Kantenstückes wurzelseitig liegt, das Kantenstück also vom End– zum Startpunkt in Richtung Blätter wächst. Welcher der beiden Endpunkte eines Kantenstückes dabei Startpunkt und welches der Endpunkt ist, ist durch die Editionsreihenfolge bestimmt. Für Kantenstücke, die weder Stamm noch Äste besitzen, die also Wurzel und Blatt in einem sind, ist das Wachstumsattribut nicht definiert. Diese Angaben reichen aus, um die Beziehungen aller Kantenstücke innerhalb eines Baumes vollständig zu beschreiben. Bild–16 zeigt dazu ein Beispiel eines Kantenstückbaumes aus ungerichteten Kantenstücken.



Bild–16: Referenzierung zwischen Stamm– und Astkantenstücken

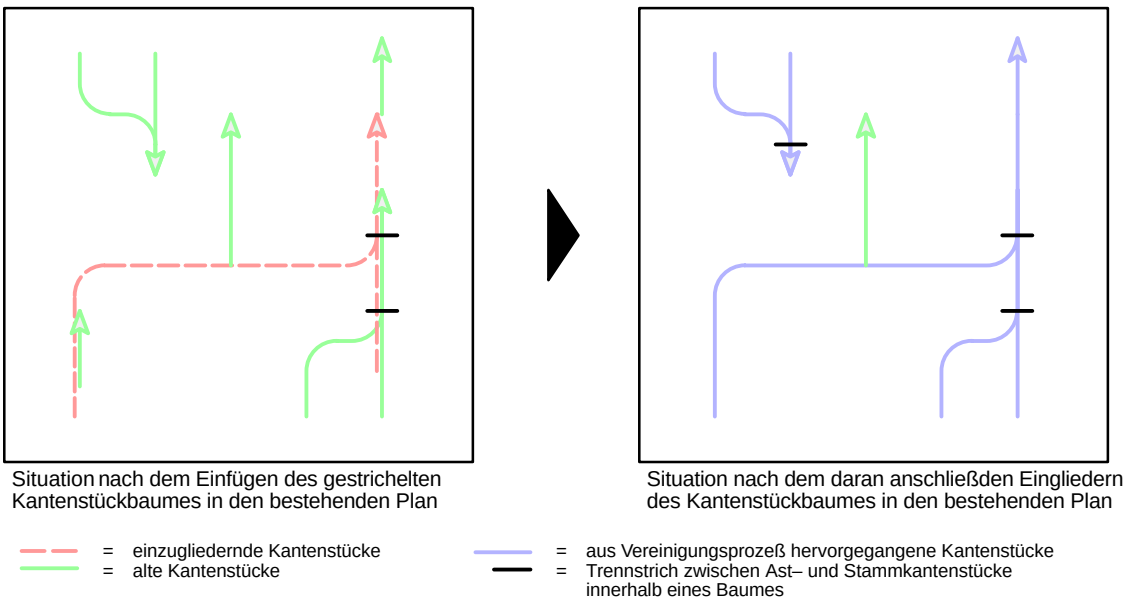
Der Ansatz, Referenzen auf den Stamm, sowie auf die Äste als Attribute eines Kantenstückes zu realisieren, hat den Vorteil, daß ganze (Teil–)Bäume aus einem Plan ausgeschnitten oder herauskopiert und an anderer Stelle wieder eingefügt werden können, ohne daß hierdurch die kopierte Baumstruktur beim Wiedereinfügen verloren gehen kann. Bei einer zentralen Verwaltung dieser Information für einen Plan, beispielsweise durch den Planakteur, könnte diese Information nicht planübergreifend erhalten werden. Die Kantenstücke müßten nacheinander als unabhängige Kantenstücke in die bestehende Kantenbaumstruktur eingepaßt werden. Werden diese unglücklicherweise über ursprünglichen Kantenstücken des neuen Planes positioniert, könnte hierdurch die Struktur des einzufügenden Baumes durch eine Neuzuordnung zu unterschiedlichen ursprünglichen Bäumen zerstört werden.

Allerdings müssen auch bei diesem Ansatz die Referenzen beim Einfügen erneut angepaßt werden, da es sonst durch das mehrfache Auftreten desselben Schlüssels für unterschiedliche Objekte zu Problemen kommen kann. Diese werden in Kapitel 1.4 erläutert.

1.3.3 Eingliedern von Kantenstücken

1.3.3.1 Problemstellung

Nach jedem Erzeugen eines Kantenstückes (vgl. Bild–A11), sowie allen anderen Aktionen, bei denen Kantenstücke an neuen Stellen im Plan auftauchen, muß anschließend dafür gesorgt werden, daß alle neu entstandenen Kantenbaumstrukturen in Minimalform überführt werden. Bild–17 gibt eine Vorstellung von der Aufgabe. Ausgehend von den ursprünglichen Kantenbaumstrukturen und den neu hinzugekommenen Kantenstückbäumen müssen zuerst mögliche neue Kantenbäume erkannt werden und dann anschließend die daran beteiligten Kantenstücke entsprechend angepaßt werden. Linienzüge müssen verlängert oder gekürzt, Referenzen gesetzt und teilweise komplette Kantenstücke erzeugt oder gelöscht werden. Allgemein formuliert besteht das zu lösende Problem darin, eine beliebige Anzahl beliebig gestalteter Kantenstückbäume in Minimalform in einen bestehenden Plan, der seinerseits bereits eine beliebige Anzahl beliebig gestalteter Kantenstückbäume in Minimalform enthält, so einzugliedern, daß der Plan anschließend wiederum nur Kantenstückbäume in Minimalform enthält, wobei die ursprünglichen Strukturen der Kantenbäume komplexer geworden, nicht aber zerstört worden sein dürfen. Das heißt, man findet nach dem Eingliedern alle Bäume wieder, sei es unverändert oder innerhalb neuer Bäume als Teil dieser Bäume.



Bild–17: Eingliederungsbeispiel

In Bild–17 entstehen so zwei neue Minimalformkantenbäume. Der kleinere links oben im Bild gezeigte Kantenbaum entsteht durch das Vereinigen zweier zuvor isolierter Kantenstücke zu ei-

nem Stammkantenstück und zwei Ästen. Der zweite neue Kantenstückbaum entsteht dadurch, daß durch übereinanderlegen eines neuen Kantenbaumes über zwei anfangs isolierte Kantenstücke sowie einen weiteren ursprünglichen Kantenstückbaum eine einzige größere Kantenbaumstruktur beschrieben wird, die ebenfalls in Minimalform umgewandelt worden ist.

Sinnlose Anordnungen von Kantenstücken können hierdurch natürlich nicht besser werden, umgekehrt muß die gesuchte Lösung so allgemeingültig sein, daß sie auch bei den widersinnigsten Vorgaben der Aufgabenstellung entsprechend nur Bäume in Minimalform liefert.

1.3.3.2 Grobablauf des Eingliederns

Einen groben Überblick über den Ablauf des Eingliederungsprozesses gibt Bild–A12. Er läßt sich im wesentlichen in 4 Abschnitte einteilen, die nachfolgend kurz beschrieben werden und in den folgenden Unterkapiteln detailliert diskutiert werden.

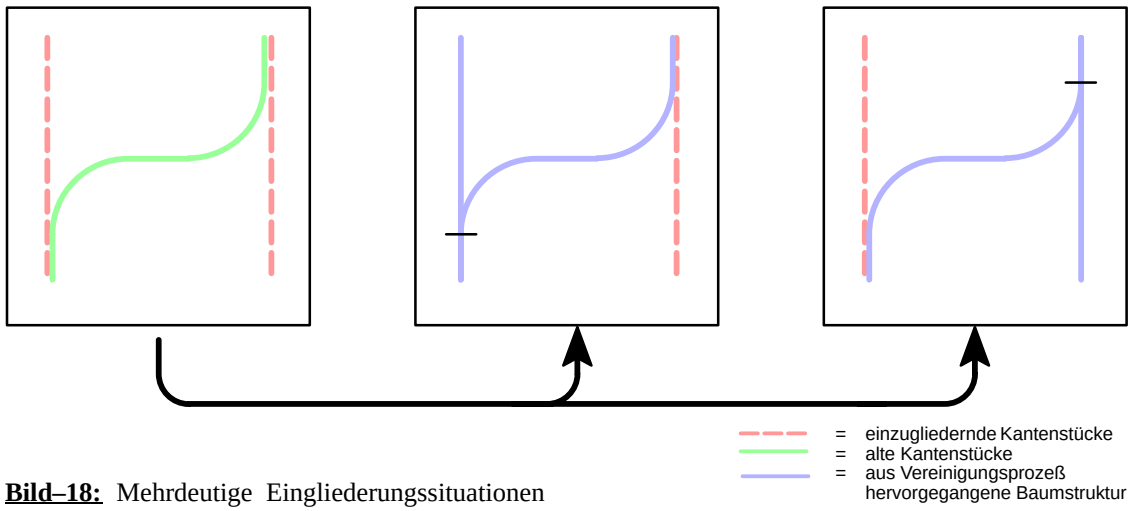
- Bestimmen der Struktur neu einzufügender Kantenstückbäume
- Ermitteln der Kontaktstellen von Baumwurzeln
- Prüfen, ob die Vereinigung der am Kontakt beteiligten Kantenstück(bäum)e zulässig ist
- Vereinigung der an zulässigen Kontakten beteiligten Kantenstück(bäum)e

Ausgangspunkt ist die Menge aller neu in den Plan eingefügten Komponenten. Hierunter sind fallweise neu erzeugte Kantenstücke, aus anderen Plänen kopierte, aber auch bewegte Komponenten eines Planes zu verstehen, die quasi an einer Stelle aus dem Plan gelöscht und an anderer Stelle wiedereingefügt werden. Für die weitere Betrachtung ist nur die Teilmenge der Kantenstücke von Bedeutung. Für alle anderen Komponenten ist nach dem Einfügen der Editionsschritt beendet. Die Menge der Kantenstücke läßt sich partitionieren in Gruppen von Kantenstücken, die jeweils einen vollständigen Kantenstückbaum bilden.

Um die Mitglieder eines Baumes zu bestimmen, reicht es aus, ein beliebiges Kantenstück eines Baumes zu nehmen und sukzessive nach dem Stamm zu fragen, bis man ein Kantenstück erhält, daß selbst keinen Stamm besitzt. Dieses Kantenstück muß die Wurzel des Baumes sein. Von dieser Wurzel ausgehend notiert man dann, die Wurzel inklusive, rekursiv alle Äste des Baumes bis man an den Blättern angelangt ist.

Nacheinander werden diese Gruppen von Kantenstücken, das heißt Kantenstückbaum für Kantenstückbaum in den bestehenden Plan eingegliedert. Das heißt, der erste Kantenstückbaum wird in den ursprünglichen Plan eingegliedert, wodurch ein neuer Zwischenplan entsteht. In diesen wird der nächste Kantenstückbaum eingliedert usw., bis alle Bäume eingefügt sind. Die dabei gewählte Reihenfolge ist zufällig. Bei unglücklicher Positionierung ursprünglicher und neu einzufügender Bäume zueinander kann dies jedoch zu unterschiedlichen Ergebnissen führen. In Bild–18 ist ein solcher Sonderfall gezeigt, in dem das Hinzukommen zweier neuer Kantenstücke

zusammen mit einem alten Kantenstück zu einer mehrdeutigen Situation führt. Entweder ergibt sich der rechte Fall, in dem das rechte Kantenstück mit dem alten einen Baum bildet und das linke isoliert bleibt oder aber das linke Kantenstück wird mit dem alten zu einem Baum vereinigt und das rechte bleibt isoliert.

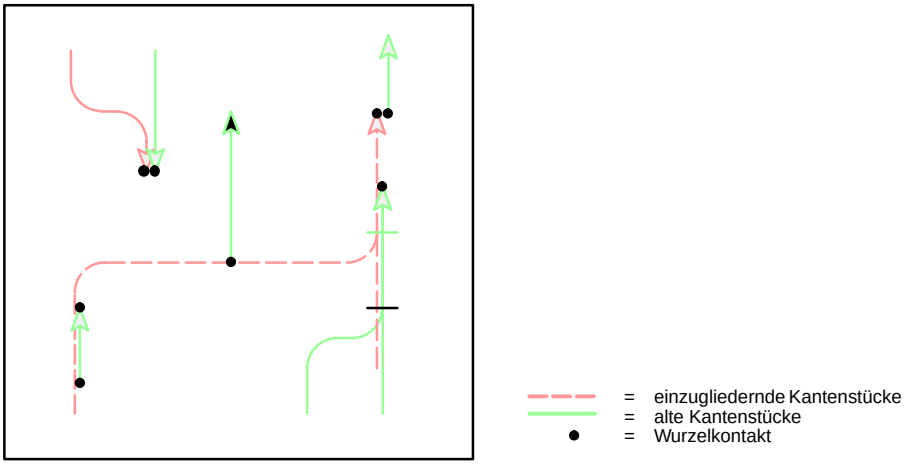


Bild–18: Mehrdeutige Eingliederungssituationen

Da dies jedoch seltene Sonderfälle sind, in denen es keine eindeutige Lösung gibt, darf das Ergebnis in diesen Fällen vom Zufall abhängig sein, zumal die Lösung keineswegs der Aufgabenstellung widerspricht.

Beim Eingliedern eines einzelnen Kantenbaumes in einen bestehenden Plan wird aus einer Menge neuer Kantenstücke (*"Neu–Menge"*), die alle einem einzigen Baum angehören und einer Menge alter Kantenstücke (*"Alt–Menge"*), die alle bereits eingegliederten Bäume umfaßt, eine neue Menge von Kantenstücken (*"Nachher–Menge"*). Diese Menge umfaßt alle Kantenstücke der Bäume, die aus dem Zusammenführen der beiden Ausgangsmengen entstanden sind.

Um einen Kantenstückbaum in einen bestehenden Plan einzugliedern, müssen zunächst alle *Wurzelkontakte* zwischen den Kantenstücken der Neu–Menge und denen der Alt–Menge bestimmt werden. Der Begriff Wurzelkontakt bezeichnet hierbei eine grafische Berührungsstelle zwischen einem *Wurzelpunkt* eines Baumes und einem beliebigen Punkt auf dem Linienzug eines anderen Kantenstückes. Der Wurzelpunkt eines Baumes ist der Endpunkt des Wurzelkantenstückes eines Baumes, in dem der Baum beginnt. Besteht ein Kantenstückbaum nur aus einem Kantenstück, sind dessen beide Endpunkte Wurzelpunkte, da sich je nach Kontaktsituation zu einem anderen Kantenstück in beide Richtungen ein Baum ergeben kann. Enden zwei Kantenstücke in demselben Punkt und sind beide Endpunkte Wurzelpunkte oder aber überlappen sich zwei Kantenstücke, die ebenfalls beide mit einem Wurzelpunkt aufeinander enden, so werden hierdurch zwei *duale* Wurzelkontakte beschrieben. Es genügt in diesen Fällen nur einen Kontakt für die Vereinigung beider Bäume zu berücksichtigen. Bild–19 zeigt alle Wurzelkontakte zwischen neuen und alten Bäumen des einleitenden Beispiel.
Nur an diesen Wurzelkontakten soll es erlaubt sein, daß Kantenstücke beider Mengen zu einem



Bild–19: Wurzelkontakte

Baum zusammenwachsen. Können keine Wurzelkontakte gefunden werden, gibt es keine Möglichkeit Kantenbäume beider Mengen zu vereinigen, ohne Baumstrukturen zu zerstören. Dies liegt im Wesen von Baumstrukturen begründet. In diesen Fällen ist der Eingliederungsprozeß des durch die Neu–Menge beschriebenen Baumes abgeschlossen.

Können dagegen Wurzelkontakte gefunden werden, werden die Beschreibungen all dieser Kontakte in einer Liste gesammelt[18.]. Anfänglich besteht eine solche Beschreibung aus einem Verweis auf das Wurzelkantenstück und das berührte Kantenstück, sowie den Koordinaten des Kontaktes. Es ist wichtig, alle Wurzelkontakte am Anfang des Vereinigungsprozesses zu bestimmen. Nur am Anfang sind alle betrachteten Kantenstücke eindeutig entweder der Neu– oder der Alt–Menge zugeordnet. Im sich anschließenden Eingliederungsprozeß geht diese Zuordnung dadurch verloren, daß Kantenstücke aus beiden Mengen verändert, gelöscht und durch neue Kantenstücke ersetzt werden. Diese Kantenstücke sind keiner der beiden Ausgangsmengen zugeordnet, sondern bilden eine dritte sogenannte Übergangsmenge. Weitere Wurzelkontakte müßte man dann zwischen Kantenstücken aller drei Mengen – der Neu–, der Altmenge und der Übergangsmenge – sowie innerhalb der Übergangsmenge suchen. Da sich alle drei Mengen bei jeder Vereinigung von Bäumen ändern können, müßte die Suche jedesmal von Grund auf neu begonnen werden. Daher wäre es sinnvoll auch bei diesem Verfahren eine Liste von Wurzelkontakten anzulegen. In dieser müßten all die Kontakte vermerkt werden, die schon erkannt wurden, aber wegen Unverträglichkeit nicht zu einer Vereinigung geführt haben. Stößt man bei der wiederholten Suche nach neuen Kontakten auf alte bereits als unbrauchbar befundene Kontakte, kann man diese anhand der Negativliste identifizieren und ohne aufwendige Prüfung verwerfen. Werden beim Eingliederungsprozeß Kantenstücke verändert, zerstört oder neu erzeugt, müssen die Kontaktbeschreibungen dieser Liste überarbeitet werden.

18.. Die Beschreibung eines Wurzelkontaktes wird in der Implementierung jeweils in einem eigenen Datenobjekt abgelegt. Hierfür wurde eine eigene Klasse, die *spk–root–contact–class* definiert, um so den Aufbau der Datenstruktur nach außen zu verbergen. Die erwähnte Liste enthält dadurch lediglich Verweise auf die jeweiligen Datenobjekte.

Aufgrund des unüberschaubaren und ineffektiven Suchprozesses des zweiten Verfahrens wurde die Entscheidung getroffen, alle Kontakte am Anfang des Eingliederungsprozesses zu bestimmen. Auch bei diesem Verfahren müssen allerdings die Kontaktbeschreibungen gepflegt werden.

Ist die anfängliche Kontaktliste erstellt, beginnt man die Einträge dieser Liste nacheinander auszuwerten. Zuerst wird getestet, ob der Kontakt eine verträgliche Berührung zweier Kantenstückbäume darstellt. Dazu muß eine Reihe von Bedingungen erfüllt sein, die zunächst kurz angesprochen werden sollen.

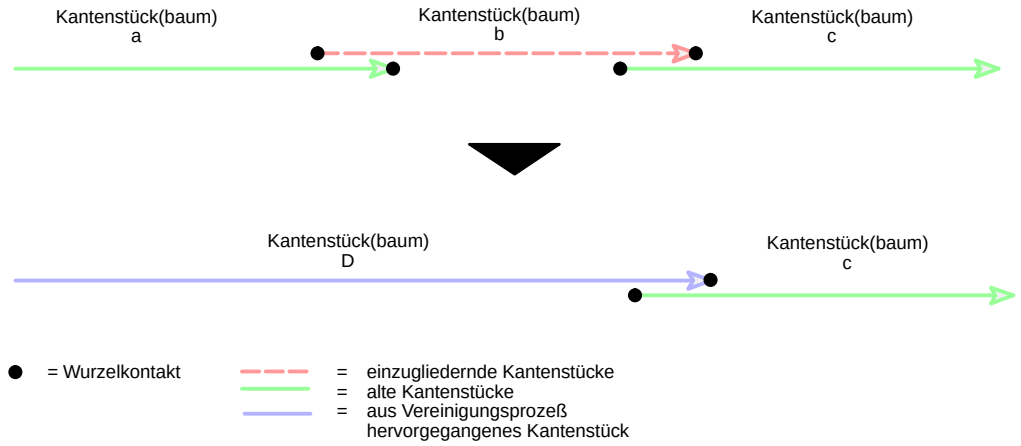
Zum einen müssen die Attribute der Kantenstücke miteinander verträglich sein. Dies muß deshalb so sein, weil die Attribute einer Kante den Attributen aller Kantenstücke entsprechen, aus denen die Kante zusammengesetzt ist. Insofern müssen diese Attribute widerspruchsfrei sein. Beispielsweise kann es nicht sein, daß im Kantenstückbaum eines Petrinetzes eine Kante abschnittsweise unterschiedliche Flußzahlen besitzt. Ebenso ist es natürlich nicht zulässig, diese Widerspruchsfreiheit dadurch zu erzielen, daß willkürlich Attribute von Kantenstücken angepaßt werden.
Die andere Gruppe von Verträglichkeitsbedingungen beschreibt die grafischen Beziehungen, die zwischen zwei Kantenstückbäumen bestehen müssen, damit die Vereinigung beider Bäume wieder einen Baum ergibt. Zum Beispiel muß immer eindeutig eine Wurzel definiert sein.

Ist ein Kontakt für zulässig befunden worden, können die beiden betroffenen Kantenstückbäume vereint werden. Von der gemeinsamen Wurzel ausgehend werden dabei Kantenstück für Kantenstück der beiden Bäume verglichen und durch gemeinsame Stücke ersetzt, bis man auf Äste stößt, an denen die Bäume sich trennen. Die daraus verzweigenden Äste bleiben unverändert. Das genaue Verfahren wird detailliert im Anschluß beschrieben.

Bevor der nächste Kontakt betrachtet werden kann, muß die Kontaktliste den Konsequenzen des letzten vorangegangenen Vereinigungsschrittes angepaßt werden.

In Bild–20 wird dieser Zusammenhang an einem Beispiel erläutert. Die Alt–Menge besteht aus den beiden Kantenstücken a und c. Das neue einzugliedernde Kantenstück sei b. In dieser Situation existieren insgesamt 4 Wurzelkontakte, von denen jeweils 2 als dual zueinander anzusehen sind:

– Wurzelkontakt zwischen Wurzelkantenstück a und berührtem Kantenstück b
– Wurzelkontakt zwischen Wurzelkantenstück b und berührtem Kantenstück a
– Wurzelkontakt zwischen Wurzelkantenstück b und berührtem Kantenstück c
– Wurzelkontakt zwischen Wurzelkantenstück c und berührtem Kantenstück b



Bild–20: Wegfall von Wurzelkontakten durch Vereinigung

Beim Abarbeiten der Kontaktliste wird der erste Kontakt für zulässig befunden und die Kantenstücke a und b durch das neue Kantenstück D ersetzt. Damit ist die erste Kontaktbeschreibung berücksichtigt und kann aus der Kontaktliste entfernt werden. Die verbleibenden Kontaktbeschreibungen müssen zunächst an die neue Situation angepaßt werden:

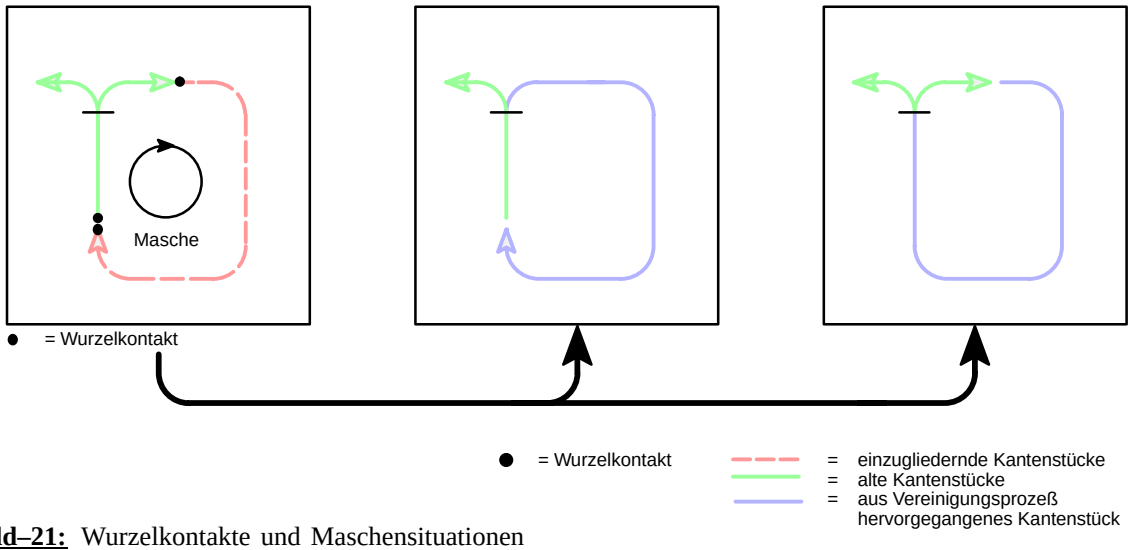
– Wurzelkontakt zwischen Wurzelkantenstück D und berührtem Kantenstück D
– Wurzelkontakt zwischen Wurzelkantenstück D und berührtem Kantenstück c
– Wurzelkontakt zwischen Wurzelkantenstück c und berührtem Kantenstück D

Danach wird die Liste untersucht, ob sie Kontaktbeschreibungen zwischen Kantenstücken enthält, die demselben neu entstandenen Baum angehören. Diese Beschreibungen müssen aus der Kontaktliste entfernt werden. Auf diesem Weg löst man zwei Probleme auf einmal:

- Die Beschreibung eines eventuellen, hinfällig gewordenen dualen Kontaktes zu dem eben bearbeiteten Kontakt wird entfernt. Im Beispiel wäre dies der ehemals zweite Wurzelkontakt, der nun einen recht unsinnigen Kontakt eines Kantenstückes D zu sich selbst beschreibt.

- Bei unglücklicher Positionierung von Kantenstückbäumen zueinander können Schleifen oder Maschen beschrieben werden. Werden alle Wurzelkontakte am Anfang des Vereinigungsprozesses bestimmt, ist für solche Situationen kennzeichnend, daß man immer zwei Wurzelkontakte zwischen diesen beiden Kantenstückbäumen bestimmen kann. Jeder dieser Kontakte für sich genommen, beschreibt einen verträglichen Kontakt zwischen denselben zwei Bäumen. Eine Vereinigung der beiden beteiligten Bäume in beiden Kontaktpunkten würde zu einer Maschenbildung führen. Das Löschen der zweiten Kontaktbeschreibung nach dem Auswerten des ersten Kontaktes verhindert dies und bewirkt somit, daß die Baumstruktur nicht zerstört wird. Abhängig von welchem Kontakt ausgehend die beiden Bäume vereinigt werden, entstehen unterschiedliche Ergebnisse. Auch hier ist es zulässig, daß die

jeweilige Entscheidung vom Zufall abhängig getroffen wird. Bild–21 zeigt ein mögliches Beispiel der Problematik. Das neu eingefügte Kantenstück hat an beiden Endpunkten einen Wurzelkontakt zu einen Baum des ursprünglichen Planes. Es ist zulässig entweder an dem einen oder an dem anderen Ende eine Vereinigung durchzuführen. Je nachdem ergibt sich eine Verlängerung des Wurzelkantenstückes (rechtes Bild) oder des betroffenen Astkantenstückes (rechtes Bild). Eine Vereinigung in beiden Kontaktpunktes ist nicht zulässig, da es ansonsten zu einer unzulässigen Maschenstruktur kommen würde.



Bild–21: Wurzelkontakte und Maschensituationen

Enthält die Kontaktliste nach dem Anpassungsvorgang noch Beschreibungen unbehandelter Kontakte, wiederholt sich der Ablauf ab dem Testen der Kontaktverträglichkeit. Dies geschieht solange, bis alle Kontaktbeschreibungen abgearbeitet sind (siehe hell unterlegte Schleife in Bild–A12). Danach ist ein Kantenstückbaum vollständig in den bisherigen Plan eingegliedert. Das Verfahren muß gegebenenfalls für weitere Kantenstückbäume wiederholt werden.

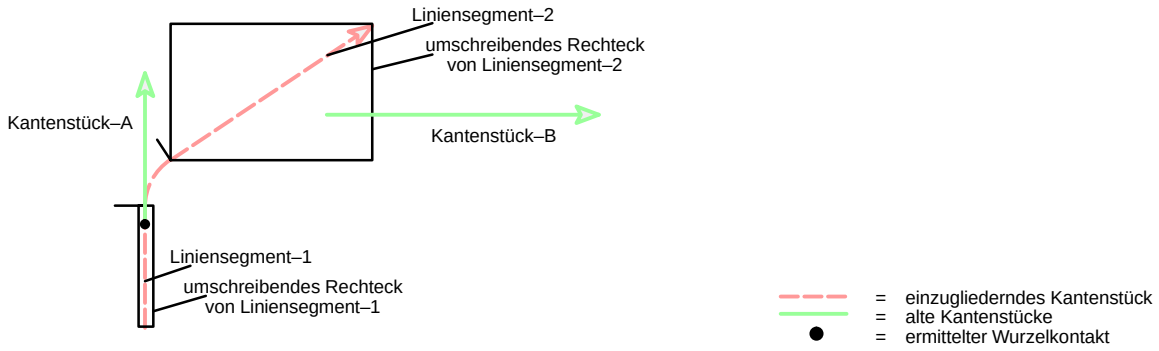
Im folgenden soll detailliert auf die einzelnen Schritte eingegangen werden:

1.3.3.3 Bestimmen der Wurzelkontakte

Zu Beginn jedes Eingliederungsprozesses werden alle Wurzelkontakte des einzugliedernden Baumes mit dem ursprünglichen Plan bestimmt (vgl. Neu–/Alt–Menge). Dabei wird nach dem in Bild–A13 gezeigten Verfahren vorgegangen. Dieses Petrinetz stellt eine Verfeinerung der Transition ”Bestimme alle diesen Baum betreffenden Wurzelkontakte” aus Bild–A12 dar. Es geht zunächst darum, die Anzahl der in Frage kommenden Kontaktpartner möglichst schnell von allen im ursprünglichen Plan vorkommenden Kantenstücken auf eine möglichst kleine Gruppe der engeren Wahl zu reduzieren. In Frage kommen sicher nur Kantenstücke, die den Linienzug irgendeines Kantenstückes der Neu–Menge berühren. Darüber hinaus wurde im Anforderungsprofil (vgl. [8]) zum Kantenstückkonzept gefordert, daß ein zulässiger Wurzelkontakt auf der

Geraden zwischen zwei Eckpunkten eines Linienzuges liegen muß. Daraus folgt unmittelbar, daß nur solche Wurzelkontakte in Frage kommen, die auf Liniensegmenten liegen, nicht aber solche, die auf verrundenden Bogensegmenten liegen.

Der einzige Weg um festzustellen, ob sich bestimmte Grafikobjekte an einer bestimmten Position des Planes befinden oder nicht, besteht bei Interleaf–V darin, die Grafikelemente einer Gruppe bzw. des ganzen Planes zu bestimmen, die ganz oder teilweise in einem bestimmten rechteckigen Bildschirmausschnitt liegen[19.]. In Sonderfällen kann dieser Ausschnitt zu einem Punkt entarten. Um dies ausnutzen, bestimmt man zunächst das *umschreibende Rechteck* eines Liniensegmentes. Hierbei handelt es sich um das kleinste Rechteck, das in horizontaler und vertikaler Richtung aufgespannt genau das betrachtete Grafikobjekt, in diesem Fall ein Liniensegment, umfaßt. Die Position und Ausdehnung des umschreibenden Rechteckes ist für jedes Grafikobjekt in Interleaf–V erfragbar. Danach erfragt man die Grafikobjekte des Planes, die sich ganz oder teilweise in dem umschreibenden Rechteck befinden. Befinden sich unter diesen Objekten Kantenstücke, die vom gleichen Typ wie das untersuchte Kantenstück sind, werden diese zur weiteren Untersuchung in einer Liste notiert. Bei Liniensegmenten, die nicht horizontal oder vertikal verlaufen, beschreibt die Linie im Rechteck eine Diagonale. Hierdurch werden auch solche Kantenstücke ermittelt, die zwar im Rechteck liegen, nicht aber ein Liniensegment des untersuchten Kantenstückes berühren. Bild–22 zeigt ein Beispiel.



Bild–22: Ermittlung der Wurzelkontakte

Der Linienzug des einzugliedernden Kantenstückes enthält zwei Liniensegmente. In dem umschreibenden Rechteck des Liniensegmentes 1 liegt teilweise das Kantenstück A, in dem umschreibenden Rechteck des Liniensegmentes 2 liegt teilweise das das Kantenstück B. Obwohl nur Kantenstück A wirklich das einzugliedernde Kantenstück berührt, werden zunächst beide Kantenstücke als mögliche Kandidaten notiert. Eventuelle weitere Kantenstücke des Planes, die nicht innerhalb der umschreibenden Rechtecke liegen werden durch diesen Test ausgefiltert.

Diese Prozedur, die durch die obere Schleife im Petrinetz Bild–A13 beschreiben wird, erlaubt es, auf schnellem Wege den Kreis der Kandidaten erheblich einzugrenzen. Da sich aber die aneinander grenzenden Kantenstücke eines Baumes ebenfalls gegenseitig berühren, können zunächst noch irrtümlicherweise Kantenstücke des einzugliedernden Baumes in der Kandidatenliste ver-

19.. siehe Interleaf–Online–Dokumentation zu *dg–selpt* (vgl. auch [8])

merkt sein. Diese, sowie Mehrfacheinträge durch mehrfach berührte Kantenstücke, müssen aus der Liste entfernt werden.

Überall dort, wo der Wurzelpunkt des einzufügenden Baumes (ggf. zwei Wurzelpunkte, wenn es ein einzelnes Kantenstück ist) einen Kandidaten berührt, besteht ein Wurzelkontakt. Umgekehrt können alle Kandidaten, bei denen es sich um Wurzelkantenstücke handelt, ebenfalls in ihrem Wurzelpunkt Äste des einzufügenden Baumes berühren. Um diese Kontakte zu ermitteln, muß man für jedes Wurzelkantenstück der einen Menge die Linienzüge aller Kantenstücke der anderen Menge auf Berührung im Wurzelpunkt testen. Die Beschreibungen aller so ermittelten Kontakte werden wie zuvor beschrieben in der Kontaktliste vermerkt. Die Menge aller näher zu überprüfenden Situationen wird damit auf die ermittelten Wurzelkontakte beschränkt. Diese Kontakte können nun nacheinander näher untersucht werden. In Bild–22 bleibt so nur der zu überprüfende Wurzelkontakt des Kantenstückes A zum einzugliedernden Kantenstück übrig. Der Kandidat B fällt weg, da keiner seiner Endpunkte auf dem einzufügenden Kantenstück liegt.

1.3.3.4 Verträglichkeitsprüfung

Wie zuvor beschrieben, wird in diesem Schritt die Verträglichkeit der in der die Kontaktbeschreibung spezifizierten sich berührenden Kantenstückbäume ermittelt.

Dazu wird zuerst die *Verträglichkeit der (unsichtbaren) Attribute* des Wurzelkantenstückes und des berührten Kantenstückes überprüft. Je nach Klassenzugehörigkeit der Kantenstücke können unterschiedliche Attribute und Verträglichkeitsbedingungen definiert sein. Das einzige für alle Kantenstücke gleichermaßen definierte Attribut ist der Verrundungsparameter. Dieser muß für beide sich berührenden Kantenstücke gleich sein. Ansonsten kann durch Vereinigen zweier Kantenstücke ein Kantenstücklinienzug mit unterschiedlichem Radius entstehen. Für dieses könnte der Verrundungsparameter nicht bestimmt werden. Umgekehrt gibt es Attribute, die grundsätzlich miteinander verträglich sind. Dazu gehört zum Beispiel der Bedingungstext von Petrinetzkantenstücken. In solchen Fällen bildet die Vereinigung beider Attribute nach einem eventuellen Vereinigen der beiden Kantenstücke das entsprechende Attribut für das aus dem gemeinsamen Teil entstandene Kantenstück. Im Falle der Petrinetzkantenstücke wäre dies die Disjunktion der Bedingungstexte. Da es sich bei den Attributverträglichkeitstest in der Regel um einfache Vergleiche und daher um sehr schnell zu überprüfende Bedingungen handelt, ist es sinnvoll diese an den Anfang der Kontaktüberprüfung zu setzen.

Ist die Attributverträglichkeit gegeben, wird überprüft, ob ein Vereinigen beider Bäume wieder zu einer Baumstruktur führen würde. Dazu ist eine Reihe von unterschiedlichen, zum Teil aufeinander aufbauenden Bedingungen zu erfüllen. Einige der hierbei ermittelten Informationen dienen gleichzeitig einer Erweiterung der Kontaktbeschreibung, die bei positivem Testergebnis als Ausgangspunkt für den Vereinigungsprozeß verwendet wird. Zu Beginn einer Überprüfung besteht eine Kontaktbeschreibung aus:

- einem Verweis auf das Wurzelkantenstück
- der Angabe, ob es sich bei dem Wurzelpunkt um Start– oder Endpunkt des Wurzelkantenstückes handelt
- der Position des Wurzelpunktes
- einem Verweis auf das berührte Kantenstück

Die erste Bedingung besagt, daß *die Zeichenrichtung der Linienzüge der beiden am Kontakt beteiligten Kantenstücke im Kontaktpunkt bei ungerichteten Kanten parallel, bei gerichteten Kanten gleich sein muß.*

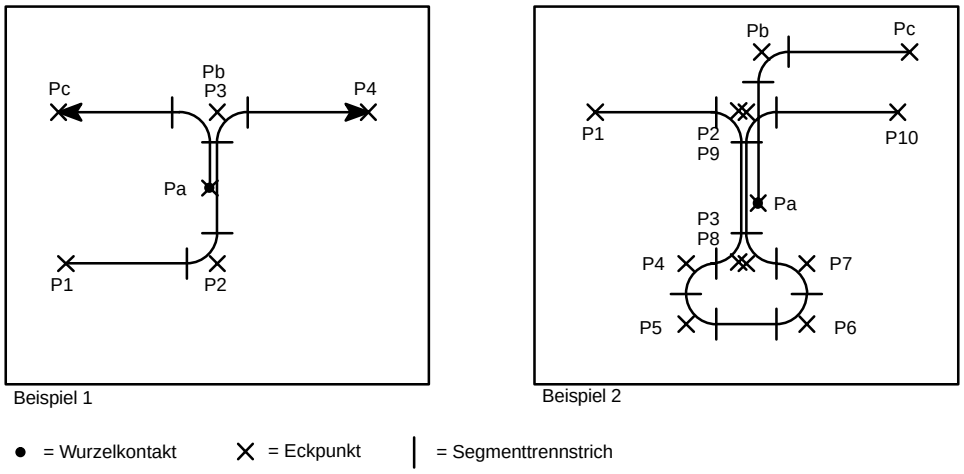
Die Zeichenrichtung ist in jedem Punkt eines Linienzuges als die Richtung vom Start– zum Endpunkt des Kantenstückes definiert. Die Festlegung, welcher der beiden Randpunkte Start– und welcher Endpunkt ist, hängt zunächst von der Erzeugungsreihenfolge der Punkte ab und kann u.U. durch weitere Editionsschritte (z.B. Richtungsumkehr) verändert werden.

Da bei SPIKES–Editoren per Definition die Pfeilrichtung gerichteter Kanten der Zeichenrichtung entspricht, es ist einsichtig, daß die Zeichenrichtung zweier zu vereinigender Kantenstücke übereinstimmen muß, da sich ansonsten die Pfeilrichtung widersprechen würde, was logisch betrachtet Unsinn ist.

Die Zeichenrichtung im Kontaktpunkt des zum Wurzelkantenstück gehörenden Liniensegmentes kann über den Vektor vom Wurzelpunkt zum darauffolgenden Eckpunkt berechnet werden. Ist der Wurzelpunkt der Endpunkt des Linienzug, dann ist die Richtung entgegengesetzt definiert.

Die Richtung des berührten Liniensegmentes zu finden, ist aufwendiger. Bei einfachen Linienzügen könnte man die Eckpunktfolge des berührten Kantenstückes auswerten. Die Bedingung ist dann erfüllt, wenn sich der Wurzelpunkt auf der Geraden zwischen zwei Eckpunkten befindet und die durch diese Eckpunkte definierte Zeichenrichtung der Zeichenrichtung des Wurzelkantenstückes im Kontaktpunkt entspricht. So liegt in Bild–23, Beispiel 1, der Startpunkt Pa des berührenden Kantenstückes zwischen den beiden Eckpunkten P2 und P3 des berührten Kantenstükkes.

Es gibt jedoch Gründe, die gegen dieses Verfahren sprechen. Existieren viele Kontaktpunkte oder handelt es sich um komplexe Linienzüge mit vielen Eckpunkten, wird die Anzahl der durchzu-



Bild–23: Ein– und mehrdeutige Kontaktsituationen

führenden Rechenoperationen sehr groß, so daß die Performance darunter leiden würde[20.]. Bei unglücklichen Linienzügen, ist darüber hinaus keine eindeutige Zuordnung zwischen zwei aufeinander folgenden Eckpunkten und einem Liniensegment möglich. Bild–23, Beispiel 2, zeigt eine solche unglückliche Situation. Ohne eine zusätzliche Information, beispielsweise über die Reihenfolge der Liniensegmente, ist es hier nicht möglich, zu bestimmen, welches der beiden positionsgleichen Punktpaare (P2,P3) bzw. (P8,P9), welches der beiden abgeleiteten dazwischen liegenden Liniensegmente spezifiziert und somit welches der beiden berührten Liniensegmente für den Kontakt relevant ist. Diese Information wird aber, wie man im folgenden noch sehen wird, für die Vereinigung zweier Kantenstücke benötigt.

Einfacher ist es deshalb, nacheinander alle Liniensegmente des berührten Kantenstückes zu überprüfen, ob sie von dem Wurzelkontakt berührt werden. Trifft dies für ein Liniensegment zu, erfragt man einfach das *Richtungsattribut* dieses Linienstückes, das beim Erzeugen entsprechend der Zeichenrichtung gesetzt wird[21.]. Natürlich ist es sinnvoller, auch für das Liniensegment des Wurzelkantenstückes die Zeichenrichtung auf diesem Wege zu bestimmen.

Für weitere Betrachtungen, sowie den späteren Vereinigungsprozeß vermerkt man als Ergebnis der Richtungsüberprüfung in der Kontaktbeschreibung

- einen Verweis auf das Liniensegment des Wurzelkantenstückes
- einen Verweis auf das Liniensegment des berührten Kantenstückes

20.. Die nötigen Prozeduren müssen in Interleaf mittels LISP implementiert werden. Größere Rechenoperationen wirken sich im Vergleich mit üblichen Programmumsetzungen unverhältnismäßig zeitintensiv aus, da in LISP vor dem Abwickeln der Programme keine Übersetzung der Programmanweisungen in maschinennahe Prozessorbefehle stattfindet, sondern mehr oder weniger direkt die LISP–Anweisungen durch einen speziellen LISP–Interpreter abzuwickeln sind.

21.. Hierzu wurde eigens eine Unterklasse für Linien *(spk–dir–dg–line–class)* definiert, die dieses Richtungsattribut, sowie ein Positionsattribut zur Bestimmung der Reihenfolge der Segmente innerhalb eines Linienzuges, verwaltet.

Die zweite Verträglichkeitsbedingung *besagt, daß ein gemeinsamer Baum nur eine einzige Wurzel besitzen kann.*

Zur Betrachtung dieser Bedingung ist es erforderlich, den Begriff der Wachstumsrichtung einzuführen. Bei echten Kantenstückbäumen bestehend aus einer Wurzel und mehreren Ästen (mehrästiger Kantenstückbaum) ist diese definiert, durch den Weg von der Wurzel des Baumes zu seinen Blättern und gibt an, in welche Richtung der Baum wachsen, d.h. erweitert werden kann. Bei isolierten Kantenstücken (entarteter Kantenstückbaum aus einem einzigen Kantenstück) sind beide Enden Wurzelpunkte und potentielle Anknüpfpunkte für Kantenbäume. Sie bestimmt sich hier aus dem Weg ausgehend vom betrachteten Wurzelkontaktpunktes hin zum jeweils anderen Randpunkt. Dadurch wird die Richtung ausgewählt, in die das Kantenstück (zuerst) erweitert werden soll. Für berührte isolierte Kantenstücke ist keine Wachstumsrichtung definiert.

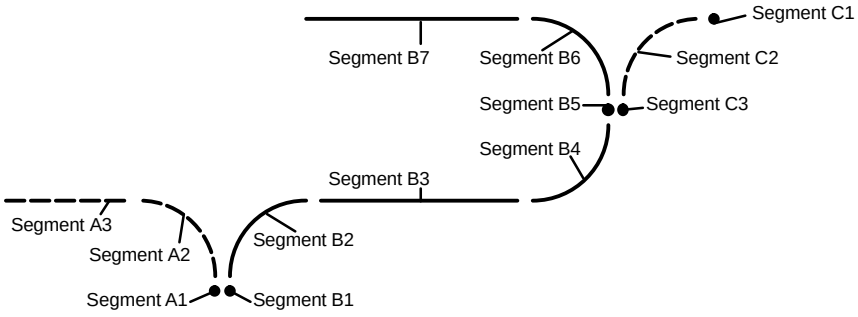
Unter der Voraussetzung, daß die erste Bedingung erfüllt ist, kann entsprechend Bild–A14 jede erdenkliche Kontaktsituation einer von insgesamt sechs Gruppen zugeordnet werden. Betrachtet man Wurzelkontakte, bei denen der berührte Kantenstückbaum nur aus einem einzigen Kantenstück besteht, lassen alle denkbaren Situationen eine Vereinigung zu einem Baum zu. Besteht der berührte Kantenstückbaum hingegen aus mehreren Kantenstücken kann es Probleme geben. Dies ist dann der Fall, wenn die Wachstumsrichtung der beiden am Kontakt beteiligten Bäume im Kontaktpunkt entgegengesetzt ist. Diese Situation ergibt sich einerseits bei zwei voneinander weggerichteten mehrästigen Kantenstückbäumen, aber auch dann, wenn das Wurzelkantenstück dem berührten Kantenstückbaum in Richtung Wurzel folgt. Beim ersten Fall ist eine Verträglichkeit beider Bäume grundsätzlich ausgeschlossen, da bei einer Vereinigung keine Wurzel mehr definiert ist. Beim zweiten Fall ist eine Vereinigung möglich, wenn das Wurzelkantenstück dem berührten Kantenstückbaum bis zu dessen Wurzelpunkt folgt oder auf dem Weg dorthin endet. Wie man in Bild–A14 erkennen kann, gibt es in diesen Situationen immer einen dualen Wurzelkontakt, so daß durch Rollentausch von berührtem und berührendem Kantenstück diese Fälle auf grundsätzlich sichere Situationen zurückgeführt werden können. Daher kann man den zweifelsbehafteten Kontakt einfach verwerfen und statt dessen den dualen Kontakt berücksichtigen. Auf diesem Weg umgeht man die aufwendige Rückverfolgung der Linienzüge entgegen der Wachstumsrichtung des berührten Baumes, möglicherweise über mehrere Kantenstücke hinweg, nur um festzustellen, ob es sich um eine zulässige Situation handelt.

Um die zweite Bedingung zu überprüfen, geht man daher wie folgt vor. Hat das berührte Kantenstück weder Äste noch einen Stamm, ist die Situation unkritisch, die Bedingung ist erfüllt. Ansonsten ermittelt man aus dem Richtungsattribut des berührten Liniensegmentes und dem Wachstumsattribut des berührten Kantenstückes die Wachstumsrichtung des berührten Linienstückes. Ebenso ermittelt man aus dem Richtungsattribut des berührenden Liniensegmentes und der Angabe ob der Wurzelpunkt Start– oder Endpunkt des Wurzelkantenstückes ist, die Wachstumsrichtung des berührenden Liniensegmentes. Sind beide Richtungen identisch ist die zweite

Bedingung ebenfalls erfüllt.

Die dritte und letzte Bedingung besagt, daß *ein gemeinsamer Stamm zweier Kantenstücke, die zu einem Baum gehören, nicht beliebig kurz werden darf.*

Diese Problematik hängt direkt mit der Struktur der verrundeten Linienzüge zusammen. Wie zuvor beschrieben handelt es sich hier um eine alternierende Folge von Linien– und Bogensegmenten der Form *Linie, Bogen, ..., Linie.* Das Verrunden eines Linienzuges kann man sich so vorstellen, daß, ausgehend vom einem zunächst unverrundeten Linienzug, die Liniensegmente beidseitig jeden Eckpunktes um eine vorgegebene Strecke verkürzt werden. Die so entstandenen Lücken werden mit tangential anschließenden Bogenstücken überbrückt. Das Problem entsteht daraus, daß die Liniensegmente dabei beliebig kurz, das heißt, punktförmig werden dürfen. Bild–24 zeigt ein Beispiel, in dem die sonst unsichtbaren punktförmigen Liniensegmente vergrößert eingezeichnet sind.



Bild–24: Vereinigung von Kantenstücken mit punktförmigem Startsegment

Alle drei Kantenstücke beginnen mit einem Liniensegment der Länge Null. Man erkennt, daß sowohl Kantenstück A und B als auch Kantenstück B und C alle bisher eingeführten grafischen Bedingungen für eine Vereinigung erfüllen. Es spricht sicherlich nichts dagegen, B und C zu einem Baum zu vereinigen. Man erhält einen Stamm ausgehend von Segment B1 bis zu der Verzweigung in den Segmenten B5 und C3. Anders sieht die Situation bei den Kantenstücken A und B aus. Beide Linienstücke verzweigen auch sofort im Kontaktpunkt, aber anders als im ersten Fall kann man keinen Stamm bestimmen. Es wäre sinnlos, formal einen unsichtbaren Stamm der Länge Null einzuführen – wie sollte ein solcher Stamm bearbeitet werden ? Aus diesem Grund findet in diesen Fällen keine Vereinigung statt.

In Bild–A15 ist dargestellt, wie diese Bedingung überprüft werden kann. Dazu zuerst ist es nötig, den Begriff des *Verlaufes* zu einzuführen. Die Verlaufsrichtung ist für Kantenstücke definiert, bei denen der Kontaktpunkt auf einem ihrer Randpunkte, also entweder auf dem Start– oder auf dem Endpunkt liegt. Unter Verlauf soll in diesen Fällen die Richtung verstanden werden, in man sich

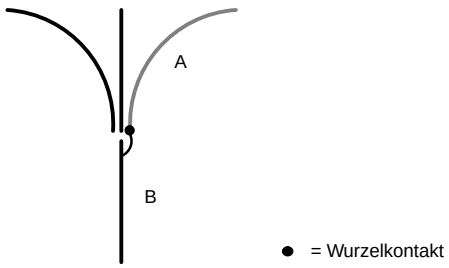
ausgehend von diesem Randpunkt hin zum anderen Ende bewegt[22].

Sowohl bei entgegengesetztem Verlauf zweier stumpf aneinander stoßender Kantenstücke, als auch bei sich überlappenden Kantenstücken tritt die Stammproblematik nicht auf (vgl. Bild–A15 – Situation rechts bzw. links außen). Zu dem geschilderten Problem kann es nur kommen, wenn beide Kantenstücke sich in einem Randpunkt berühren und bezogen auf diesen Randpunkt denselben Verlauf haben. Abhängig davon, welches der beiden am Wurzelkontakt beteiligten Liniensegmente zu einem Punkt entartet ist, können drei Situationen unterschieden werden. Unkritisch ist der Fall dann, wenn beide Segmente echte Linien sind. In diesem Fall ist für den gemeinsamen Teil beider Kantenstücke ein gemeinsames Kantenstück definiert. Dies ist ansonsten nur noch für den Fall erfüllt, daß beide Linienzüge mit einem punktförmigen Liniensegment beginnen und noch im Startpunkt in dieselbe Richtung abbiegen. In allen anderen Fällen ist das Stammkriterium nicht erfüllt.

Nur wenn ein Wurzelkontakt alle zuvor genannten Bedingungen erfüllt, wird nun mit dem Vereinigen der beiden Bäume begonnen.

1.3.3.5 Vereinigen von Kantenstückbäumen

Im folgenden treten nur noch Kontaktsituationen auf, die eine Vereinigung der beiden beteiligten Bäume zulassen. Ausgehend von der vorangehenden Überprüfung der dritten Bedingung kann man einen einfachen Sonderfall identifizieren, der getrennt von allen übrigen Fällen betrachtet werden muß. Haben zwei Kantenstücke einen gemeinsamen Startpunkt, aber entgegengesetzten Verlauf (linke Situation in Bild–A15) kann es sich entweder um eine einfache Verlängerung eines Kantenstückes oder aber um den in Bild–25 gezeigten Fall handeln.



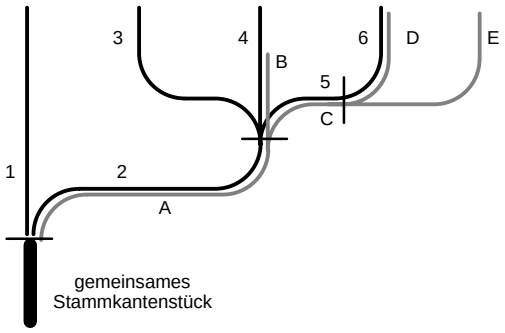
Bild–25: Einzugliederndes Kantenstück wird zum Ast

Es handelt sich um einen Wurzelkontakt eines Kantenstückes A zu einem Stammkantenstück mit mehreren Ästen in dessen Verzweigungsendpunkt. Zum Vereinigen beider Bäume wird bei Kantenstück A als Wurzelattribut ein Verweis auf das Kantenstück B eingetragen. Umgekehrt wird bei diesem als weiterer Ast ein Verweis auf das Kantenstück A hinzugefügt. Ist das Kantenstück A der Stamm weiterer Äste, wurde somit der gesamte Baum mit der Wurzel A als Teilbaum auf das Stammstück B sozusagen aufgepfropft. Hat Kantenstück A keine weiteren Äste, so wird

22.. Für Wurzelkantenstücke stimmt diese Richtung grundsätzlich mit der Wachstumsrichtung überein.

durch das Vereinigen erstmals das Wachstumsattribut definiert. Dieses muß dann, je nachdem, ob der Wurzelkontakt den Start– oder Endpunkt von A betrifft, entsprechend gesetzt werden. Die eigentliche Vereinigung für diesen Wurzelkontakt ist hiermit abgeschlossen. Anschließend muß noch die entsprechende Kontaktbeschreibung aus der Kontaktliste entfernt werden und wie zuvor beschrieben, all die Kontaktbeschreibungen entfernt werden, die aufgrund der vorgenommen Vereinigung zu Maschen führen würden.

Es ist möglich, daß das als neuer Ast zum Baum dazugekommene Kantenstück A auf einem der anderen Äste des ursprünglichen Teilbaumes mit dem Stamm B liegt. Allgemein formuliert gilt in diesen Fällen, daß Teilbäume ein und desselben Baumes ausgehend von einem gemeinsamen Stammstück ganz oder teilweise übereinander liegen. Bild–26 zeigt ein Beispiel.



Bild–26: Vereinigung übereinanderliegender Bäume

Beginnend ab dem gemeinsames Stammkantenstück liegt der Teilbaum aus den Kantenstücken A, B, C, D und E über einem Teilbaum des ursprünglichen Baumes aus den Kantenstücken 2, 3, 4, 5 und 6. Die übereinanderliegenden Teile müssen selbstverständlich miteinander vereint werden. Definiert man den gemeinsamen wurzelseitigen Endpunkt des Stammstückes solcher Teilbäume erneut als Wurzelpunkt und legt eine Kontaktbeschreibung für die beiden Stammstücken zu diesem Wurzelpunkt an, leitet man damit eine eventuelle Vereinigung der Teilbäume ein. Da man nicht wissen kann, zwischen welchem der Teilbäume des ursprünglichen Baumes und dem neu einzugliedernden Teilbaum eine Vereinigung möglich ist, legt man Kontaktbeschreibungen für alle möglichen Kombinationen an. Aufgrund der grafischen Vereinigungsbedingungen führen nur solche Kontaktbeschreibungen zu einer Vereinigung, die übereinander liegende Äste beschreiben. Im gezeigten Beispiel wird man so eine Kontaktbeschreibung für einen Kontakt zwischen Kantenstück 1 und A angelegen und eine weitere für einen Kontakt zwischen 2 und A, wobei nur letztere zu einer Vereinigung führen würde.

Diese Idee kann man nutzen, um rekursiv zwei übereinanderliegende Bäume zu vereinigen. Man vereinigt von einem Wurzelkontakt ausgehend zunächst immer nur die beiden direkt beteiligten Kantenstücke bis zu der Stelle, an der sich der gemeinsame Baum verzweigt. Ist es erkennbar, daß gemeinsame Fortsetzungen von Ästen beider Teilbäume über diese Stelle hinweg möglich sind, legt man für alle Astkombinationen zwischen diesen Teilbäumen jeweils eine weitere Wurzelkontaktbeschreibung an. Dieses Verfahren wiederholt sich nötigenfalls an jeder weiteren

gemeinsamen Verzweigung. Entsprechend der zuvor beschriebenen grafischen Verträglichkeitsprüfung führen anschließend auch hier wieder nur die Kontakte zu einer Vereinigung, die zwei übereinanderliegende Kantenstücke bezeichnen.

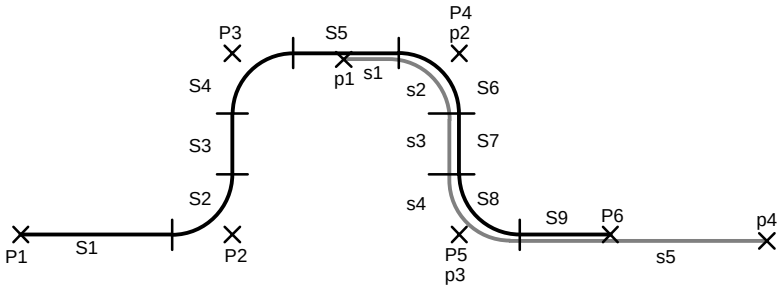
Im Beispiel würden so an der folgenden Verzweigung die folgenden sechs Kontaktbeschreibungen angelegt: (3, B), (3, C), (4, B), (4, C), (5, B), (5, C), wovon allerdings nur (4,B) und (5, C) zu einer Vereinigung führen würden. Letzterer Vereinigung würde sich dasselbe Verfahren zum Verschmelzen von Kantenstück 6 mit D anschließen.

Lediglich das Verfahren zur Vermeidung von Maschen muß diesem Konzept angepaßt werden. Da diese neu hinzugekommenen Kontakte grundsätzlich nicht die Gefahr eine Maschenbildung mit sich bringen, müssen diese von der Löschbedingung des Verbotes von Kontakten innerhalb eines Baumes ausgeschlossen werden. Dies kann entweder durch eine entsprechende Analyse der Kontaktbeschreibung geschehen oder, was wesentlich einfacher und vor allem schneller ist, durch eine entsprechende Ausnahmemarkierung solcher Beschreibungen.

Mit Ausnahme der zuvor geschilderten in Bild–25 dargestellten Situation, lassen sich alle übrigen Fälle entsprechend Bild–A16 in sechs verschiedene Situationen A bis F einteilen. Allen Fällen gemeinsam ist die Gliederung in *Vorlauf* und *gemeinsamen Linienzugverlauf*, die zusammen den *gemeinsamen Teil* bilden, und daran anschließend den *individuellen Teil*, wobei in Ausnahmefällen Teile entfallen können. Bei der Vereinigung aller gezeigten Situationen arbeitet man sich in eben dieser Richtung, beginnend am äußersten Teil des Vorlaufes, Schritt für Schritt vor. Dabei wird je nach Fall ein Kantenstück vom anderen einfach geschluckt oder die ursprünglichen Kantenstücke können verlängert bzw. verkürzt werden oder im Falle einer Verzweigung muß zusätzlich ein völlig neues Kantenstück erzeugt werden. Zu Beginn der Vereinigung steht die Information, um welchen der sechs Fälle es sich handelt, noch nicht zur Verfügung. Würde man die Situation zuerst analysieren und anschließend fallweise die Kantenstücke vereinigen, müßte man unter Umständen den gemeinsamen Teil aus Vorlauf und gemeinsamem Verlauf zweimal überarbeiten, wobei man viele Schritte unnötig ein zweites Mal wiederholen müßte. Will man dies umgehen, bietet sich folgendes Verfahren an:

Man legt für jedes der beiden Ausgangskantenstücke und für den gemeinsamen Teil eine Punktliste und eine Segmentliste an. Die beiden Listen für den gemeinsamen Teil sind zu Beginn leer. Die Punktlisten der beiden Ausgangskantenstücke enthalten die jeweilige Folge von Eckpunkten eines Kantenstücklinienzuges. Die Punkte sind dabei entsprechend des Verlaufes des Wurzelkantenstückes, das heißt vom Beginn des Vorlaufes in Richtung des individuellen Teiles, zu sortieren. Ebenso enthalten die Segmentlisten die in gleicher Weise sortierten Verweise auf die entspre-

chenden Kantenstücksegmente[23.]. Bild–27 zeigt dazu ein Beispiel zweier zu vereinigender sich überlappender Kantenstücke nach der Initialisierung der erwähnten Listen.



Segmentliste berührtes Kantenstück:	(S1, S2, S3, S4, S5, S6, S7, S8, S9)
Segmentliste berührendes Kantenstück:	(s1, s2, s3, s4, s5)
Segmentliste gemeinsamer Teil	()
Punktliste berührtes Kantenstück	(P1, P2, P3, P4, P5, P6)
Punktliste berührendes Kantenstück	(p1, p2, p3, p4)
Punktliste gemeinsamer Teil	()

Bild–27: Vereinigungsbeispiel

Schritt für Schritt verfolgt man nun den gemeinsamen Teil und überträgt dabei die entsprechenden Elemente der Listen der ursprünglichen Kantenstücke in Listen des gemeinsamen Teiles. Das Entscheidende ist, daß dabei nur die Listen, zunächst aber nicht die ursprünglichen Kantenstücke verändert werden. Beim Übertragen von Segmenten wird nicht der Verweis auf ein Segment kopiert, sondern der Verweis auf eine eigens dazu angelegte Kopie des Segmentes in der Liste eingetragen. Das Ziel dieses Verfahrens besteht darin, daß man am Abschluß der gesamten Prozedur die vollständige Punktliste und Segmentliste für jedes nach der Vereinigung vorkommende Kantenstück ermittelt hat. Darauf aufbauend kann man fallweise die entsprechenden Kantenstücke löschen, verändern oder erzeugen. Der Zweck der Segmentlisten dabei ist, daß man auf diesem Wege einfach alle aufgeführten Segmente in die Basisgrafikgruppe einfügt und bei bestehenden Kantenstücken alle nicht aufgeführten Segmente aus der Basisgrafikgruppe löscht. Man umgeht durch diese Übernahme bestehender Segmente das zeitaufwendige erneute Erzeugen der verrundeten Linienzüge basierend auf den ermittelten neuen Punktfolgen, welches die Folge eines nur auf den Eckpunktbeschreibungen basierenden Verfahrens wäre. Außerdem ist man durch diese Lösung von der speziellen Implementierung der Bogensegmente unabhängig.

Bevor das Verfahren nun im einzelnen erläutert wird, soll die Richtungsproblematik erörtert werden. Die Tatsache, daß beide Linienzüge in der Verlaufsrichtung des Wurzelkantenstückes vereinigt werden, bedeutet gleichzeitig, daß je nach Situation der Linienzug keines, eines oder beider der Kantenstücke entgegen der Zeichenrichtung bearbeitet werden muß. Dazu muß die Reihenfolge der Elemente in der Segment– bzw. Punktliste des betroffenen Kantenstückes gege-

23.. Um die Reihenfolge der Segmente ermitteln zu können, verfügt jedes Segment über eine seiner Position im Linienzug entsprechende Nummer. Diese Indizes werden beim Erzeugen eines Linienzuges entsprechend gesetzt. Es kann sich nicht darauf verlassen werden, daß die Ordnung der Grafikobjekte innerhalb einer Grafikgruppe irgendwelchen unveränderlichen Regeln, z.B. entsprechend der Reihenfolge des Erzeugens, folgt.

benenfalls umgekehrt werden. Damit dieser Richtungswechsel am Ende wieder rückgängig gemacht werden kann, muß zusätzlich für jedes Kantenstück diese Richtungswechselinformation hinterlegt werden (*start–end–rev*). Damit beim eventuellen Erzeugen eines neuen Kantenstückes, das auf den für den gemeinsamen Teil angelegten Listen basiert, die Reihenfolge der Segmente in Zeichenrichtung mit dem Richtungsattribut der einzelnen Segmente verträglich ist, ist per (willkürlich gewählter) Definition bei allen Richtungsattributen durchgängig die Zeichenrichtung des berührten Kantenstückes für den gemeinsamen Teil maßgeblich. Da die damit verbundenen Konsequenzen keinen Einfluß auf das weitere Verständnis haben, wird dieser Punkt nicht weiter untersucht.

Das Verfahren soll nun im einzelnen erläutert werden.

Bild–A17 zeigt den Vereinigungsablauf für den gemeinsamen Teil. Dieser Ablauf ist in fünf Unterabschnitte gegliedert. Um eine anschauliche Darstellung zu erzielen, wird in dem erwähnten Bild jeder einzelne Schritt als grafischer Übergang dargestellt. Parallel dazu zeigt Bild–A18 entsprechend jeweils den Vereinigungsstand des in Bild–27 gezeigten Beispiels nach jedem der fünf Unterabschnitte.

- Im Initialisierungsteil werden, wie zuvor beschrieben, die Listen angelegt.

- Bei der nun folgenden *Übernahme unberührter Segmente des Vorlaufes* werden die Segmente und Punkte auf den gemeinsamen Teil ”übertragen”, die vor dem ersten berührten Liniensegment liegen. Dazu wird Schritt für Schritt jeder Eintrag der Segmentliste des berührten Kantenstückes ausgewertet, bis man auf das in der Kontaktbeschreibung angegebene Liniensegment trifft. Ist dies das erste Element in der Segmentliste, kann dieser Schritt übersprungen werden. Von jedem Eintrag wird eine Kopie erzeugt und der jeweilige Verweis in der gemeinsamen Segmentliste vermerkt. Der ursprüngliche Verweis in der Segmentliste des berührten Kantenstückes wird gelöscht. Handelt es sich bei dem betrachteten Element um ein Liniensegment, wird gleichermaßen ein Eckpunkt übertragen. Dieses Prinzip wird für die Eckpunkte auch bei allen folgenden Schritten angewandt, so daß der letzte gemeinsame Eckpunkt immer derjenige vor dem letzten gemeinsamen Liniensegment ist.

  Im Beispiel werden dabei die ersten vier Segmente (S1 bis S4) des berührten Kantenstückes in den gemeinsamen Teil übertragen. Das Segment S5 als erstes berührtes Segment wird nicht mehr übernommen.

- Nach dem vorangehenden Schritt sind insgesamt 4 mögliche Situationen denkbar. Um eine einheitliche Behandlung zu ermöglichen, ist der folgende anpassende Zwischenschritt nötig. Die Anpassung sorgt dafür, daß der gemeinsame Verlauf mit einem gemeinsamen Startpunkt der nächsten Segmente von berührtem und berühren-

dem Linienzug beginnt, wobei entweder beide Segmente Liniensegmente oder Bogensegmente darstellen müssen.
Falls der Startpunkt beider Segmente bezogen auf die Betrachtungsrichtung identisch ist, ist somit nichts zu tun, da es sich nur um zwei in demselben Punkt beginnende Liniensegmente handeln kann.
Einen weiteren Sonderfall stellt die Situation dar, daß die zwei sich berührenden Liniensegmente stumpf aufeinander stoßen. In diesem Fall wird einfach das berührte Segment in den gemeinsamen Teil übernommen. Einen gemeinsamen Verlauf beider Linienzüge gibt es in dieser Situation nicht.
In allen anderen Fällen wird der berührte Linienzug dahingehend verkürzt, daß die gemeinsame Startsituation anschließend hergestellt ist. Dabei muß entweder das komplette Liniensegment für den gemeinsamen Teil übernommen werden, wenn es sich um eine Berührung im Endpunkt handelt, oder das Liniensegment wird in zwei Teilsegmente gebrochen, von denen das vordere Segment in den gemeinsamen Teil übernommen wird und das hintere Segment das ursprüngliche berührte Liniensegment ersetzt.

Für alle der drei anpassenden Situationen wird der nächste gemeinsame Eckpunkt von der Punktliste des berührten Kantenstückes übernommen.

Im Beispiel muß eine Anpassung der beiden sich überlappenden Liniensegmente S5 und s1 stattfinden. Der nicht überlappende vordere Teil von S5 wird zu einem neuen Segment S5a, der dem gemeinsamen Teil zugeordnet wird. Der mit s1 überlappende Teil von S5 wird zu dem neuen Segment S5b, der dem ursprünglichen berührten Linienzug zugeordnet wird, so daß die beiden ursprünglichen Linienzüge jetzt eine Folge deckungsgleicher Segmente beschreiben.

- Anschließend erfolgt analog dem ersten Schritt die Übernahme aller folgenden dekkungsgleichen Segmente in den gemeinsamen Teil. Solange die Segmente beider Listen grafisch identisch sind oder bis man am Ende einer Liste angelangt ist, wird dabei jeweils für zwei identische Segmente der entsprechende Verweis aus den Listen der ursprünglichen Segmente entfernt und der Verweis einer Kopie eines dieser Segmente in der Liste für den gemeinsamen Teil abgelegt. Wie bei den Schritten zuvor, muß auch hier immer wenn es sich um ein Liniensegment handelt, zudem der nächste Eckpunkt in den gemeinsamen Teil übertragen werden.
  Beim allerersten Schritt kann eine Ausnahmebehandlung erforderlich sein. Dies ist dann der Fall, wenn durch das Zerlegen eines Liniensegmentes in zwei Teile aufgrund einer Überlappung zwei in einer Geraden liegende Linien direkt aufeinander folgen würden. Da dies ein Verstoß gegen die alternierende Folge von Linien und Bögen darstellen würde, müssen solche Linien durch ein einziges Liniensegment im

gemeinsamen Teil ersetzt werden. Ebenso muß natürlich der Eckpunkt, der den Startpunkt der zweiten Linie spezifiziert, übergangen werden[24.].

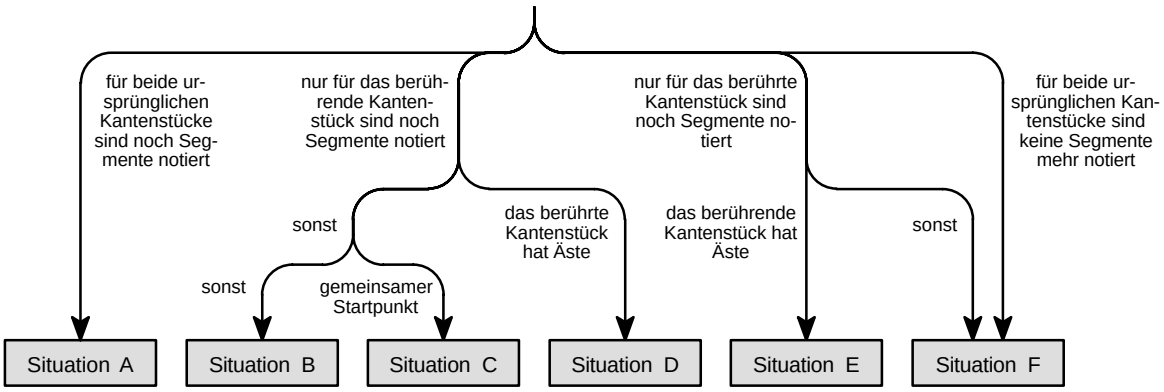
Im Beispiel wird der gemeinsame Teil so bis zu den letzten zueinander deckungsgleichen Segmenten s4 und S8 fortgesetzt. Das erste Segment dieses deckungsgleichen Abschnittes, das Liniensegment S5b, wird hierbei mit dem unmittelbar vorangehenden Liniensegment S5a zu einem einzigen Liniensegment (S5ab) vereinigt.

- Der folgende Schritt entfällt, wenn mindestens eine der beiden ursprünglichen Kantenstücksegmentlisten keine weiteren Elemente enthält. Ansonsten weisen die folgenden Elemente beider Listen ein unterschiedliches Aussehen auf. Ähnlich wie im zweiten Schritt ist dann möglicherweise ein Anpassungsschritt nötig, der ausgehend von hier insgesamt 5 möglichen Situationen eine einheitliche anschließende Behandlung ermöglicht.
  Führen beide Segmente in eine unterschiedliche Richtung, so gehören beide nicht mehr zum gemeinsamen Teil beider Kantenstücke und es muß nichts geschehen.
  In allen anderen Fällen handelt es sich um zwei Liniensegmente von denen jeweils eines das andere ausgehend vom gemeinsamen Startpunkt (bezogen auf die Betrachtungsrichtung) teilweise überlappt. Ist eines dieser Liniensegmente zu einem Punkt entartet, kann dieses einfach verworfen werden. In den anderen Fällen wird, wie schon zuvor beim zweiten Schritt, das längere Segment in zwei Stücke aufgespalten. Der erste Teil verläuft vom Startpunkt bis zum Ende der Überlappung und wird dem gemeinsamen Teil zugeordnet. Der andere Teil ersetzt das ursprüngliche Segment in der jeweiligen ursprünglichen Linienzugbeschreibung. In allen vier Fällen wird der nächste Eckpunkt der gemeinsamen Punktliste zugeordnet.

  Im Beispiel überlappen sich die beiden Endsegmente S9 und s5 der ursprünglichen Linienzüge. Da S9 aber vor s5 endet ist eine Anpassung nötig. Dabei wird S9 in den gemeinsamen Teil übernommen und s5 entsprechend verkürzt.

24.. Ursprünglich war geplant gewesen, daß auch nicht verrundete Linienzüge bearbeitet hätten werden können. In diesem Fall reicht es nicht aus, zwei aufeinander folgende Linien zu erkennen, um folgern zu können, daß diese auf einer Geraden liegen. Zusätzlich muß die Richtung beider Linien explizit überprüft werden. Um nicht den ersten Schritt separat implementieren zu müssen und um sich andererseits den Testaufwand für jeden Durchlauf zu ersparen, wurde ein Flag eingeführt *(splice–lines)*. Dieses Flag wird gesetzt, wenn es bei einem Anpassungsschritt erkennbar ist, daß es zu einer solchen Situation kommen kann. Nur wenn dieses Flag gesetzt ist, wird die verhältnismäßig aufwendige Richtungsüberprüfung durchgeführt. Nach einer solchen Überprüfung wird unabhängig vom Ergebnis das Flag zurückgesetzt und für alle weiteren Schritte die Überprüfung umgangen.

In allen Vereinigungsfällen kann ausgehend von diesem Zwischenergebnis, wie Bild–28 zeigt, leicht bestimmt werden, um welchen der sechs in Bild–A16 dargestellten Fälle es sich handelt.



Bild–28: Identifizieren der Vereinigungssituationen A bis F

Basierend auf dem erreichten Zustand der Segment– und Punktlisten kann nun der situationsspezifische Abschluß des Vereinigungsprozesses der zwei Kantenstücke erfolgen.

- Situation A: *Die Linienzüge beider Kantenstücke verzweigen sich.* Für den gemeinsamen Teil entsteht ein neues Kantenstück, die beiden alten Kantenstücke müssen dementsprechend verkürzt werden. Ergänzt man alle drei Punktlisten um den Endpunkt des letzten gemeinsamen Segmentes, ist die Beschreibung der Eckpunkte für alle drei Kantenstücke vollständig. Ebenso sind die Segmentlisten meist bereits vollständig. Eventuell ist eine Anpassung an die alternierende Struktur durch das Hinzufügen eines punktförmigen Liniensegmentes von Nöten. Zur Anpassung der bereits bestehenden Kantenstücke werden die alten Punktlisten gegen die zuvor ermittelten neuen Punktlisten ersetzt. Um den Linienzug gemäß der entsprechenden ermittelten Segmentliste anzupassen, werden Segmente, die nicht mehr in der Beschreibung enthalten sind, aus der Basisgrafikgruppe entfernt und Segmente, die in der Segmentliste neu hinzugekommen sind, werden hinzugefügt. Das neue Kantenstück wird basierend auf der Punktliste und der Segmentliste für den gemeinsamen Teil erzeugt. Zum Erzeugen des Linienzuges werden alle in der Segmentliste des gemeinsamen Teils aufgeführten Segmente in die neue Basisgrafikgruppe eingefügt.

  Bei allen beteiligten Kantenstücken ist gegebenenfalls eine Anpassung der Kontrollendpunkte (siehe Kapitel 1.2.5) zur Positionierung im Raster des Planes erforderlich. Diejenigen Kantenstückkontrollendpunkte, die nach dem Vereinigungsprozeß immer noch Endpunkte darstellen, bleiben bestehen, wobei der Kontrollendpunkt des gemeinsamen Stammlinienzuges dem Stammkantenstück zugeordnet wird. Am Verzweigungspunkt wird für jedes Kantenstück ein neuer Kontrollpunkt erzeugt. Da bis auf Anfangs– und Endpunkt eines Kantenstücklinienzuges, die eventuell durch Anziehung auf den Rand eines Knotens aus dem Raster geraten sein können, alle

übrigen Endpunkte von Segmenten im Raster liegen, ist sichergestellt, daß auch der Verzweigungspunkt im Raster liegt.

Um die Kantenstücke zu einer Baumstruktur zu verbinden, müssen nun entsprechend die Ast– und Stammreferenzen eingetragen werden. Für die Blattenden der beiden alten Kantenstücke ändert sich nichts. Hatte hingegen eines der beiden Kantenstücke vor dem Vereinigen einen Stamm, oder hatten möglicherweise beide denselben Stamm, so wird ein Verweis auf diesen Stamm bei dem neuen Kantenstück eingetragen. Umgekehrt werden bei dem alten Stamm die Blattverweise auf die beiden alten Kantenstücke entfernt und statt dessen das durch die Vereinigung neu entstandene Kantenstück als Ast vermerkt. Da das neue Kantenstück als Stamm der beiden alten Kantenstücke anzusehen ist, müssen auch zwischen diesen Kantenstücken die entsprechenden Verweise eingetragen werden. Daraus folgt, daß ebenfalls das Wachstumsattribut des neuen Kantenstückes gesetzt werden muß.

Darüber hinaus erbt der neue Stamm der beiden alten Kantenstücke möglicherweise zusätzliche Attribute. Hierbei muß man zwischen solchen Attributen unterscheiden, die nur kopiert werden und solchen die den alten Kantenstücken entnommen werden und die dem neuen Kantenstück zugeordnet werden. Um welche Attribute es sich hierbei handelt, ist vom Typ der betrachteten Kantenstücke abhängig. Lediglich das Verrundungsattribut ist für alle Kantenstücke gleichermaßen definiert. Dieses Attribut wird für das neue Kantenstück lediglich kopiert. Bei gerichteten SPIKES–Kantenstücken existiert eine Attributgruppe zusätzlicher Pfeilspitzen. Diejenigen der darin enthaltenen Pfeilspitzen des zu beerbenden Kantenstückes, die auf dem Linienzug des erbenden Kantenstückes liegen, werden übernommen.
Für die logische Auswertung wird das Konzept interessant, wenn die Kantenstücke über formal auswertbare Attribute verfügen. Zum Beispiel wurde für Petrinetze definiert, daß die Bedingungstexte grundsätzlich auf auf den gemeinsamen Stamm übergehen. Da die verschiedenen Kantenstücktypen der unterschiedlichen Typen von SPIKES–Plänen jedoch über unterschiedliche Attribute verfügen, die eine unterschiedliche Behandlung erfordern, soll im Rahmen dieser allgemeinen Betrachtung nicht weiter auf diesen Punkt eingegangen werden.

Die eigentliche Vereinigung der beiden alten Kantenstücke ist hiermit abgeschlossen. Die folgenden Aktionen sind im Prinzip für alle anderen Fälle identisch. Die Kontaktbeschreibung, die zu dieser Vereinigung geführt hat, ebenso, wie die Beschreibungen der aufgrund dieser Vereinigung maschenbildenden Kontakte müssen entfernt werden. Bezieht sich ein beschriebener Kontakt auf den Teil des alten Kantenstückes, der durch das neue Kantenstück ersetzt wurde, ist der Verweis entsprechend auf das neue Kantenstück umzusetzen.

Die übrigen Situationen sollen nur noch insoweit beschrieben werden, als es sich um situationstypische Schritte handelt.

- Situation B: *das berührende Kantenstück enthält das berührte (astlose) Kantenstück.* In diesem Fall ist der Linienzug des berührten Kantenstückes überflüssig. Das berührende Kantenstück übernimmt die möglichen Attribute und gegebenenfalls den Stamm des berührten Kantenstückes (gleicher Startpunkt), welches danach gelöscht werden kann. Ebenfalls sind die für den gemeinsamen Teil angelegten Segmente überflüssig und können gelöscht werden.

- Situation C: *das berührende Kantenstück stellt eine überlappende Verlängerung des berührten Kantenstückes dar.* Im Sonderfall (Situation–2 aus Bild–25) stoßen beide Kantenstücke stumpf aneinander. Den Linienzug des verlängerten Kantenstückes erhält man, wenn man den verbleibenden Teil des berührenden Kantenstückes in die Beschreibung für den gemeinsamen Teil aufnimmt. Haben die Linienzüge beider Kantenstücke unterschiedliche Zeichenrichtung müssen dazu die Richtungsattribute der betroffenen Segmente angepaßt werden.

  Das in Bild–A18 betrachtete Beispiel entspricht der eben beschriebenen Situation. Das verbleibende Segment s5b definiert den gesamten verbleibenden Rest des berührenden Linienzug, der vollständig in den gemeinsamen Teil zu übernehmen ist. Da hierdurch zwei Liniensegmente, S9’ und s5b, in gleicher Richtung unmittelbar aufeinander folgen würden, werden diese zuvor zu einem einzigen Liniensegment (s5b9) verschmolzen werden.

  Diese Beschreibung dient dem berührten Kantenstück als neue Linienzugdefinition. Von dem berührenden Kantenstück werden neben eventuellen typspezifischen Attributen mögliche Blätter übernommen. Danach kann das berührende Kantenstück aus dem Plan entfernt werden.

- Situation D: *Das berührte Kantenstück endet auf dem berührenden Kantenstück und hat im Endpunkt eine Verzweigung zu mehreren Ästen*. Der Linienzug des berührten Kantenstückes bleibt unverändert. Der Linienzug des berührenden Kantenstück wird entsprechend der verbleibenden Listenelemente für das berührende Kantenstück gekürzt. Das berührte Kantenstück stellt damit den Stamm für das berührende Kantenstück dar. In dieser Situation ist es denkbar, daß der neue Ast vom Stamm ausgehend ganz oder bis zu einem gewissen Punkt deckungsgleich mit einem alten Ast des berührten Kantenstückes verläuft. Um eine Vereinigung solcher Äste zu ermöglichen, werden Wurzelkontaktbeschreibungen von allen alten Ästen zu dem neuen Ast angelegt.

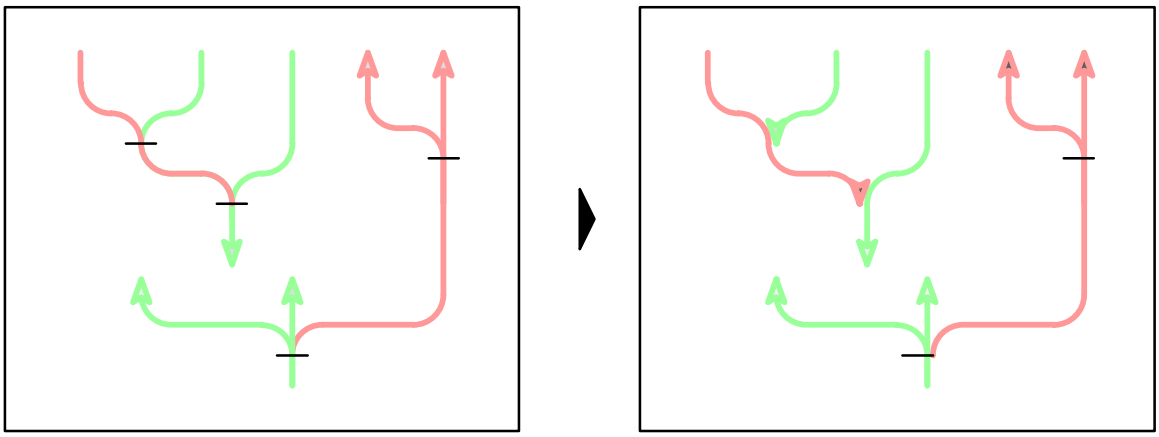
- Situation E: *Das berührende Kantenstück endet auf dem berührten Kantenstück und hat im Endpunkt eine Verzweigung zu mehreren Ästen*. Diese Situation entspricht Situation D, nur daß die Rolle von berührtem und berührendem Kantenstück und die damit verbundenen Schritte exakt vertauscht sind.
- Situation F: *Das berührte Kantenstück enthält das berührende Kantenstück*. Diese Situation entspricht im Wesentlichen der Situation B, jedoch sind die Rollen der Kantenstücke vertauscht. Zusätzlich ist in diese Situation die Möglichkeit aufgenommen worden, daß beide Kantenstücke im selben Punkt mit Ästen enden können. Der Unterschied rührt daher, daß aufgrund der gewählten, in Bild–28 gezeigten Klassifizierung der Vereinigungssituationen, bei Situation A das Enthaltensein ein Überstehen des berührenden Linienzuges über den berührten Linienzug bedingt. Hierdurch sind in dieser Situation keine Äste des berührten Kantenstückes möglich. Bei Situation F hingegen gibt es diese Einschränkungen bezüglich des Enthaltenseins nicht, wodurch der Unterschied auftritt. Es handelt sich hierbei um die Situation, die auftritt, wenn (Teil–)Bäume mit Ästen direkt übereinanderliegen. Haben beide Kantenstücke Äste, so folgt daraus, daß für jede Kombination zwischen einem Ast des berührenden Stammkantenstückes und einem Ast der berührten Stammkantenstükkes eine Wurzelkontaktbeschreibung angelegt werden muß.

Nach Abarbeiten aller Kontaktbeschreibungen ist der eigentliche Eingliederungsprozesses eines Baumes abgeschlossen. In einer Schleife über alle verbliebenen alten und neuen Kantenstücke des Baumes werden abschließend die Attribute angepaßt, die von der Position und Richtung eines Kantenstückes im Baumes abhängig sind[25.]. Bei Bäumen aus gerichteten Kantenstücken bedeutet dies, daß entweder alle Blattkantenstücke oder aber das Wurzelkantenstück eine Pfeilspitze erhalten. Alle anderen Kantenstücke dürfen keine Pfeilspitze besitzen.

1.3.4 Heraustrennen von Kantenstücken

Werden ein oder mehrere Kantenstücke eines SPIKES–Planes ausgeschnitten, so können davon ganze Bäume oder Teile eines oder mehrere Bäume betroffen sein. Stämme können einen oder mehrere Äste verlieren. In Sonderfällen wird sogar die Minimalform zerstört. Dies ist dann der Fall, wenn alle Äste eines Stammes bis auf einen entfernt werden. Ebenso können sich von der Position im Baum abhängige Attribute ändern, wie z.B. Pfeilspitzen bei gerichteten Kantenstükken. Die Aufgabe besteht allgemein formuliert darin, aus einer gegebenen Menge von Kantenstückbäumen in Minimalform durch Trennen dieser Menge in zurückbleibende und auszuschneidende Kantenbäume zwei neue Mengen von Kantenstückbäumen in Minimalform zu erzeugen (vgl. Bild–29).

25.. In der gegenwärtigen Implementierung des Editors wird davon ausgegangen, daß von der Lage im Baum abhängige Attribute nur für Blatt– oder Wurzelkantenstücke definiert sind. Ein Beispiel dafür, daß dies auch anders sein kann, zeigen die meisten in der Natur vorkommenden Bäume. Bei diesen werden die Äste ausgehend vom Stamm bis hin zu den Blattstielen immer dünner.



Bild–29: Ausgliederungsbeispiel

Dieser Trennungsprozeß ist ebenfalls erforderlich, bevor Kantenstücke bewegt oder kopiert werden. Das Bewegen kann dabei als Ausschneiden und anschließendes Einfügen aufgefaßt werden. Das anschließende Einfügen in einen Plan, egal ob nach explizitem Ausschneiden, Kopieren oder interaktivem Bewegen ist der Grund, warum nicht nur die Baumstruktur der betroffenen zurückbleibenden Kantenstückbäume, sondern auch die der auszuschneidenden Kantenstückbäume angepaßt werden muß. Werden die ausgeschnittenen oder kopierten (Teil–)Bäume dann wieder in einen SPIKES–Plan gleichen Typs eingegliedert, kann dann der zuvor beschriebene Eingliederungsprozeß angewendet werden.

Da es im Zusammenhang mit dem Kopieren, aber auch beim Bewegen nicht sehr hilfreich ist, von auszuschneidenden Kantenstücken zu reden, werden diese Kantenstücke deshalb besser als *zu lösende* Kantenstücke bezeichnet. Die anderen Kantenstücke des Planes sind direkt nach der Trennung zunächst nicht von Bedeutung. Sie werden deshalb weiterhin als *zurückbleibende* Kantenstücke bezeichnet. Es ist wichtig, sich zu verdeutlichen, daß durch den Trennungsprozeß Elemente beider Mengen durch Vereinigung von Stämmen mit nur noch einem Ast durch neue gemeinsame Kantenstücke ersetzt werden müssen. Es können also Elemente gelöscht werden und andere neu hinzukommen. Allen anschließenden Aktionen, wie löschen, bewegen oder kopieren, beziehen sich auf diese durch das Heraustrennen *zu lösender* Plankomponenten entstandene Menge von Plankomponenten, die hier als *gelöste* Plankomponenten bezeichnet werden.

Bild–29 zeigt ein Beispiel. Man erkennt zwei Kantenstückbäume, aus denen die fünf rot markierten Kantenstücke herausgelöst werden sollen. Nach dem Trennungsprozeß liegen vier gelöste Kantenstücke vor, die zwei von den zurückbleibenden Kantenstücken unabhängige Bäume darstellen. Diese stehen nun für den eigentlichen Editionsschritt zur Verfügung. Die zurückbleibenden Kantenstücke bilden ihrerseits drei neue Kantenstückbäume, wovon zwei lediglich aus ei-

nem einzigen isolierten Kantenstück bestehen. Bei einem davon (mittlerer Baum) ist durch das Vereinigen vom einem Stamm und einem Ast ein neues Kantenstück entstanden.

Bild–A19 zeigt den Grobablauf des des Trennungsvorganges. Allgemein ist von einer Liste zu lösender Plankomponenten auszugehen, in der auch Elemente enthalten sein können, die keine Kantenstücke sind. Dies ist immer dann der Fall, wenn eine Menge von Plankomponenten unterschiedlichen Typs gelöscht, bewegt oder kopiert werden soll. Diese Nichtkantenstücke sind von dem Trennungsvorgang nicht betroffen und sind somit grundsätzlich als gelöste Plankomponenten anzusehen.

Basierend auf der oben beschriebenen Aufgabenstellung besteht die Idee darin, in einem ersten Schritt zwei disjunkte Mengen von gelösten und zurückbleibenden Kantenstückbäumen herzustellen. Dazu werden einfach alle Referenzen zwischen zu lösenden und zurückbleibenden Kantenstücken entfernt. Eventuelle Strukturanpassungen bezüglich Minimalform oder positionsabhängiger Attribute werden zunächst gänzlich vernachlässigt. In einem zweiten Schritt werden dann für alle gelösten und zurückbleibenden Kantenstückbäume die nötigen Anpassungen durchgeführt. Diese Trennung ermöglicht eine überschaubare Lösung bei gleichzeitig verhältnismäßig geringem Rechenaufwand.

Für dieses Verfahren sind insgesamt vier Listen nötig. Die erste Liste enthält alle zu lösenden Plankomponenten vor dem Trennungsprozeß. Zwei weitere Listen werden benötigt um die Verweise auf die Wurzelkantenstücke einerseits der gelösten und andererseits der zurückbleibenden Bäume nach dem ersten Schritt zu vermerken. In einer vierten Liste werden alle gelösten Plankomponenten vermerkt, die anschließend weiterbearbeitet werden können.

Im ersten Schritt werden die Nichtkantenstücke direkt aus der Liste der gegebenen gelösten Plankomponenten in die Liste der endgültigen Plankomponenten übertragen. Handelt es sich bei einer gegebenen zu lösenden Plankomponente um ein Kantenstück, so müssen in dem durch dieses Kantenstück bezeichneten Baum alle Referenzen zwischen zu lösenden und zurückbleibenden Kantenstücken entfernt werden. Bild–A20 zeigt dazu den detaillierten Ablauf:

Zu dem gegebenen Kantenstück wird zunächst das Wurzelkantenstück bestimmt. Ausgehend von dieser Wurzel wird der ganze daraus hervorgehende Kantenstückbaum untersucht. Jedes Kantenstück wird überprüft, ob es über Äste verfügt. Ist dies der Fall, so kann es sein, daß von den beiden Kantenstücken, Ast und Stamm, nur eines der Menge der zu lösenden Plankomponenten angehört. In diesem Fall müssen Stamm und Ast voneinander getrennt werden, wozu die wechselseitigen Verweise aufeinander entfernt werden. Der Stamm verliert einen Ast und der Ast verliert seinen Stamm. Auf dieselbe Art und Weise überprüft man anschließend den Ast auf eventuelle weitere Äste seinerseits, danach den Nachbarast usw. bis man schließlich rekursiv den ganzen ursprünglichen Baum in gelöste und zurückbleibende Teilbäume zerlegt hat. Dabei muß immer wenn ein Wurzelkantenstück auftritt, sei es das Wurzelkantenstück des ursprünglichen

Baumes oder ein neues Wurzelkantenstück, das durch Löschen seines ursprünglichen Stammes dazu wurde, dieses entsprechend seiner Zugehörigkeit in der Liste für zu lösende bzw. zurückbleibende Wurzelkantenstücke vermerkt werden. Alle zu lösenden Kantenstücke, die man bei dieser Prozedur findet, gelten als berücksichtigt und können aus der Liste der ursprünglichen der zu lösenden Plankomponenten entfernt werden.

Ist diese Liste abgearbeitet, wird man zum zweiten Schritt übergehen. Basierend auf den Listen für die kalten und heißen Wurzelkantenstücke können die neu entstandenen Bäume überarbeitet werden. Bild–A21 zeigt den Anpassungsvorgang, wie er für jeden Baum, dessen Wurzelkantenstück im ersten Schritt notiert wurde, durchzuführen ist:

Beginnend an der Wurzel zunächst überprüft, ob eine Anpassung der wurzeltypischen Attribute Wurzelkantenstückes nötig ist und diese gegebenenfalls vorgenommen. Eine solche Anpassung ist möglicherweise dann erforderlich, wenn es sich bei einem Wurzelkantenstück um einen ursprünglichen Ast handelt.

Danach wird der übrige Baum von der Wurzel ausgehend auf Fehler in der Minimalformstruktur untersucht. Ein solcher Fehler ist dann aufgetreten, wenn das Heraustrennen von Ästen dazu geführt hat, daß ein Stammkantenstück über genau ein Astkantenstück verfügt. Von der grafischen Struktur her betrachtet, beschreibt dieses Gebilde aber nur einen Ast. Deshalb müssen die beiden Kantenstücke vereinigt werden. Man legt dazu eine vollständige Wurzelkontaktbeschreibung an. Da beide Kantenstücke aufgrund ihrer Lage zueinander als Stamm und Ast in einem Baum verträglich sein müssen, kann damit auf einen Verträglichkeitstest verzichtet werden. Die beiden Kantenstücke werden entsprechend dem im vorangehenden Kapitel beschriebenen Vereinigungsprozeß verschmolzen und die positionsrelevanten Attribute werden angepaßt. Danach geht die Prozedur weiter, indem man nun die Äste des neuen Kantenstückes bestimmt und die nächste Verzweigungsebene überprüft.

Hat ein Kantenstück keine Äste, muß an der eigentlichen Baumstruktur nichts verändert werden. Es ist aber möglich, daß dieses Kantenstück dadurch zum Blatt geworden ist, daß alle seine ursprünglichen Äste entfernt wurden. Deshalb werden auch für solche Kantenstücke die von der Lage im Baum abhängigen Attribute überprüft und gegebenenfalls angepaßt. Verfügt ein solches Blattkantenstück über keinen Stamm, ist sein Wachstumsattribut nicht definiert und muß gegebenenfalls gelöscht werden.

Verfügt ein Kantenstück über mehrere Äste, so sind an dieser Stelle keine Anpassungs– oder Korrekturschritte am Baum nötig. Nacheinander werden Ast für Ast rekursiv alle weiteren Verzweigungsebenen überprüft, bis man, an allen Blättern angelangt, den ganzen Baum überarbeitet hat.

Damit alle Kantenstücke eines überarbeiteten Baumes zur weiteren Bearbeitung greifbar sind, werden in den Fällen, in denen feststeht, daß sich ein Kantenstück nicht mehr verändert, diese in einer Liste endgültiger Baumkomponenten vermerkt. Ist die Anpassungsprozedur eines Baumes beendet, enthält diese Liste alle endgültigen Kantenstücke dieses Baumes. Bei gelösten Bäu-

men sind diese Kantenstücke in der Liste der gelösten Plankomponenten zu vermerken. Sind alle gelösten und zurückbleibenden Bäume überarbeitet, werden die als gelöst vermerkten Plankomponenten zur weiteren Bearbeitung ausgewählt. Sie können jetzt gelöscht, bewegt oder kopiert werden.

1.3.5 Implementierungskonzept

In den vorangegangenen zwei Kapiteln wurde ausführlich der Ablauf des Ein– und Ausgliederns von Kantenstücken beschriebenen. Im folgenden werden anhand von Bild–A22 die dazu benötigten Akteure im Editorsystem und ihre Beziehungen untereinander dargestellt. Das Bild basiert auf dem in Kapitel 1.2.3 vorgestellten implementierungsnahen Aufbaumodell (siehe Bild–A9). Der grafische Ein–/Ausgabedienst, der Akteurs für animierte Edition, der Steuerakteur, sowie der bereichszuständige Menüanforderungsakteur, deren Funktionsweise bereits ausführlich erklärt wurden, sind in diesem Modell in dem *erweiterten Ein–/Ausgabedienst* vereinigt. Weggefallen sind die Defaultwertverwalter, insofern sie nichts zum Verständnis der Kantenstückproblematik beitragen und als Teil der Klassenakteure angesehen werden können. Desweiteren bleiben in diesem Modell die Editionsakteure unberücksichtigt. Dies ist zulässig, da diese sowieso nur auf Bedarf hin erzeugt werden und danach wieder entfernt werden. Da sich die Kantenstückproblematik auf oberstem Planniveau abspielt, werden keine Editionsakteure benötigt. Neu eingeführt wurde hingegen der *Kantenstückbaumakteur*[26.], der im bisherigen Bild als Teil des Planeditors angesehen werden konnte. Um die Rolle dieses neuen Akteurs, die der Kantenstückakteure und des Planeditors beim Ein– und Ausgliedern zu verdeutlichen, wird dieser Vorgang anhand des Bildes noch einmal erläutert[27.].

Soll ein bereits bestehendes Kantenstück ausgegliedert werden, kann dies zwei Gründe haben. Entweder der Benutzer möchte den Linienzug eines einzelnen Kantenstückes bearbeiten. In diesem Fall wird der Kantenstückakteur eine Ausgliederung beim Kantenstückbaumakteur beantragen. Oder der Benutzer will eine *globale* Edition ausführen. Darunter sollen hier Aktionen verstanden werden, die gleichzeitig mehrere Plankomponenten betreffen können, unter denen sich Kantenstücke befinden können. Diese Editionsschritte fallen in den Zuständigkeitsbereich des Planeditors. Es handelt sich dabei um Aktionen wie z.B. Ausschneiden, Bewegen oder Kopieren. Diese Schritte werden im folgenden Kapitel detailliert behandelt. Hier sei zunächst nur von Interesse, daß der Planeditor vor dem eigentlichen Editionsschritt ein Ausgliedern der betroffenen Plankomponenten beim Kantenstückbaumakteur in Auftrag gibt.

26.. Der Kantenstückbaumakteur wird im Editor–System als einziges Exemplar der eigens dafür eingeführten *spk–tree–manager–class* realisiert. Um den Zugriff auf den Kantenstückbaumakteur innerhalb des SPIKES–Systems zu ermöglichen, wird in der globalen Variable *spk–tree–manager* ein Verweis auf dieses Objekt hinterlegt.

27.. Auch wenn im folgenden Nachrichten aus Gründen der Anschaulichkeit scheinbar direkt zwischen den Operationsakteuren unterschiedlicher Akteure verschickt werden, treten in der Implementierung nur der Planeditor, der Kantenstückbaumakteur und die Kantenstückakteure namentlich in Erscheinung. Bei den Operationsakteuren handelt es sich nicht um eigenständige Objekte im Sinne der objektorientierten Programmierung, sondern um einzelne oder mehrere zu einer semantisch zusammenhängen Gruppe zusammengefaßter Methoden besagter drei Akteure.

Zuständig für diesen Ausgliederungsprozeß ist innerhalb des Kantenstückbaumakteurs der *Ausgliederungsakteur.* Dieser enthält seinerseits zwei weitere Komponenten.

Der Trennungsakteur ist für das Aufheben aller Referenzen zwischen zu lösenden und zurückbleibenden Kantenstücken innerhalb eines Baumes zuständig. Er wird dazu mit dem *Baumrelationsverwaltern* der betroffenen Kantenstücke kommunizieren müssen, welche dazu dienen, den Stamm bzw. die Äste eines Kantenstückes zu ermitteln oder neu festzulegen. Darüber hinaus kann man von dem Baumrelationsverwaltern alle Äste des Teilbaumes ab dem betreffenden Kantenstück oder des gesamten Baumes, sowie die Wurzel des Baumes erfragen. Der Baumrelationsverwalter ist der einzige der aufgeführten Operationsakteure eines Kantenstückes, dessen Aufgaben nicht vom Kantenstücktyp abhängen.

Nach dem ersten Ausgliederungsschritt werden ausgehend von den Wurzeln alle zu lösenden und betroffenen zurückbleibenden Bäume falls nötig vom Anpassungsakteur des Ausgliederungsakteurs wieder in eine Minimalform überführt. Ist es in diesem Zusammenhang erforderlich, ein Stammkantenstück mit seinem einzigen Astkantenstück zu verschmelzen, übergibt der Anpassungsakteur dem Kontaktlistenverwalter des Kantenstückbaumakteurs eine vollständige Wurzelkontaktbeschreibung. Dieser trägt die Beschreibung in den Speicher für alle erfaßten Kontaktbeschreibungen ein. Danach beauftragt der Anpassungsakteur den Eingliederungsakteur für einzelne Bäume zur Vereinigung der beiden im Kontakt beschriebenen Kantenstücke. Von den beiden betroffenen Kantenstückakteuren erfragt der Eingliederungsakteur die aktuellen Punkt– und Segmentlisten und trägt eine Kopie dieser Listen in seinem lokalen Speicher ein. Basierend auf diesen Listen setzt sich, wie zuvor beschrieben, der Eingliederungsprozeß fort.

Am Ende des Vereinigungsprozesses eines Stammkantenstückes mit seinem einzigen Astkantenstück wird dem Stammkantenstück vom Eingliederungsakteur der neue verlängerte Linienzug inklusive Punktliste übergeben. Der Kantenstückakteur paßt den Linienzug basierend auf diesen Listen selbstständig an und vermerkt die neue Punktliste. Anschließend muß der verlängerte Stamm vom zu löschenden Ast dessen Attribute übernehmen. Dazu wird der *Attributübernahmeakteur* des Stammkantenstückes vom Eingliederungsakteur aufgefordert. Dem kantenstücktypspezifischen Attributübernahmeakteur wird dazu ein Verweis auf das ursprüngliche Aststück übergeben. Dieses fordert er dann zur Übergabe der Attribute auf. Nach Anpassen der neuen Ast–Stamm–Referenzen und Löschen des ehemaligen Astkantenstückes fordert der Eingliederungsakteur für einzelne Bäume den Kontaktlistenverwalter auf, den bearbeiteten Kontakt zu löschen und die verbleibenden Kontaktbeschreibungen anzupassen. Da es in diesem Falle keine weiteren Kontaktbeschreibungen gibt, wird dem Anpassungsakteur die Vereinigung als abgeschlossen gemeldet. Um die Attribute des neuen (alten) Stammkantenstückes entsprechend der Lage im Baum anzupassen, wird abschließend der *Anpassungsakteur für baumabhängige Attribute* des Kantenstückes angestoßen. Ist der Ausgliederungsprozeß abgeschlossen, wird dies dem Auftraggeber, das heißt, dem zu bearbeitenden Kantenstück oder dem Planeditor, mitgeteilt.

Nach dem Ausgliedern und dem eigentlichen Editionsschritt erfolgt dann in der Regel mit Ausnahme des Löschens von Plankomponenten der Wiedereingliederungsprozeß. Hierzu wird dem *übergeordneten Eingliederungsakteur* eine Liste mit Verweisen auf alle einzugliedernden Plankomponenten übergeben. Baum für Baum werden alle in dieser Menge aufgeführten, zu einem Kantenstückbaum gehörenden Kantenstücke ermittelt. Darauf aufbauend gliedert der *Eingliederungsakteur für einzelne Bäume* diese nacheinander in den Plan ein. Auf das in Bild–A12 gezeigte Petrinetz abgebildet, bedeutet dies, daß der übergeordnete Eingliederungsakteur für die Abwicklung des gesamten dunkelgrau hinterlegten Ablaufes zuständig ist, wobei der Eingliederungsakteur für einzelne Bäume den hellgrau hinterlegten Aufgabenteil übernimmt.

Zum Ermitteln aller vorkommenden Wurzelkontakte wird in einem ersten Schritt der Wurzelkontaktsucher eines jeden Kantenstückes aufgefordert die berührenden Kantenstückes gleichen Typs des ursprünglichen Planes zu ermitteln. Zwischen den ermittelten berührenden Kantenstükken und den einzugliedernden Kantenstücken werden nun die Wurzelkontakte ermittelt, indem man wiederum die Wurzelkontaktsucher aller vorkommenden Kantenstücke auffordert, alle Kontakte des oder der Wurzelpunkte zur Menge der potentiellen Kontaktpartner zu ermitteln. Findet der Wurzelkontaktsucher einen Kontakt, legt er eine Kontaktbeschreibung an und übermittelt diese dem Kontaktlistenverwalter des Kantenstückbaumakteurs zum Eintragen in die Kontaktbeschreibungsliste. Bevor – basierend auf einem Kontakt – eine Vereinigung der beiden beteiligten Kantenstücke erfolgen kann, wird der *Verträglichkeitstester* des berührenden Kantenstückes die betreffende Kontaktbeschreibung untersuchen, ob es sich um einen erlaubten Wurzelkontakt handelt. Dabei erweitert der Verträglichkeitstester die Kontaktbeschreibung um die Verweise auf die beiden beteiligten Segmente im Kontaktpunkt. Das Ergebnis der Untersuchung wird an den Eingliederungsakteur für einzelne Bäume zurückgeschickt, der bei positivem Ergebnis daraufhin die Eingliederung fortsetzt oder sie bei negativem Ergebnis abbricht und zur nächsten Kontaktbeschreibung übergeht. Ist die Eingliederung aller Kantenstücke abgeschlossen, erfolgt auch hier eine Rückmeldung an den Auftraggeber. In der Regel ist der Editionsschritt hiermit beendet. Die eigentliche Beschreibung des Kantenstückbaumakteurs ist hiermit abgeschlossen.

Bislang unerwähnt geblieben ist der Auftragskanal des Kantenstückbaumakteurs zum Planeditor zum Eintragen der Undoinformation für zu ändernde Kantenstücke. Jedesmal bevor ein Kantenstück geändert wird, wird dem Planeditor die nötige Information zum Rückgängigmachen dieses Schrittes mitgeteilt. Wählt der Benutzer im nächsten Editionsschritt den Menüeintrag ”Aufheben” aus, wird der Undoakteur beauftragt, basierend auf der Undoinformation des vorausgehenden Schrittes den letzten Editionsschritt rückgängig zu machen. Die damit verbundenen Schritte werden im folgenden Kapitel geschildert.

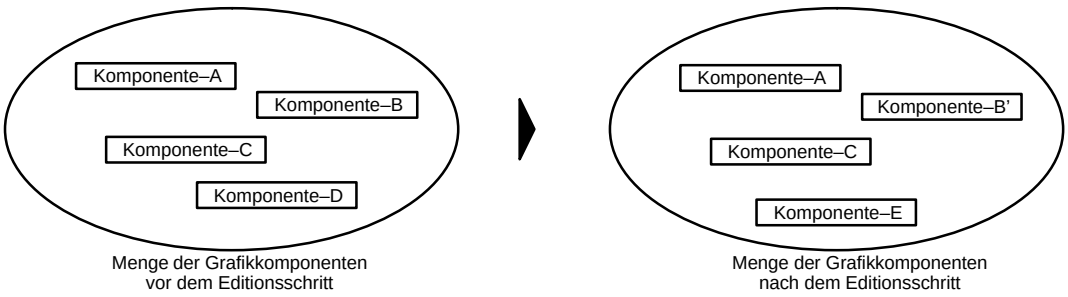
1.4 Symbolunabhängige Editionsschritte

In diesem Kapitel werden die symbolunabhängigen oder auch globalen Editionsschritte beschrieben. Dies sind solche Schritte, die gleichermaßen auf alle Komponenten des Planes und mit Einschränkungen auch auf Symbolkomponenten anwendbar sind. Es handelt sich dabei um das Ausschneiden und Wiedereinfügen, das Bewegen und das Kopieren einer oder mehrerer Komponenten der aktuellen Grafikgruppe, sowie um das Aufheben von Editionsschritten (Undo). Zuständig für diese Aufgaben, die im folgenden betrachtet werden, ist der *Planeditor.* Weitere Aufgaben des Planeditors bestehen darin, alle Plankomponenten eines bestimmten Typs aus– oder abzuwählen oder Plankomponenten in freie Grafik des Planes umzuwandeln.

1.4.1 Das Undo–Konzept

Bei SPIKES–Editoren ist ein *hierarchisches Toggle–Undo* vorgesehen. Das bedeutet, daß es möglich sein soll, zwischen dem Zustand des SPIKES–Planes vor und nach dem jeweils letzten Editionsschritt hin und herzuwechseln. Entsprechend dem hierarchischen Bearbeiten ineinandergeschachtelter Untergruppen soll es dabei möglich sein, innerhalb einer Untergruppe beliebiger Ebene jeweils den letzten Editionsschritt wiederaufzuheben. Nach Schließen der Untergruppe, soll es dann möglich sein, alle Änderungen, die während dieser Untergruppenbearbeitung am Inhalt der Untergruppe vorgenommen wurden, in einem Schritt aufzuheben.

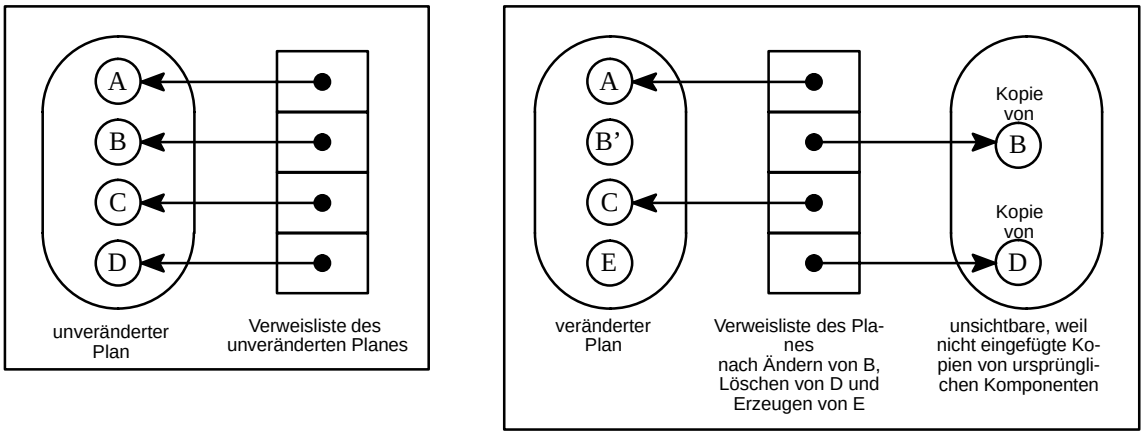
Zum Aufheben des letzten Editionsschrittes muß in irgendeiner Art und Weise die Information hinterlegt sein, wie man den Zustand des Planes vor diesem Editionszustand wieder herstellen kann. Da man von einem einfachen einschrittigen Undo zu einem hierarchichen Undo einfach durch Einführen eines Stapels, auf dem zuoberst jeweils die Undoinformation der aktuellen Untergruppe liegt, gelangen kann, soll zunächst das Hauptaugenmerk auf der Realisierung eines einschrittigen Undo liegen. Was dabei passiert, soll anhand Bild–30 erläutert werden.



Bild–30: Beispiel einer Menge von Grafikobjekten vor und nach einem Editionsschritt

Aus einer Menge von Grafikkomponenten vor dem Editionsschritt wird eine andere Menge von

Grafikkomponenten nach dem Editionsschritt. Denkt man an das Bewegen von Kantenstückbäumen, so können sich allein durch das Ausgliedern der Kantenstücke an einer Stelle und das anschließende Wiedereingliedern an anderer Stelle beide Menge gewaltig voneinander unterscheiden. Es gibt Komponenten, die unverändert in beiden Mengen auftauchen (A, C), es gibt Komponenten, die nur in einer Menge auftauchen (D, E) und es gibt Komponenten die in beiden Plänen auftauchen, aber verändert wurden (B/B'). Um einen Editionsschritt aufzuheben, muß man die neu entstandenen Komponenten (E) wieder löschen, die verschwundenen Komponenten (D) wieder einsetzen und die geänderten Komponenten (B') wieder durch die unveränderten Originale (B) ersetzen. In umgekehrter Reihenfolge kann dieses Undo selbst wieder aufgehoben werden. Im Beispiel stellt dies keinerlei Problem dar, da man beide Pläne direkt nebeneinander vorliegen hat. Man weiß, wie der Plan vor dem letzten Editionsschritt ausgesehen hat. Es ist sicherlich eine Speicherverschwendung, eine Kopie des gesamten Planes zu machen. (In diesem Fall wäre es sicherlich günstiger, die ganzen Pläne einfach gegeneinander auszutauschen, anstatt nach Unterschieden zu suchen.) Anstatt den ganzen Plan zu kopieren, kann man eine Liste mit Verweisen auf alle Komponenten anlegen, die vor dem Editionsschritt im Plan oder in der bearbeiteten Grafikgruppe existieren.



Bild–31: Zugriff auf die Undo–Information mittels Verweisliste

Wurden während des Editionsschrittes neue Komponenten erzeugt, so erkennt man bei einem Vergleich der Liste mit dem Plan oder der Untergruppe, daß auf diese in der Liste keine Verweise enthalten sind. Diese müssen also bei einem Undo gelöscht werden.

Werden Komponenten geändert, so ist der Verweis hiervon nicht betroffen und eine Änderung kann somit auf diesem Wege gar nicht erkannt werden und selbst wenn dies möglich wäre, stünde die Information zum Zurücknehmen einer solchen Änderung nicht zur Verfügung. Dasselbe gilt in ähnlicher Form für Komponenten, die gelöscht wurden – auch hier fehlt die Information zum Wiederherstellen des ursprünglichen Zustandes.

Um bei einem Undo über die nötige Information zu verfügen, legt man daher vor dem Editieren einer ursprünglichen unveränderten Komponente eine Kopie dieser Komponente an, die jedoch

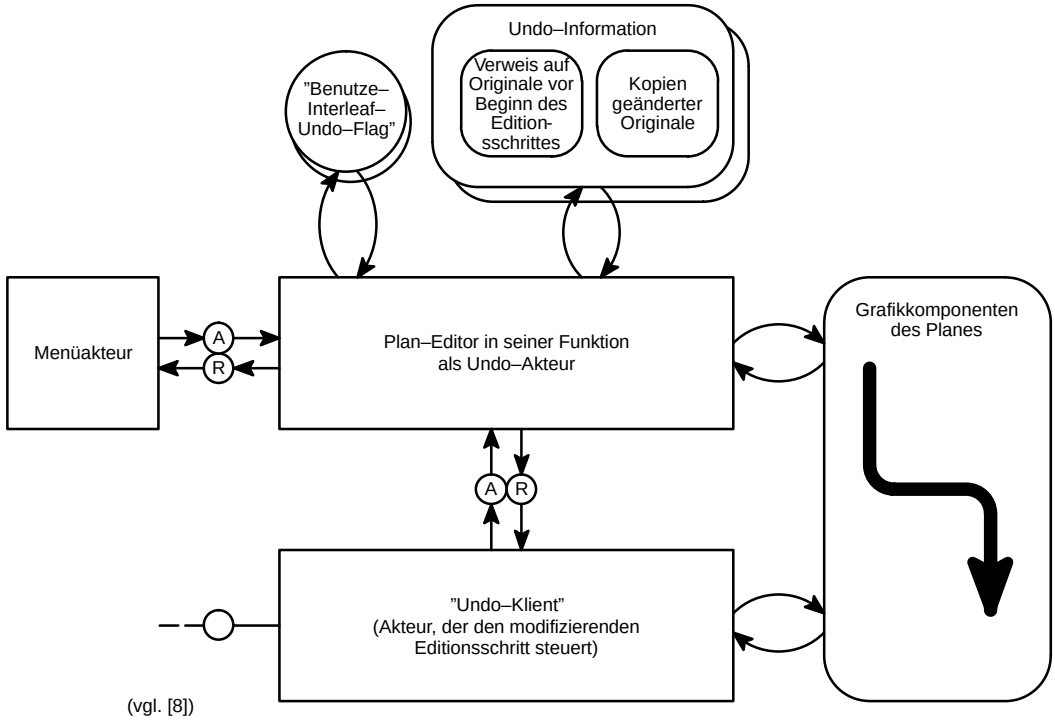
nicht in den Plan eingefügt wird und daher unsichtbar ist. Mittels dieser Kopie ist es möglich, den ursprünglichen Zustand wiederherzustellen.

Bei der Edition einer Grafikkomponente geht man daher wie folgt vor: Man überprüft vor dem Löschen oder Ändern einer Komponente, ob sich ein Verweis auf die betroffene Komponente in der Undo–Verweisliste befindet. Ist dies der Fall, handelt es sich um eine ursprüngliche Komponente, und man legt von dieser eine unsichtbare, das heißt nicht im Plan eingefügte, Kopie an und vermerkt einen Verweis auf diese Kopie in der Liste. Außerdem entfernt man den Verweis auf die ursprüngliche Komponente aus der Liste. Diese beiden Schritte sind gleichbedeutend mit dem Ersetzen der ursprünglichen Komponente durch die Kopie und einem anschließenden Heraustrennen der Kopie aus dem Plan. Die tatsächliche ursprüngliche Komponente ist nicht mehr in der Undoliste vermerkt und kann nun gelöscht oder bearbeitet werden. Im zweiten Fall erscheint sie gleichsam als neue Komponente. Damit werden auch alle eventuellen weiteren Änderungen innerhalb dieses Editionsschrittes an dieser Komponente nicht mehr in der Undo–Verweisliste eingetragen. Beim Aufheben eines solchen Editionsschrittes stellt man fest, daß der nachträglich angelegte Verweis eine nicht im Plan enthaltene Komponente bezeichnet. Diese ist demzufolge wieder in den Plan einzufügen, was jetzt möglich ist, da es sich um die bislang unsichtbare Kopie der unveränderten ursprünglichen Komponente handelt. Findet man die veränderte ursprüngliche Komponente im Plan, jedoch keinen Verweis auf diese in der Liste, wird die demzufolge als neu erscheinende Komponente entfernt.

Werden Komponenten verändert oder gelöscht, auf die kein Verweis in der Liste eingetragen ist, kann es sich nicht um Komponenten handeln, die ursprüngliche Grafiken verkörpern. In diesen Fällen muß natürlich keine Kopie zum Erhalt der ursprünglichen Grafik angelegt werden.

Bild–A23 zeigt den zusammenhängenden Ablauf vom Beginn bis zum Aufheben eines Editionsschrittes. Um die semantische Zusammengehörigkeit der Schrittfolge Undo–Information Initialisieren, Editionsschritt und Undo–Schritt klarer darzustellen, wurde bewußt darauf verzichtet Editionsschritt und Undo–Schritt nebeneinander als Editionsschritte auf gleichem Niveau darzustellen zwischen denen der Benutzer bei jedem Schritt die frei Wahl hat. Aus demselben Grund wurde ebenfalls die explizite Darstellung des hierarchischen Undo ausgeklammert. In Bild–32 ist gezeigt, wie sich das Undo–Konzept auf den Aufbau des SPIKES–Editor–System auswirkt. Auf einige implementierungsabhängige Besonderheiten in diesem Aufbau wird im Anschluß eingegangen.

Ein Editionsschritt wird damit eingeleitet, daß der Benutzer eine Aktion aus einem Menü auswählt. In den Fällen, in denen es sich um eine Aktion zur Veränderung der Menge von Plankomponenten handelt, ausgenommen dem Aufheben des letzten Editionsschrittes, muß eine neue Undo–Verweisliste erstellt werden. Da nur ein Toggle–Undo unterstützt wird, wird die alte Verweisliste hinfällig und kann gelöscht werden. Das gleiche gilt für die nicht benötigten, nicht eingefügten Kopien von ursprünglichen Grafikkomponenten, die seit dem letzten Editionsschritt verändert wurden. Verantworlich für diese Aktionen ist der Plan–Editor in seiner Funktion als



Bild–32: Der SPIKES–Undo–Akteur

Undo–Akteur. Dieser wird dazu noch vom Menüakteur beauftragt[28.]. Danach stößt der Auftragszusteller (*menu–handler*) den Editionsakteur für den entsprechenden Editionsschritt an.

Vor jeder Änderung von Plankomponenten, wozu auch das Löschen zählt, wird dem Undo–Akteur von dem für den Editionsschritt zuständigen Akteur, dem "Undo–Klient", die Änderung der betroffenen Komponenten mitgeteilt. Denkbare Undo–Klienten sind zum Beispiel der Kantenstückbaumakteur oder die diversen Editionsakteure im SPIKES–Editorsystem. Ist anhand der Undo–Verweisliste erkennbar, daß es sich nicht um eine unveränderte ursprüngliche Komponente handeln kann, muß nichts geschehen. Ansonsten ersetzt, wie zuvor beschrieben, ein Verweis auf eine nicht eingefügte Kopie von dem unveränderten Original das Original in der Undo–Information. Für den Editionsakteur bleibt der damit verbundene Interpretationswechsel des Originals zur neuen Plankomponente verborgen. Dies ist der Grund, warum die Kopie die Rolle des Originals übernimmt und nicht, was ebenso möglich wäre, das Original aus dem Plan gelöst wird und stattdessen die Kopie eingefügt und bearbeitet wird. Handelt es sich bei dem Editionsschritt um das mehrschrittige Bearbeiten einer Grafikgruppe, so wird zunächst die gesamte Grafikgruppe als zu ändernd vermerkt. Danach wird mit dem Öffnen der Grafikgruppe eine neue Undo–Verweisliste angelegt, die die alte Liste nicht ersetzt, sondern nur für die Bearbeitungsdauer der Untergruppe überdeckt. Das bedeutet, daß man einen Stapel benötigt, auf dem die Undo–Verweislisten aller in Bearbeitung befindlichen hierarchisch ineinander enthaltenen Untergruppen abgelegt sind. Zuoberst befindet sich auf diesem Stapel immer die Undo–Informa-

28.. In der Implementierung ist dieser Teil verdeckt in der Funktion *popup–run–with–control* realisiert. In den die Menüakteure beschreibenden *Editor*modulen ist diese Funktionalität nicht offen erkennbar.

tion, die nötig ist, um den letzten Editionsschritt bezüglich der aktuellen Gruppenebene aufzuheben. Wird eine Grafikgruppe geschlossen, wird die oberste Verweisliste vom Stapel entfernt, so daß die Liste wieder relevant wird, die einen Verweis auf die Kopie der gesamten unveränderten Gruppe enthält. In dem gezeigten Petrinetz erscheint eine solche Untergruppenbearbeitung als ein einziger Schritt. Die Editionsschritte und die damit verbundenen Undoaktionen innerhalb einer Untergruppenbearbeitung sind nicht gezeigt. Sie entsprechen jedoch innerhalb ihrer Ebene wiederum genau dem dargestellten Ablauf (mit der selben Einschränkung für eine weitere Untergruppenbearbeitung).

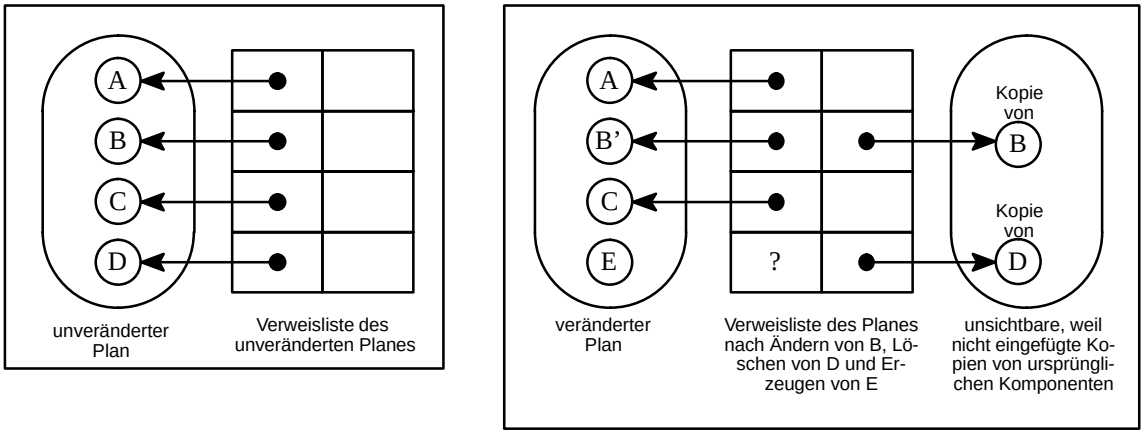
Ist ein Editionsschritt abgeschlossen, so kann der Benutzer diesen durch Auswählen des Punktes ”Aufheben” aus dem Menü rückgängig machen. Der Menüakteur beauftragt dazu den Undo–Akteur. Dieser hat dabei zwei Aufgaben zu erfüllen. Zum einen muß er natürlich den letzten Editionsschritt aufheben, zum anderen soll – entsprechend der Definition des geforderten Toggle–Undo – dieser Undo–Schritt selbst aufhebbar sein. Dazu muß er außerdem eine neue Undo–Verweisliste erstellen. Geht man von einer Verweisliste entsprechend Bild–31 aus, bleiben die Komponenten des Planes, auf die Verweise existieren, unverändert. Die Verweise werden in die neue Liste übernommen. Die Komponenten des Planes, für die keine Verweise existieren, werden aus dem Plan herausgelöst und unsichtbar abgelegt. In der neuen Verweisliste werden Verweise auf diese Komponenten vermerkt. Die zu diesem Zeitpunkt nicht eingefügten Komponenten, die ein Abbild der unveränderten ursprünglichen Komponenten vor ihrer Bearbeitung darstellen, werden in den Plan eingefügt. Da diese bezogen auf den Undoschritt als neu anzusehen sind, wird auf diese kein Verweis in der neuen Undoliste hinterlegt. Es handelt sich somit sozusagen um einen Austausch der unsichtbar hinterlegten Komponenten gegen die im vorangegangenen Editionsschritt neu hinzugekommenen Komponenten. In diesem Zusammenhang müssen allerdings die Referenzen zwischen den einzelnen Komponenten berücksichtigt werden. Ersetzt eine Kopie eine ursprüngliche Komponente, müssen in der Übersetzungstabelle des Kennungsverwalters (vgl. Kapitel 1.1.3) die zu den Schlüsseln vermerkten Laufzeit–ID’s der ursprünglichen Komponente und deren Unterkomponenten gegen die der Kopie und deren Unterkomponenten ausgetauscht werden. Auf diesem Wege werden Falschverweise durch mehrfach benutzte Schlüssel vermieden (vgl. Kapitel 1.4.2.3).

Besonderheiten in der Implementierung:

Das oben beschriebene Undo–Konzept entstand aus der Notwendigkeit, ein ’Undo’ für Editionsschritte zu entwickeln, die mehr als einen Editionsauftrag umfassen, wie dies zum Beispiel beim Eingliedern von Kantenstücken der Fall ist. In diesen Fällen reicht die von Interleaf–V angebotene Undo–Funktionalität, mit der lediglich das Zurücknehmen der Bearbeitung einer einzelnen Grafikkomponente möglich ist, nicht aus. Es gibt aber auch viele Fälle, in denen es möglich ist, auf die angebotene Interleaf–Funktionalität zurückzugreifen. In diesen Fällen ist das Interleaf–Undo dem SPIKES–Undo aus Performance–Gründen vorzuziehen.

Um je nach Fall das geeignete Undo–Verfahren anwenden zu können, wurde ein zusätzliches Flag eingeführt. Dieses wird vom Menüakteur bei allen Editionsschritten vor dem Anstoßen des Editionsakteurs entsprechend gesetzt oder gelöscht, je nachdem ob ein Interleaf–Undo vorgesehen ist oder nicht. Wird dem Undo–Akteur bei gesetztem Flag ein Auftrag zum Aufheben des letzten Editionsschrittes geschickt, wird das von Interleaf–V angebotenen Undo verwendet. Wie für die Undo–Verweisliste ist auch für dieses Flag ein Stapel vorgesehen, um das Arbeiten mit verschiedenen Grafikgruppenebenen zu ermöglichen.

Anstatt das Undo–Verfahren mit Hilfe der oben beschrieben Undo–Verweisliste zu realisieren, wurde Performance–Gründen, die Undo–Information in leicht veränderter Form zu hinterlegt. Das zugrunde liegende Prinzip wurde hierdurch nicht berührt, jedoch sollen kurz die sich daraus resultierenden Änderungen dargestellt werden. Als Beispiel in Bild–33 dient noch einmal dieselbe Situation, wie in Bild–31. Anstatt der Verweisliste verwendet man eine Übersetzungstabelle. Diese Tabelle besteht aus zwei Spalten, von denen die Einträge der ersten Spalte die Schlüssel zum Auffinden und Lesen der entsprechenden Einträge der zweiten Spalte darstellen[29].



Bild–33: Implementierung der Verweisliste

Beim anfänglichen Erstellen der Undo–Information zu Beginn eines Editionsschrittes trägt man in der erste Spalte die Verweise auf alle Komponenten ein, die innerhalb der aktuellen Grafikgruppe enthalten sind. Die zweite Spalte zunächst bleibt leer. Sie dient dazu, im Falle einer Änderung einer ursprünglichen unveränderten Komponente einen Verweis auf die unsichtbare Kopie aufzunehmen. Wird eine Komponente zum Ändern gemeldet, muß auch jetzt nur etwas geschehen, wenn es sich um eine unveränderte ursprüngliche Komponente handelt. Während man dies bei der einfachen Liste daran erkennt, daß ein Verweis auf diese Komponente in der Liste vorkommt, erkennt man dies im Falle der Übersetzungstabelle daran, daß der Schlüssel, also ein Element der ersten Spalte, gleichzeitig auf eine Komponente des Planes verweist und kein Element der zweiten Spalte referenziert. Ist dies erfüllt, wird eine Kopie der Komponente erzeugt und der Verweis auf diese unter dem besagten Schlüssel in der Tabelle in der zweiten Spalte abgelegt. Das Verfahren mit der einfachen Liste muß in diesem Punkte langsamer sein, da hier zwei

29.. vgl. *hashtables* bei Interleaf–V

Schritte nötig sind. Zum einen muß der ursprüngliche Verweis aus der Liste entfernt werden und zum anderen muß auch hier der Verweis auf die Kopie vermerkt werden. In beiden Verfahren übernimmt die Kopie die Rolle der ursprünglichen unveränderten Komponente, wonach die eigentliche ursprüngliche Komponente beliebig verändert oder gelöscht werden kann. In letzterem Fall, zeigt der in der ersten Spalte eingetragene Verweis auf ein nicht mehr existierendes Grafikobjekt. In Interleaf–V erkennt man dies daran, daß man beim Lesen eines solchen Verweises nicht mehr den ursprünglichen Wert, sondern statt dessen einen nur für diesen Fall definierten 'Nullpointer' erhält. Dies ist jedoch unschädlich, da die direkte Zuordnung einer Kopie zu der im Plan verbleibenden Komponente nicht benötigt wird und es die Möglichkeit gibt, alle Einträge der Tabelle zu erfragen ohne über den Schlüssel zugreifen zu müssen. Werden neue Komponenten angelegt, oder solche Komponenten verändert, deren Verweis keinen leeren Eintrag in der Tabelle referenziert, muß nichts geschehen. Beim Aufheben eines Editionsschrittes gelten dieselben Überlegungen, wie bei der einfachen Liste. Hier ergibt sich kein Gewinn durch die Verwendung einer Übersetzungstabelle.

1.4.2 Ausschneiden, Kopieren und Einfügen

Aufgrund der engen Verwandtschaft der Editionsschritte Ausschneiden, Kopieren und Einfügen zueinander ist es zweckmäßig, die Funktionsweise dieser globalen Editionsschritte, sowie die Schritte *Bewegen* und *Duplizieren&Bewegen* nebeneinander zu betrachten. In Bild–A24 ist der Ablauf der jeweiligen Schritte gezeigt. Faßt man Ausschneiden und anschließendes Wiedereinfügen von Grafikkomponenten in denselben Plan zu einem Schritt zusammen, so entspricht dies gleichsam dem Bewegen von Grafikkomponenten. Ebenso kann man das Kopieren und unmittelbare Wiedereinfügen von Grafikkomponenten als Duplizieren&Bewegen interpretieren. Demzufolge reicht es, im folgenden nur das Ausschneiden, Kopieren und Einfügen zu untersuchen. Auf die Besonderheiten des Bewegens mit Kontakt soll an dieser Stelle nicht eingegangen werden.

1.4.2.1 Ausschneiden und Einfügen[30.]:

Bevor das eigentliche Ausschneiden erfolgt, muß der Planeditor dafür sorgen, daß die vorzunehmenden Änderungen bei einer eventuell nachfolgenden UNDO–Operation wieder rückgängig gemacht werden können. Da der Planeditor auch für die UNDO–Funktionalität zuständig ist, muß er nur einen Auftrag an seinen entsprechenden Operationsakteur geben.

Als nächstes muß der Planeditor dafür sorgen, daß alle Verweise zwischen auszuschneidenden und im Plan verbleibenden Grafikkomponenten gelöscht werden. Das Aktualisieren der Kantenstückbaumstruktur übernimmt der Kantenstückbaumakteur. Dabei werden die Kantenstückbäume, die durch die auszuschneidenden Kantenstücke definiert sind, aus den bestehenden Kan-

30.. Dieser Abschnitt, "Ausschneiden und Einfügen", entspricht in wesentlichen Teilen einem Dokument von W. Kleis (vgl. [8]), auf den Konzept und Implementierung des Cut&Paste–Akteurs zurückgehen.

tenstückbäumen ausgegliedert. Hierdurch kann sich möglicherweise die Menge der auszuschneidenden Grafikkomponenten ändern.

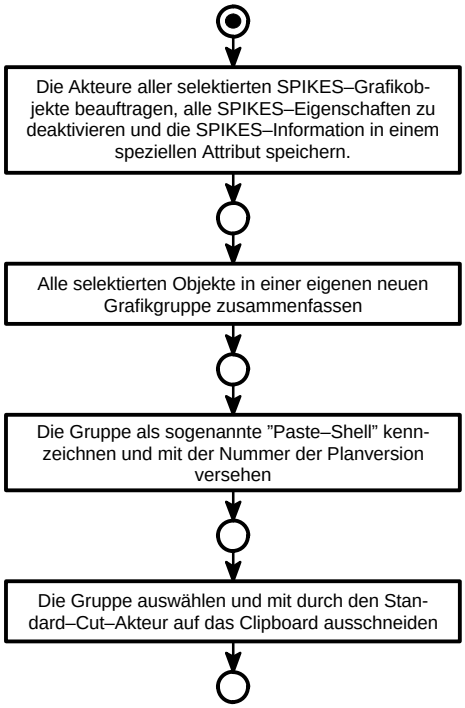
Nachdem diese Vorbereitungen abgeschlossen sind, beauftragt der Planeditor den Cut+Paste–Akteur das eigentliche Ausschneiden durchzuführen (*do–cut*). Die Einbindung dieses Akteurs in das Editorsystem, der im bisherigen Aufbaumodell als Teil des Plan–Editors angesehen werden kann, ist in Bild–A25 gezeigt. Dieser spezielle Cut+Paste–Akteur wurde notwendig, weil INTERLEAF für das Ausschneiden und Einfügen eine einzige globale dokumentenübergreifende Zwischenablage benutzt. Dadurch wird es möglich, Grafikobjekte, die aus einem Plan eines bestimmten Typs ausgeschnitten wurden, in einen Standardgrafikrahmen, einen Plan eines anderen Typs oder in einen Plan einer falschen Version einzufügen. Die ausgeschnittenen Objekte müssen deshalb mit der Plantypbezeichnung und der Nummer der Editorversion gekennzeichnet werden. Beim Einfügen man kann auf Grund dieser Kennzeichnung erkennen, ob die Objekte als SPIKES–Objekte eingefügt werden dürfen. Wenn die Objekte in einen Standardgrafikrahmen eingefügt werden, wird diese Aktion komplett vom Paste–Akteur des INTERLEAF–Basissystems durchgeführt. Dieser weiß aber nicht, daß es SPIKES–Objekte gibt und daß die aus SPIKES–Rahmen ausgeschnittenen Objekte spezielle Kennzeichnungen tragen. Er fügt die Objekte einfach ein. Aus diesem Grund müssen die Objekte als Standardgrafikobjekte ausgeschnitten und damit auf der Zwischenablage abgelegt werden, um sie so später problemlos in Standard–Grafikrahmen einfügen zu können. Die SPIKES–Eigenschaften der Objekte werden deshalb vor dem Ausschneiden der Objekte deaktiviert. Die Einträge in der Zwischenablage sind nur noch mit einem Stempel versehen, der einem Cut+Paste–Akteur eines SPIKES–Planes sagt, daß es sich um ein deaktiviertes SPIKES–Objekt handelt. Der Stempel wird so implementiert, daß er nicht stört, wenn man das Objekt in einen Standardrahmen einfügt.

Um Aufwand zu sparen, werden Planversion und der Plantyp nicht bei jedem Objekt vermerkt, sondern alle auszuschneidenden SPIKES–Objekte werden vor dem Ausschneiden in einer Grafikgruppe zusammengefaßt, so daß nur diese Gruppe mit der Planversion und der Plantypbezeichnung gestempelt werden muß. Diese Gruppen, die nur zur Implementierung eines korrekten Einfügens ausgeschnittener SPIKES–Objekte gebraucht werden, werden im SPIKES–Jargon ”*Paste–Shells*” genannt.

Nach dem Einfügen einer als Paste–Shell gekennzeichneten Gruppe in einen Standardgrafikrahmen bleibt die Gruppe mitsamt des Stempels erhalten. Der Benutzer kann sie wieder ausschneiden und die enthaltenen Objekte später als SPIKES–Objekte in einen SPIKES–Rahmen einfügen. Wenn der Benutzer aber die Gruppe auflöst oder deren Inhalt bearbeitet, geht der Stempel

verloren[31.]. Wenn die betroffenen Objekte wieder ausgeschnitten werden, können sie nun nur noch als reine Grafikobjekte in einen SPIKES–Rahmen eingefügt werden. Dies ist sinnvoll, da der Benutzer während der Bearbeitung in einem Standardgrafikrahmen die Grafikgruppenstruktur deaktivierter SPIKES–Objekte unbemerkt ändern könnte. Wenn die Objekte wieder in einen SPIKES–Plan eingefügt und wieder reaktiviert werden, könnten diese Manipulationen zu Fehlern führen.

In Bild–34 ist gezeigt, wie der Cut+Paste–Akteur auf einen Ausschneideauftrag reagiert.



Bild–34: Ausschneiden von SPIKES–Grafikobjekten

Zunächst beauftragt er alle betroffenen Grafikakteure, sich auf das Ausschneiden vorzubereiten (*mid:prepare–cut*). Dies bedeutet für diese Akteure, daß sie sich aus der ID–Übersetzungstabelle des Kennungsverwalters austragen lassen und ihre SPIKES–Eigenschaften deaktivieren. Dazu wird die Klassenzugehörigkeit des SPIKES–Objektes vor dem Ausschneiden beim dem Objekt vermerkt und danach die Klassenzugehörigkeit auf die nächste Oberklasse gewechselt, die eine Interleaf–Basisklasse darstellt. Da die betroffenen Grafikkomponenten in der Regel selbst weitere SPIKES–Grafikkomponenten enthalten, wird dieser Auftrag rekursiv an alle Grafikakteure innerhalb der gesamten Hierarchie aller ausgewählten Komponenten geschickt. Als nächstes werden die auszuschneidenden Objekte in einer Paste–Shell gruppiert. Die neu entstandene

31.. Die Kennzeichnung als Paste–Shell wird mit Hilfe der *saved–data*–Attributs für applikationspezifische Objektdaten implementiert. Beim Bearbeiten des Inhalts einer Gruppe wird die bearbeitete Gruppe aufgelöst und der zuständige Objektakteur wird zerstört. Beim Schließen der Untergruppenbearbeitung wird ein neuer Grafikgruppenakteur erzeugt. Der applikationsspezifische Gedächtnisinhalt des ursprünglichen Grafikgruppenakteurs geht dabei verloren. Wegen dieser Eigenheit des INTERLEAF–Basissystems geht beim Bearbeiten einer als Paste–Shell gestempelten Gruppe auch dieser Stempel verloren.

Paste–Shell wird mit Versionsnummer und Plantypkennung gestempelt und dann vom Standard–Cut–Akteur auf die Grafikzwischenablage ausgeschnitten.

Das Einfügen des Inhaltes der Zwischenablage wird durch einen Auftrag an den Planeditor eingeleitet (*mid:do–paste*). Daraufhin beauftragt der Planeditor zunächst den Cut+Paste–Akteur, die Objekte aus der Zwischenablage einzufügen und gegebenenfalls deren SPIKES–Eigenschaften zu restaurieren. Dieser Ablauf ist detailliert in Bild–A26 gezeigt. Der Cut+Paste–Akteur beauftragt wiederum den Paste–Akteur des INTERLEAF–Basissystems, das eigentliche Einfügen vorzunehmen. Dabei werden die in der Zwischenablage beschriebenen Objekte erzeugt und in die Planstruktur eingefügt. Der Standard–Paste–Akteur fügt die Objekte so in den Plan ein, daß sie sofort selektiert sind. Wenn man davon ausgeht, daß ein Paste–Auftrag immer nur dann erfolgt, wenn nichts selektiert ist, dann sind nach dem Einfügen nur die neuen Objekte selektiert[32.].

Der Cut+Paste Akteur untersucht nun alle eingefügten Objekte und stellt zunächst einmal fest, ob sie als deaktivierte Paste–Shells gekennzeichnet sind. Wenn dies nicht der Fall ist, handelt es sich entweder um normale Grafikobjekte oder um SPIKES–Objekte, deren Paste–Shell zerstört worden war. In letzterem Fall sind bei diesen Objekten noch die deaktivierten SPIKES–Eigenschaften vermerkt. Da diese Daten nicht mehr benötigt werden, entfernt sie der Cut+Paste Akteur.

Falls es sich bei einem eingefügten Objekt um eine deaktivierte Paste–Shell handelt, wird geprüft, ob die Versionsnummer und der Typ des Herkunftsplanes mit Versionsnummer und Typ des aktuellen Planes verträglich sind. Wenn dies nicht der Fall ist, wird der Benutzer gefragt, wie weiter zu verfahren ist. Er kann sich dafür entscheiden, daß die Objekte entweder irreversibel in reine Grafikobjekte umgewandelt werden oder er kann veranlassen, daß die Objekte unverändert wieder ausgeschnitten werden, weil offenbar ein Irrtum vorlag. Wenn der Benutzer die Objekte in reine Graphik wandeln will, wird die Paste–Shell aufgelöst und die Informationen über die deaktivierten SPIKES–Eigenschaften werden gelöscht. Wenn die eingefügten Objekte aus einem Plan passenden Typs und passender Version stammen, dann wird die Paste–Shell aufgelöst und die SPIKES–Eigenschaften aller enthaltenen Objekte werden reaktiviert. Nachdem dies geschehen ist, müssen die ID–Attribute der eingefügten Objekte neu vergeben und die Werte der Referenzattribute aktualisiert werden. Auf dieses Problem wird im Anschluß näher eingegangen.

Wenn alle eingefügten Objekte in der beschriebenen Weise behandelt worden sind, meldet der Cut–Paste–Akteur dem Planeditor zurück, welche neuen Objekte nun tatsächlich auf der obersten Hierarchieebene des Plans eingefügt wurden. Da wie bei der entsprechenden INTERLEAF–Standardoperation die neu eingefügten Objekte interaktiv positioniert werden, leitet der Planeditor nun das interaktive Bewegen der neuen Objekte ein. Dazu erzeugt er einen Editionsakteur für

32.. Ist etwas selektiert, wenn der Standard–Paste–Akteur angestoßen wird, so wird in Interleaf–V nichts eingefügt und die ursprünglichen Auswahlmenge bleibt unverändert.

animierte Edition und gibt diesem den Auftrag, die Editionsaktion zu starten[33.]. Der Editionsakteur wird, wenn der Benutzer die zu bewegenden Objekte fixiert hat, dem Plan–Editorakteur eine Vollzugsmeldung schicken. Der Planeditor–Akteur wird daraufhin den Editionsakteur wieder zerstören und den Kantenstückbaum–Akteur beauftragen, eventuell eingefügte Kantenstückbäume mit bestehende Kantenstückbäumen zu vereinigen.

1.4.2.2 Kopieren

Vernachlässigt man die Kantenbaumproblematik, so erstellt der Planeditor bei einem Kopierauftrag von allen ausgewählten Grafikkomponenten eine Kopie und fügt sie in den Plan ein. Anschließend beauftragt er den SPIKES–Cut+Paste–Akteur die Kopien wiederum auszuschneiden, wodurch diese in einer Paste–Shell zusammengefaßt in der globalen Zwischenablage abgelegt werden. Der Inhalts des Planes ist somit gleichsam unverändert geblieben, weshalb im Falle des Kopierens dem Undo–Akteur keine Änderungen mitgeteilt werden müssen.

Befinden sich unter den zu kopierenden Grafikkomponenten Kantenstücke, so muß für die dadurch definierten Kantenstückteilbäume erfüllt sein, daß diese Minimalform aufweisen und daß keine Stamm–Ast–Referenzen mit nicht zu kopierenden Kantenstücken bestehen. Ansonsten kann es beim Einfügen in einen anderen oder denselben SPIKES–Plan zu Problemen kommen. Die betroffenen Kantenstücke werden daher zuerst aus der nicht betroffenen Kantenstückbaumstruktur ausgegliedert. Nun erst kann von allen ausgegliederten Kantenstücken, sowie allen übrigen ausgewählten Grafikkomponenten eine Kopie erstellt werden, die anschließend durch Ausschneiden auf der Zwischenablage abgelegt werden. Dabei ist zu beachten, daß sich die Menge der auszugliedernden Kantenstücke durchaus von der Menge der ausgegliederten Kantenstücke unterscheiden kann. Die ausgegliederten ursprünglichen Teilbäume sind wieder in den Plan einzugliedern, um so den ursprünglichen Zustand des Planes wiederherzustellen.

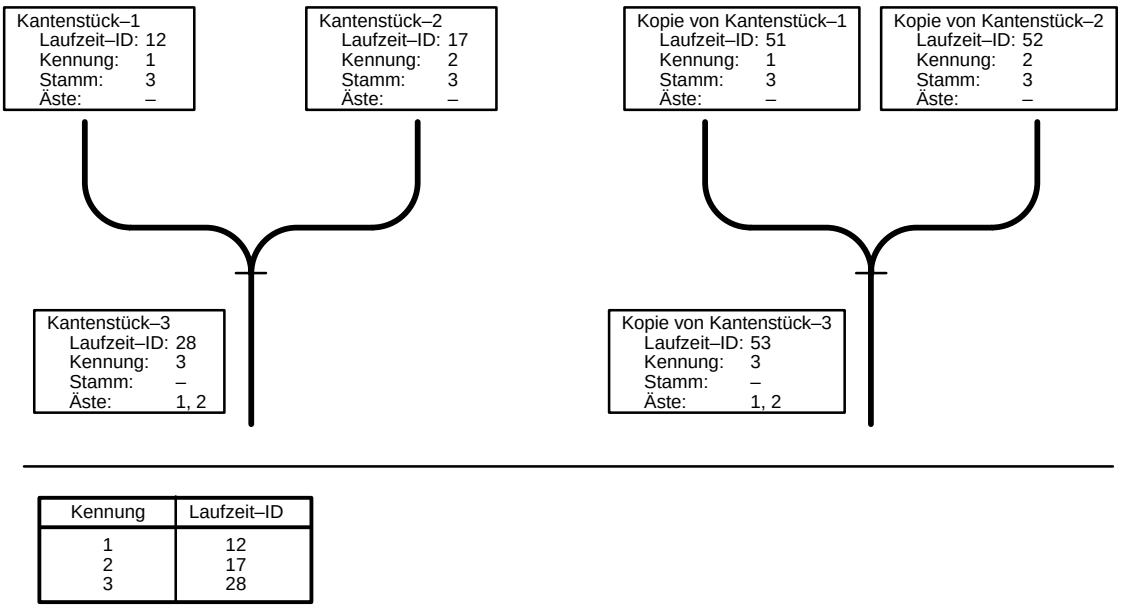
Da das Aus– und Eingliedern von Kantenstückbäumen ein recht aufwendiger Prozeß ist, kann dieses Verfahren in einigen Fällen zu unerwünschten Verzögerungen führen, die man jedoch reduzieren kann. So ist es vom Aufwand betrachtet sicherlich günstiger, den ursprünglichen Zustand wiederherzustellen, indem man dazu den Undo–Akteur beauftragt. Dies ist insofern zulässig, da für den Benutzer ein Aufheben des Kopierens sinnlos ist. Der Undo–Akteur ist bei einem Kopierschritt dadurch sozusagen arbeitslos und steht demzufolge einer anderen Nutzung frei zur Verfügung. Nutzt man dies aus, so reicht es beim Ausgliedern von Kantenstückbäumen nur die auszugliedernden Kantenstücke zu berücksichtigen. Danach werden wie zuvor beschrieben Kopien angefertigt, die selbst wieder korrekte Minimalformkantenbäume beschreiben. Bei allen betroffenen kalten Kantenstückbäumen wird auf das Entfernen von Ast–Stamm–Referenzen, sowie auf das anschließende Anpassen der zurückbleibenden Teilbäume verzichtet. Der

33.. Der Editionsakteur für animierte Edition wird nicht näher im Rahmen dieser Arbeit beschrieben. Er dient dazu, bestimmte Eigenarten von Interleaf–V bei der interaktiven Edition zu verbergen, sowie gezielte Einschränkungen bei interaktiven Editionsschritten vorzugeben.

Aufwand wird dadurch um ein weiteres reduziert, da insbesondere das zeitaufwendige Verschmelzen von Stämmen mit nur einem Ast und das Anpassen aller baumabhängigen Attribute wegfällt. Um von dieser Möglichkeit Gebrauch zu machen, muß dies dem Kantenstückbaumakteur bei einem Ausgliederungsauftrag durch zusätzliche Übergabe eines speziellen Parameters (*:only–hot–changes T*) mitgeteilt werden. Die Struktur betroffener Kantenstückbäume wird hierdurch jedoch zerstört und kann nur durch ein Aufheben dieses Schrittes mit Hilfe des Undo–Akteurs wiederhergestellt werden.

1.4.2.3 Kennungen und Referenzen anpassen

Durch das Kopieren von Plankomponenten werden ebenfalls die enthaltenen Referenzattribute zu anderen Plankomponenten oder eigenen Unterkomponenten kopiert. Dadurch ergibt sich eine Situation, wie in Bild–35 anhand zweier Kantenstückbäume verdeutlicht, von denen der eine die Kopie des anderen darstellt.



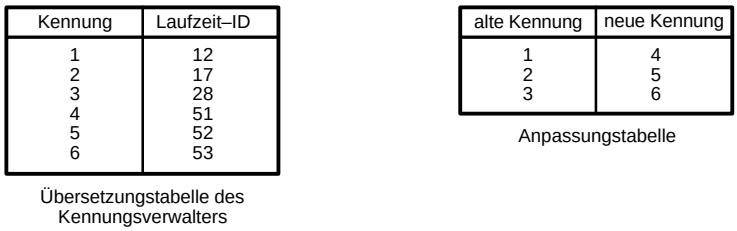
Bild–35: Kennungs– und Referenzproblematik beim Kopieren und Einfügen

Unabhängig voneinander betrachtet, ist sowohl die Struktur aus ursprünglichen Plankomponenten und den Kennungsverweisen untereinander als auch die Struktur aus den kopierten Plankomponenten und deren Kennungsverweisen untereinander eindeutig definiert und abgeschlossen. Probleme ergeben sich erst, wenn beide Strukturen gleichzeitig in einem Plan vorkommen. Dadurch, daß mehrere Komponenten dieselbe Kennung haben, ist logisch keine eindeutige Zuordnung mehr möglich. Durch die Verwendung einer Übersetzungstabelle zur Abbildung der permanenten Referenzschlüssel auf die Laufzeit–IDs kommt es dennoch zu einer eindeutigen aber falschen Zuordnung. Da die Kopien nicht in der Kennungstabelle enthalten sind, verweisen alle Referenzen der Kopie indirekt auf die ursprünglichen Objekte. Diese Situation tritt zum

einen zwangsläufig beim Kopieren aber auch beim Einfügen von SPIKES–Objekten aus der Zwischenablage ein.

Es ist daher nötig, immer wenn Kopien von SPIKES–Objekten dauerhaft in einen Plan übernommen werden, die Verweise danach innerhalb der kopierten Struktur zu erneuern. Bei Kopien, die direkt auf die Zwischenablage ausgeschnitten werden sollen, ist dies zunächst nicht nötig, da eine solche Anpassung voraussetzt, daß man die bereits vergebenen Kennungen des Planes, in den die SPIKES–Objekte eingefügt werden sollen, kennt. Die Anpassung erfolgt in diesen Fällen erst beim Einfügen.

Um die Kennungen und Referenzen einer kopierten Struktur zu Erneuern verfährt man nach dem in Bild–A27 gezeigten Verfahren. Alle Objekte innerhalb der kopierten Struktur benötigen eine neue Kennung, gleichzeitig sollen die durch die Referenzen realisierten Relationen zwischen den Objekten erhalten werden. Dazu geht man in zwei Schritten vor. In einem ersten Schritt wird jedem Objekt eine neue planweit einmalige Kennung zugewiesen. Für die Vergabe dieser Kennungen ist der Planakteur in seiner Rolle als Kennungsverwalter zuständig. Um eine Anpassung der Referenzen an diese neuen Kennungen zu ermöglichen, werden dabei alte und neue Kennung paarweise in einer zuvor angelegten Anpassungstabelle abgelegt. Da es sich bei den betrachteten Objekten um eine Menge von Grafikkomponenten handelt, die ihrerseits weitere Grafikkomponenten mit Referenzen auf beliebige Objekte der kopierten Struktur enthalten, wird dieser Schritt für jede getroffene Grafikkomponente rekursiv ebenfalls für alle enthaltenen Kinder und Kindeskinder durchgeführt. Im Beispiel ergibt sich folgendes Bild:



Bild–36: Zuweisung neuer Kennungen und Anpassung der Referenzen

In einem zweiten Schritt werden für alle betroffenen Objekte, die Referenzattribute angepaßt. Dazu wird anhand der Anpassungstabelle jedes Referenzattribut, das auf einer alten Kennung basiert, gegen ein Referenzattribut entsprechend der neuen Kennung ausgetauscht. Ist dies für alle Objekte geschehen, entsprechen alle Relationen der kopierten Objekte wieder eindeutig denen der ursprünglichen Struktur. Im Anschluß daran kann die überflüssig gewordene Anpassungstabelle gelöscht werden.

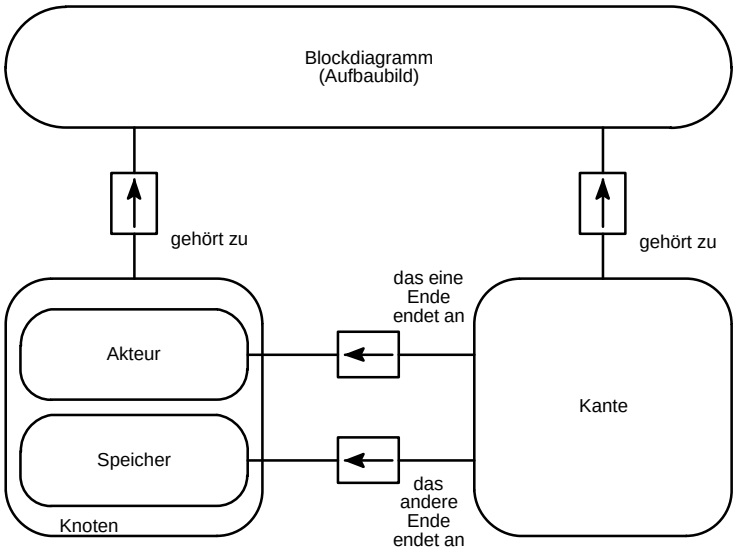
Damit der Planeditor nicht für jeden Typ von Grafikkomponenten wissen muß, ob und wenn ja welche Attribute anzupassen sind, verfügt jeder SPIKES–Grafikakteur über zwei typspezifische Operationsakteure, die entsprechend dem jeweiligen Schritt vom Planeditor angestoßen werden.

Die wesentlichen Konzepte und Aufgaben des Planeditors wurden damit vorgestellt. Der für alle SPIKES–Editoren gleichermaßen gültige erste Teil dieser Arbeit ist hiermit abgeschlossen.

2 Blockdiagramm–Editor

2.1 Blockdiagramme

Blockdiagramme, auch Aufbaudiagramme oder Instanzennetze genannt, dienen dazu, den Aufbau von Systemen mit diskretem Verhalten darzustellen (vgl. [5]). Jedes dieser Systeme kann als Gebilde aus Speichern und Akteuren aufgefaßt werden. Speicher können zum einen statische Speicher sein mit der Aufgabe Information über eine zeitliche Distanz zu bewahren und zum anderen flüchtige Speicher, die als Kanal für Ereigniskommunikation dienen. Die Akteure sind verantwortlich für das Systemverhalten, welches durch die Verhaltensbeschreibung eines jeden Akteurs und der Kopplung der Akteure untereinander bestimmt wird. Demzufolge handelt es sich bei Blockdiagrammen um bipartite Graphen, das heißt, um Graphen, in denen zwei Typen von Knoten auftreten, nämlich Speicher und Akteure, wobei immer nur Knoten der einen Klasse mit Knoten der anderen Klasse in Verbindung stehen dürfen. Diese Verbindungen werden grafisch durch Linien, sogenannte Kanten, dargestellt (vgl. Bild–37).



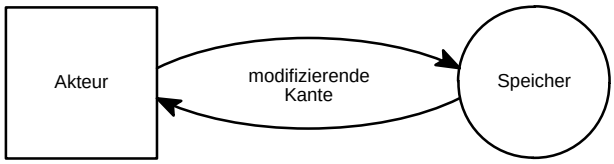
Bild–37: Objektmodell für Blockdiagramme

In Blockdiagrammen werden drei verschiedene Arten von Kanten unterschieden.

- *Gerichtete Kanten* für lesenden oder schreibenden Zugriff eines Akteurs auf einen Speicher. Da diese Darstellungsform keinerlei Auswahl (Adressierung) eines

bestimmtes Bereiches innerhalb des spezifizierten Speichers zuläßt, wird davon ausgegangen, daß bei einem Zugriff immer der gesamte Speicherinhalt gelesen oder neu geschrieben wird.

- *Ungerichtete Kanten* werden verwendet, wenn man keine Aussage über die Art des Zugriffes eines Akteurs auf einen Speicher machen möchte. Dies kann insbesondere bei unspezifizierten Kommunikationskanälen zwischen zwei Akteuren sinnvoll sein.
- *Modifizierende Kanten* werden verwendet, um einen *modifizierenden Zugriff* auf einen Speicher darzustellen. Bei einem solchen Zugriff wird nur eine bestimmte Auswahl der Information eines Speichers durch einen Akteur verändert. Man spricht auch von einem *selektiven Zugriff*. Um dies darzustellen hat man einerseits die Möglichkeit den Speicher feiner zu strukturieren und explizit den Zugriff auf einzelne Bereiche des Speichers einzuzeichnen, die wiederum selbst eigene Speicher darstellen. Die zweite Möglichkeit einen solchen Zugriff darzustellen, besteht darin, daß der Akteur scheinbar den gesamten Inhalt des Speicher zuerst liest, dann intern Teile der Information abändert und die gesamte Information inklusive des modifizierten Teiles zurückschreibt. Dieser doppelte Zugriff ist einfach durch eine schreibende und eine lesende Kante nebeneinander darstellbar. Will man dabei unterstreichen, daß nur Teile der Information geändert werden bzw. nur einzelne Teile des Speichers beschrieben werden sollen, zeichnet man beide Kanten leicht zueinander gebogen ein. Das hierdurch entstehende Symbol wird als *modifizierende Kante* bezeichnet.



Bild–38: Darstellung eines selektiven Zugriffes mittels einer modifizierenden Kante

2.2 Unterstützte Symboltypen und Objektmodell

Ausgehend von den vorangehenden Betrachtungen folgt, daß man mindestens fünf unterschiedliche Symboltypen benötigt, um mit Hilfe von SPIKES–Editoren Blockdiagramme zu erstellen: Speicher und Akteure, gerichtete und ungerichtete Kantenstücke, aus denen entsprechend dem Kantenstückkonzept (vgl. Kap. 1.3) die Kanten zusammengesetzt sind und die modifizierenden Kanten. Darüber hinaus wurde ein eigener Symboltyp für eine spezielle Klasse von Speichern eingeführt, die einfachen Kanäle. Bild–A28 zeigt das sich daraus ergebende Objektmodell für die mit dem Blockdiagrammeditor erstellbaren Pläne, sowie alle zur Bearbeitung oder Verwaltung benötigten Akteure im Editorsystem. Um unnötige Wiederholungen zu vermeiden und

trotzdem eine durchgängige Beschreibung dieses Modells zu erreichen, wird im folgenden nacheinander jeweils ein Symboltyp oder eine Gruppe verwandter Symboltypen vorgestellt, wobei Aufbau und Datenstruktur im Vordergrund stehen. Dies sind in Kapitel 2.2.1 die Knoten, in Kapitel 2.2.2 die gerichteten und ungerichteten Kanten und in Kapitel 2.2.3 die modifizierenden Kanten. Nur beim Symboltyp der modifizierenden Kanten wird dabei zusätzlich näher auf Erzeugung und Bearbeitung, sowie die dazu nötigen Akteure eingegangen, weil dies der einzige Symboltyp ist, der nicht auf allgemeinen SPIKES–Symboltypen aufbauen kann.

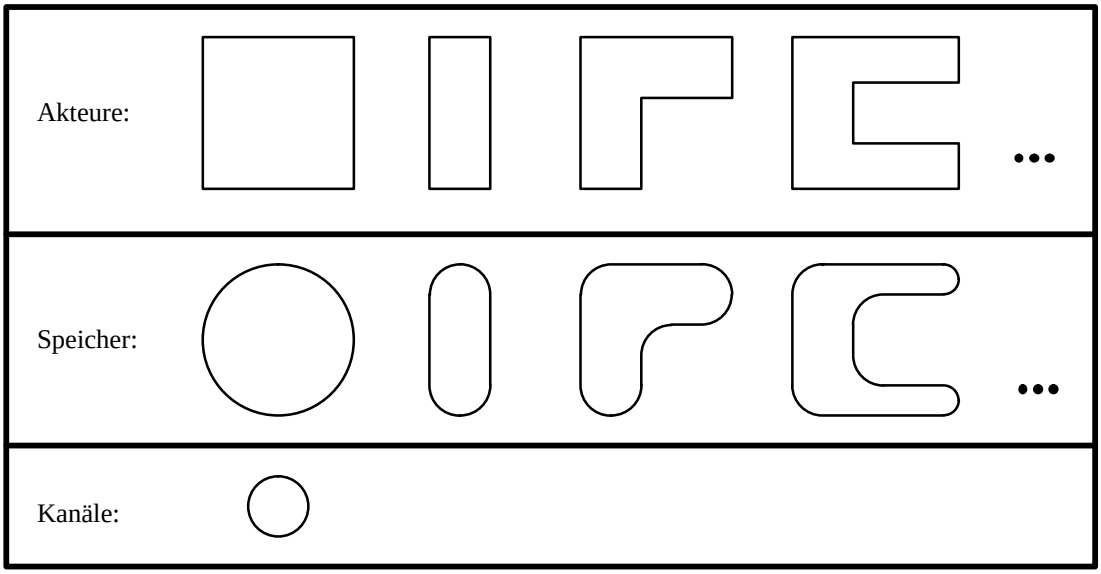
2.2.1 Knoten

2.2.1.1 Symboltypen

Die verschiedenen Knotentypen stellen sich in Blockdiagrammen wie folgt dar:

- Jeder *Akteur* wird durch ein zusammenhängendes flächenhaftes Symbol aus zueinander rechtwinkligen Umrißlinien dargestellt.
- Jeder *Speicher* wird durch ein zusammenhängendes flächenhaftes Symbol aus verrundeten zueinander rechtwinkligen Umrißlinien dargestellt. Im Extremfall können die Linien völlig verschwinden, so daß Halbkreise oder gar ein ganzer Kreis entstehen können.
- Einfache *Kanäle* werden durch einen Kreis dargestellt.

Bild–39 gibt eine Vorstellung der sich daraus ergebenden Variationsmöglichkeiten.

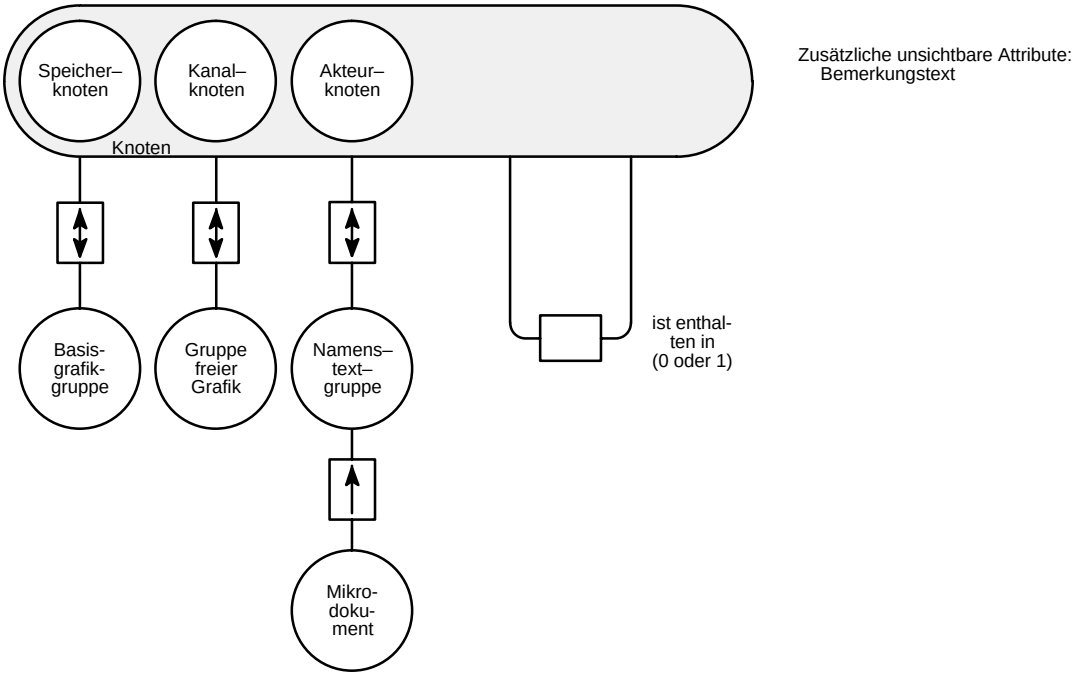


Bild–39: Darstellungsformen der in Blockdiagrammen vorkommenden Knotentypen

Die Einführung zweier Symboltypen für Speicher und Kanäle beruht auf 2 Gründen. Zum einen hat der Betrachter des Planes eine optische Unterstützung bei der Differenzierung zwischen zu Transportzwecken dienenden Kanälen und den informationshaltenden Speichern. Zum anderen besteht bei Kanälen kein Bedarf einer weiteren individuellen Spezifizierung, wohingegen allgemeine Speicher in der Regel größer dargestellt werden, zumeist über einen individuellen Namen verfügen, der die gespeicherte Information kennzeichnet und freier in ihrer Formgebung variierbar sein dürfen. Es ist daher sinnvoll, diese Differenzierung bei der Edition von Blockdiagrammen zu unterstützen und dazu zwei getrennte Symboltypen mit getrennten Defaultwerten und Editionsmöglichkeiten vorzusehen.

Die symboltypischen Umrisse sind bei allen Knoten als Basisgrafikgruppe realisiert. Abgesehen von diesen scheinbar großen Unterschieden zwischen den drei Knotentypen und der unterschied-

lichen Interpretation der Symbole, gibt es jedoch keine typspezifischen Besonderheiten. Bild–40 zeigt das für alle Knoten in SPIKES–Blockdiagrammen gültige Objektmodell.



Bild–40: Objektmodell der Knotensymbole

2.2.1.2 Unsichtbare Attribute

Es ist in Blockdiagrammen erlaubt, daß jeder Knoten eines beliebigen Typs andere Knoten beliebigen Typs enthalten kann. Aufgrund dieser Beziehung werden jedem Knoten zwei unsichtbare Attribute zugeordnet. Das eine beschreibt die Menge aller enthaltenen Knoten (*contained–nodes*) und das andere Attribut ist ein Verweis auf den diesen Knoten seinerseits enthaltenden Knoten (*containing–node*). Grafisch ist eine Enthaltenseinsbeziehung dadurch gekennzeichnet, daß der Umriß eines Knotens den Umriß eines anderen Knotens vollständig umschließt. Die Beziehung wird aber erst bei einem expliziten Aufruf des Syntaxprüfers ermittelt. Bis dahin sind die Attribute undefiniert, auch wenn grafisch offensichtlich ein Enthaltensein vorliegt. Ebenfalls erst beim Syntaxcheck definiert wird das Attribut, in dem Verweise auf alle am Knoten endenden Kanten eingetragen sind (*connected–arcs*). Wie diese Beziehungen ermittelt und ausgewertet werden, wird im Rahmen dieser Arbeit nicht dargestellt (vgl. [7]).

Das einzige normalerweise unsichtbare Attribut, das bearbeitet werden kann, ist der Bemerkungstext (*annotation–text*). Es handelt sich hierbei um eine beliebige Zeichenfolge. Dieser Text wird nur angezeigt, wenn der Benutzer das Formular (Eigenschaftenblatt) eines Knotens öffnet. Denkbare Nutzungsmöglichkeiten dieses unsichtbaren Textattributes reichen von einfachen formlosen Bemerkungen bis hin zu formal interpretierbaren Implementierungsbeschreibungen. Bezüglich Form und Länge dieses Textes existieren keinerlei Einschränkungen.

2.2.1.3 Sichtbare Attributgrafikgruppen

Alle Knoten verfügen neben der für alle SPIKES–Symbole definierten Grafikgruppe für beliebige freie Grafik nur noch über eine Textattributgrafikgruppe. Diese Gruppe dient dazu, Interleaf–Mikrodokumente beliebigen Typs aufzunehmen. Üblicherweise ist für jeden Knoten ein Dokument vorgesehen, das den Namen des Akteurs, sowie möglicherweise eine kurze ergänzende Beschreibung angibt. Aus diesem Grund wird diese Gruppe auch als *Namenstextgruppe* (*name–text*) bezeichnet. Für jeden Knotentyp ist ein Initialdokument definierbar, das beim Erzeugen eines Knotens an einer vorgebbaren Position relativ zur Basisgrafik miterzeugt werden kann. Je nach Vorgabe kann dieses Initialdokument bereits Text enthalten oder nicht. Leere Dokumente können genutzt werden, wenn lediglich das Erscheinungsbild (Schrift, Formatierung usw.) des nachträglich einzugebenden Namenstextes einheitlich vorgegeben werden soll. Von der Möglichkeit der Textvorgaben wird Gebrauch gemacht, um beispielsweise alle Auftrags– und Rückmeldungskanäle mit einem bestimmten Standardtext in der Mitte des Knotens zu kennzeichnen. Darüber hinaus sind bei anschließender Bearbeitung beliebige weitere Mikrodokumente erzeugbar. Alle Dokumente innerhalb dieser Gruppe sind völlig frei zur übrigen Grafik des Knotens positionierbar. Die Bearbeitung einer solchen Namenstextgruppe geschieht in einem eigenen Editionszustand (vgl. Bild–A31), für den der *Auswahlverwalter von Textattributen* verantwortlich ist. Dabei kann der Benutzer auf die volle Interleaf–Funktionalität bei der Bearbeitung von Mikrodokumenten zurückgreifen.

Um gegebenenfalls eine automatische Auswertung des Namenstextes vorzunehmen, existiert die Möglichkeit, den Text aller enthaltenen Mikrodokumente zu einem einzigen String verkettet zu erfragen. Eine denkbare Anwendungsmöglichkeit besteht beispielsweise in der Extraktion und Auswertung der gesamten logischen Information eines Blockdiagrammes. Auf diese Punkte soll an dieser Stelle jedoch nicht eingegangen werden.

2.2.1.4 Implementierung der Knoten

Für jeden Knotentyp existiert eine eigene SPIKES–Grafikgruppenklasse, deren Exemplare die jeweiligen Symbol(akteur)e darstellen. Bis auf die Definition des Namenstextattributes und gewissen Unterschieden der Erzeugungs– und Bearbeitungsmethoden können alle drei Knotenklassen ihre Definition von der allgemeinen SPIKES–Knoten–Klasse *(spk–node–class)* erben. Die für die Implementierung der Knoten entscheidenden Merkmale von Vorkommen dieser Klasse sollen deshalb kurz dargestellt werden (ausführliche Beschreibung und Bearbeitung siehe [7]). Die Basisgrafik jedes allgemeinen Knotensymbols stellt einen beliebigen rechtwinkligen Polygonzug dar, dessen Ecken verrundbar sind und dessen vom Polygonzug umfaßte Fläche mit einem durch den Benutzer wählbaren Muster einer ebenfalls wählbaren Farbe füllbar ist.

Bei endlichem Verrundungsradius entspricht dies genau den Anforderungen, die an Speicherbasisgrafiken gestellt werden. Kanalbasisgrafiken lassen sich dadurch erzeugen, daß das Polygon ein Quadrat darstellt, dessen Kantenlänge exakt zweimal dem Verrundungsradius entspricht. In

diesem Fall hat das verrundete Quadrat die Form eines Kreises. Wählt man bei beliebigen rechtwinkligen Polygonen den Verrundungsradius Null, so entsteht die Basisgrafik eines Akteurs.

Beim Erzeugen eines allgemeinen Knotensymboles entsteht basierend auf den Defaultwerten Höhe und Breite zunächst grundsätzlich ein rechteckiges Polygon, das entsprechend einem weiteren Defaultwert für den Radius verrundet wird. Erst durch anschließendes Strecken oder Stauchen, sowie explizites Bearbeiten des Polygonzuges mit Hilfe des *Editionsakteurs für Knotenumrisse*, kann diese Form verändert werden. Desweiteren besteht die Möglichkeit gemäß Voreinstellung automatisch beim Erzeugen ein Initialdokument zu kreieren. Im Falle der Knoten ist dieses dann in der neu zu definierenden Namenstextgruppe abzulegen. Somit sind alle grafischen Anforderungen bezüglich Blockdiagrammknoten erfüllt. Ebenfalls verfügen allgemeine Knoten bereits über alle zuvor erwähnten unsichtbaren Knotenattribute.

Die explizite Definition von blockdiagrammspezifischen Klassen dient somit nur noch dazu, die im Plan vorkommenden Symboltypen anhand ihrer Klassenzugehörigkeit voneinander zu unterscheiden, dementsprechend unterschiedliche klassenspezifische Voreinstellungen definieren zu können und die Editionsmöglichkeiten typspezifisch einzuschränken, damit nur typgerechte Formen erzeugt werden können. Bei Akteuren darf es nicht möglich sein, nachträglich die Ecken zu verrunden. Umgekehrt darf es bei Speichern nicht möglich sein, die Verrundung so klein werden zu lassen, daß sie durch einen Betrachter nicht mehr als solche erkannt werden kann. Kanäle dürfen durch Bearbeitung ihre Kreisform nicht verlieren. Dazu wird auf jegliche direkte Bearbeitung des Polygonzuges verzichtet und nur eine Vergrößerung bzw. Verkleinerung des Kreises zugelassen.

Zur Verwaltung der jeweiligen typspezifischen Voreinstellungen existiert für den Knotentyp innerhalb eines Blockdiagrammes ein Defaultwertverwalter, der sogenannter Master. Anhang I zeigt die verwalteten Voreinstellungsgrößen für Blockdiagrammknoten.

Eine Besonderheit für Kanalknoten stellt die Möglichkeit dar, verschiedene Initialtexte vorzugeben. Je nach Menüauswahl des Benutzers, wird vor dem Erzeugen eines Kanales der entsprechende Initialtext, sei es für einen Auftragskanal oder einen Rückmeldekanal, durch den Kanalklassenakteur beim Master erfragt. Dieser Text wird dann im Initialdokument eingesetzt. Soll ein leerer Kanal erzeugt werden, wird kein Text vorgegeben. Die auf diesem Wege erzeugten Kanäle können sich somit in ihrem Initialtext unterscheiden, trotzdem handelt es sich bei allen Kanälen um Vorkommen derselben Klasse.

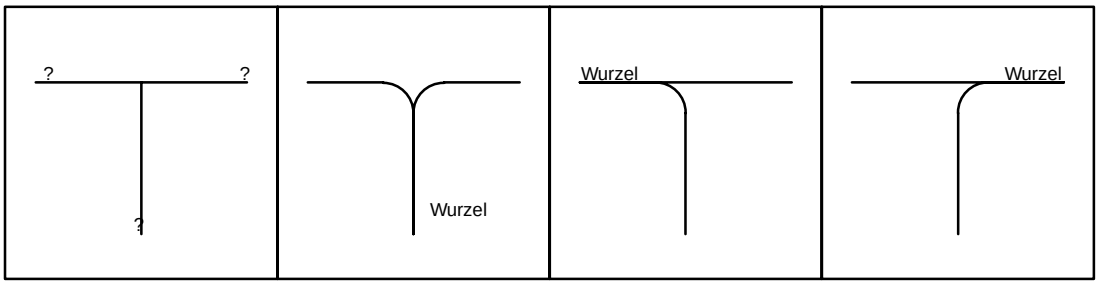
Zur Bearbeitung der Defaultwerte steht für jeden Master ein Formularakteur zur Verfügung. Dabei werden im Formularfenster (Eigenschaftenblatt) jeweils nur die Voreinstellungswerte angezeigt, die auch bearbeitbar sind und nicht völlig fest oder aber direkt von einer anderen Voreinstellungsgröße ableitbar sind. Die Bearbeitung der Voreinstellungswerte wird durch einen entsprechenden Menüaufruf des Benutzer aus dem Repertoire des Auswahlverwalters für die Blockdiagrammebene, das heißt, auf oberster Planebene, eingeleitet. Der dadurch angestoßene Blockdiagramm–Planakteur erzeugt daraufhin für jeden ihm bekannten Master einen passenden Formularakteur und stellt die damit verbundenen Eigenschaftenblätter in einem Auswahlfenster

zur Verfügung. Es ist Aufgabe des Masters in Kommunikation mit dem Formularakteur dafür zu sorgen, daß nur solche Werte akzeptiert werden, die miteinander verträglich sind. Ist dies der Fall, dienen diese Werte nach dem Bestätigen und Schließen eines Eigenschaftenblattes als Vorgaben für das Erzeugen neuer Symbole.

2.2.2 Gerichtete und ungerichtete Kanten

2.2.2.1 Symboltypen

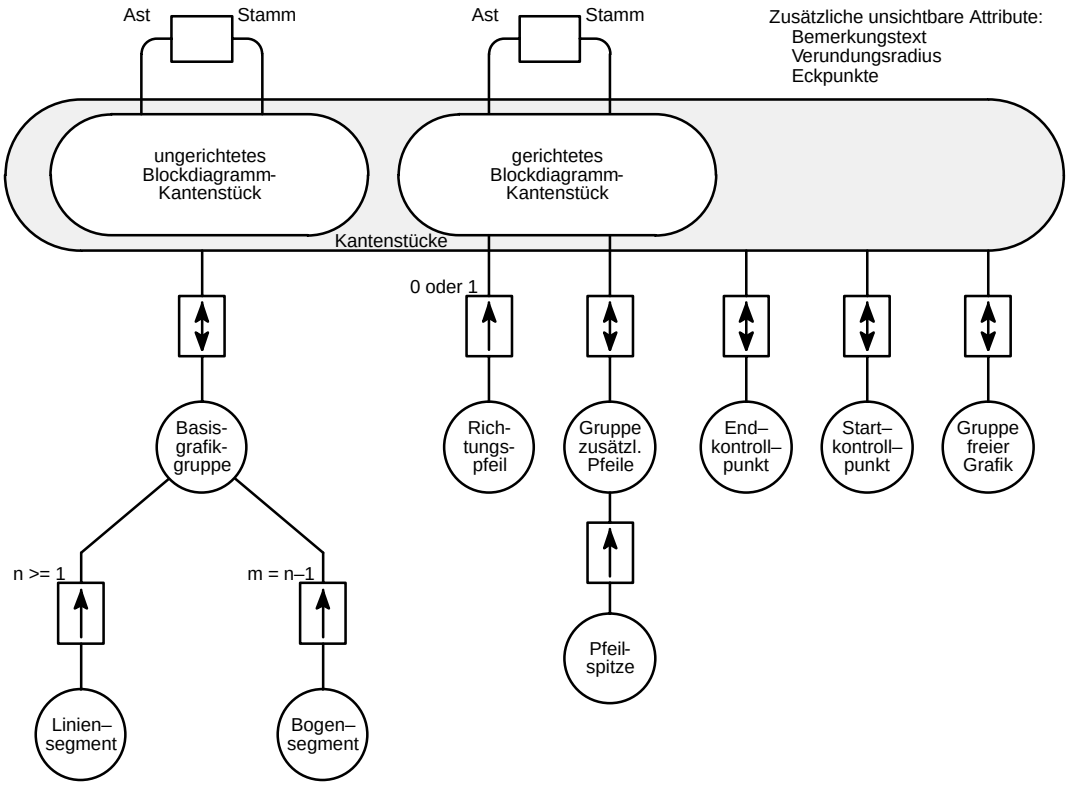
Gerichtete und ungerichtete Kanten werden in Blockdiagrammen durch einen beliebigen Linienzug aus beliebig vielen aneinander gereihten Linienstücken wechselnder Richtung dargestellt. Gerichtete Kanten weisen dabei eine Pfeilspitze an ihrem Endpunkt auf. Formal ist es zulässig, daß durch anfängliches Übereinanderlegen von Kanten ausgehend von einem gemeinsamen Knoten und anschließendem Verzweigen auf verschiedene andere Knoten Kantenbäume entstehen. Diese Kantenbäume können sowohl gerichteter als auch ungerichteter Natur sein. Damit bei ungerichteten Kantenbäumen immer eindeutig eine Wurzel bestimmt werden kann, müssen die Linienzüge der Kanten verrundet sein. Dies verdeutlicht das Beispiel in Bild–41, bei dem bei fehlender Verrundung alle 3 Kantenstücke als Wurzel des Baumes gesehen werden können, wohingegen in allen möglichen Fällen von Verrundung die Wurzel eindeutig bestimmt werden kann.



Bild–41: Festlegen der Wurzel durch Verrunden

Wie bei allen SPIKES–Plänen, wird auch bei Blockdiagrammen jede Kante innerhalb eines Kantenstückbaumes durch die Summe aller aufeinanderfolgenden Kantenstücke ausgehend vom Wurzelkantenstück bis zu dem betreffenden Blattkantenstück definiert. In Analogie zu gerichteten und ungerichteten Kanten werden dabei gerichtete und ungerichtete Kantenstücke unterschieden. Selbstverständlich können dabei innerhalb eines Kantenstückbaumes nur Kantenstücke eines Typs auftreten. Besondere typspezifische Attribute gibt es für Kantenstücke in Blockdiagrammen nicht. Es ergibt sich somit das in Bild–42 gezeigt Objektmodell für Blockdiagrammkantenstücke.

Der verrundete Linienzug ist als Basisgrafikgruppe realisiert. Es soll möglich sein, daß eine automatische Anpassung des Linienzuges auf den Rand eines Knotens erfolgt, wenn das entspre-



Bild–42: Objektmodell der Kantenstücksymbole

chende Linienzugende in die Nähe des Anziehungbereiches eines Knotens kommt. Damit das Kantenstück weiterhin exakt im Raster positioniert werden kann, existieren zwei Kontrollpunkte, von denen sichergestellt ist, daß sie immer im Raster liegen. Diese sind jeweils als eigenes Grafikattribut zu realisieren, um Anfangs– und Endpunkt zu unterscheiden. Neben der grundsätzlich vorgesehen Gruppe freier Grafik sind für ungerichtete Kantenstücke keine weiteren Grafikattribute vorgesehen.

Bei gerichteten Kantenstücken soll die Richtung mittels einer Pfeilspitze angezeigt werden. Grundsätzlich haben dabei nur solche Kantenstücke an ihrem Endpunkt eine Pfeilspitze, die das Ende einer gerichteten Kante darstellen, das heißt entweder das Wurzelkantenstück oder andernfalls die Blattkantenstücke in einem Kantenstückbaum. Darüber hinaus soll es dem Editorbenutzer möglich sein, beliebige zusätzliche Pfeilspitzen in Richtung der Kante entlang des Linienzuges hinzuzufügen. Außerdem soll es möglich sein, gerichtete in ungerichtete Kantenstücke und umgekehrt umzuwandeln.

An unsichtbaren Attributen verfügen Kantenstücke in Blockdiagrammen, außer dem auch für Knoten definierten Bemerkungstext, lediglich über die bereits im allgemeinen Teil angesprochenen Attribute Stamm, Äste, Wachstumsrichtung, sowie die geometrischen Beschreibungsgrößen des Linienzuges Eckpunkte und Verrundungsradius. Zusätzliche Blockdiagramm spezifische Attribute sind nicht definiert.

2.2.2.2 Implementierung der Kantenstücke

Für gerichtete und ungerichtete Blockdiagrammkantenstücke existiert jeweils eine eigene SPIKES–Symbol(akteur)klasse (*bde–dir–arc–comp–class* und *bde–undir–arc–comp–class*). Da die Anforderungen an Blockdiagrammkantenstücke bis auf die Umwandlung zwischen gerichteten und ungerichteten Kantenstücken der Spezifikation allgemeiner SPIKES–Kantenstücke entspricht, wie sie im allgemeinen Teil bereits beschrieben wurden, können beide Klassen als Unterklassen der allgemeinen SPIKES–Klassen für ungerichtete allgemeine Kantenstücke (*spk–arc–comp–class*) und gerichtete allgemeine Kantenstücke (*spk–dir–arc–comp–class*) realisiert werden. Die Exemplare dieser Klassen entsprechen in ihrem Aufbau exakt der im Objektmodell geforderten Vorgabe. Somit können die zum Erzeugen und Bearbeiten benötigten Editionsakteure ebenfalls unverändert übernommen werden. Für die Definition und Bearbeitung des Linienzuges ist die Einheit aus Auswahlverwalter und Editionsakteur für Kantenstücklinienzugedition zuständig. Für das Erzeugen und Bearbeiten zusätzlicher Pfeilspitzen bei gerichteten Linienstücken ist die Einheit aus Auswahlverwalter und Editionsakteur für zusätzliche Pfeilspitzen verantwortlich, die im vollständigen Objektmodell des Blockdiagramm–Editors auf Bild–A28 dargestellt sind.

Eine nähere Betrachtung ist lediglich in Bezug auf die Umwandlung zwischen gerichteten und ungerichteten Kantenstücken nötig. Für diese Umwandlung der Kantenstücke muß ein Klassenwechsel der Kantenstückobjekte vorgenommen werden. Dabei wird folgendes Prinzip benutzt, welches bei einem Klassenwechsel beliebiger Objekte zur Anwendung kommen kann.

Wird ein einzelnes Kantenstück eines bestimmten Typs A in ein Kantenstück eines anderen Typs B umgewandelt, werden zuerst alle für Typ–A spezifischen Attribute entfernt. Dazu wird aufgrund einer Nachricht an das Kantenstück der zugehörige Typ–A spezifische Umwandlungsakteur des Kantenstückobjektes angestoßen. Diesem sind alle Typ–A spezifischen Attribute bekannt, die gelöscht werden müssen. Danach wechselt der Umwandlungsakteur die Klassenzugehörigkeit[34.] des Kantenstückobjektes, wodurch sich zunächst nur die Menge der dem Objekt zugeordneten Operationsakteure klassenspezifisch ändert. Diese Klasse für Kantenstücke des Typs B muß dem Umwandlungsakteur bekannt sein. Um aus dem Kantenstück ein vollständiges Kantenstück des Typs B zu machen, muß das Kantenstück um die Typ–B spezifischen Attribute erweitert werden. Da diese einem Typ–A spezifischen Umwandlungsakteur nicht bekannt sein können, schickt dieser dem übergeordnetem Kantenstückobjekt eine Anpassungsbotschaft. Da das Kantenstück mittlerweile der Klasse B zugehörig ist, kann hierdurch der Typ–B spezifische Anpassungsakteur angestoßen werden, der die ihm bekannten Typ–B spezifischen Attribute erzeugt. Danach ist die Umwandlung abgeschlossen.

Wie bereits erwähnt kann dieses Prinzip grundsätzlich bei jedem Klassenwechsel eines beliebigen Objektes angewendet werden. Im Falle der Blockdiagrammkantenstücke beschränkt sich die

34.. Der Begriff Klasse wird in diesem Zusammenhang ausschließlich in seiner Bedeutung innerhalb der Begriffswelt der objekt–orientierten Programmierung verwendet. Demgegenüber steht der Begriff Typ für ein abstraktes Objekt, das durch alle als klassenzugehörig spezifizierenden Merkmale der Vorkommen einer Klasse definiert ist, entsprechend der Bedeutung des Begriffs Klasse nach [5].

Attributanpassung darauf, Pfeilspitzen zu entfernen oder hinzuzufügen. Allerdings reicht es nicht aus, ein Kantenstück isoliert für sich zu betrachtet. Gehört ein umzuwandelndes Kantenstück einem Kantenstückbaum aus mehreren Kantenstücken an, sind alle Kantenstücke dieses Baumes umzuwandeln. Aus diesem Grund übernimmt die Koordinierung der Umwandlung der Kantenstückbaumakteur. Sind eines oder mehrere Kantenstücke ausgewählt und fordert der Benutzer per Menüaufruf die Umwandlung der Kantenstücke an, stößt der Menüauswahlverwalter dazu den Kantenstückbaumakteur an. Dieser ermittelt die Kantenstücke aller durch die Auswahlmenge spezifizierten Kantenstückbäume und schickt jedem einzelnen Kantenstückakteur die Aufforderung, das Kantenstückobjekt dem Typ entsprechend von gerichtet in ungerichtet oder aber von ungerichtet in gerichtet zu wechseln (*mid:undirekt, mid:direkt*).

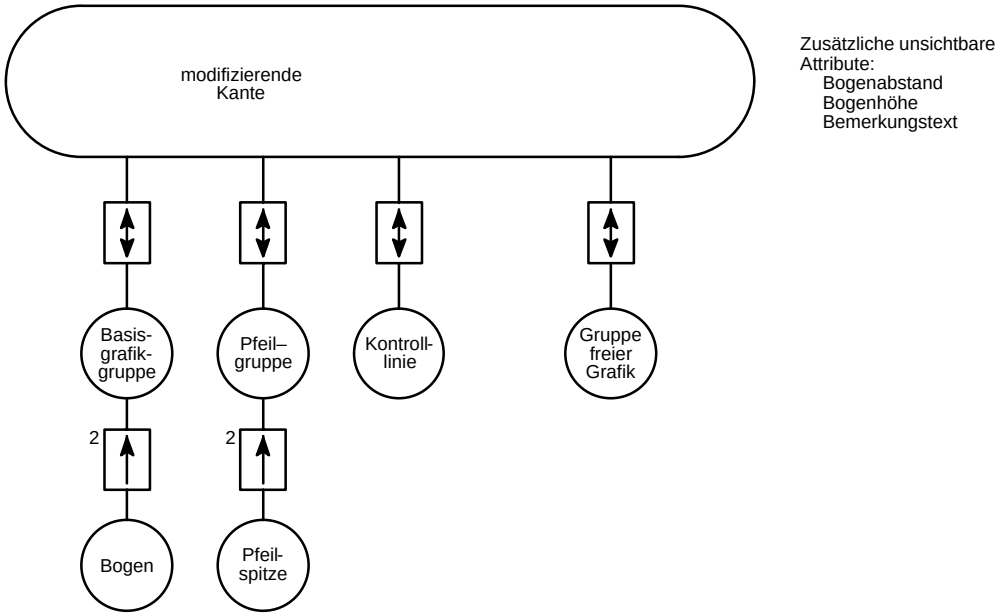
Bei SPIKES–Editoren stimmt die Zeichenrichtung eines gerichteten Kantenstückes mit seiner logischen Richtung überein. Da es bei ungerichteten Kantenstückbäumen zulässig ist, daß die Zeichenrichtung aller im Baum vorkommenden Kantenstücke in ihrer relativen Ausrichtung zueinander von der Wurzel hin zu den Blättern nicht einheitlich sein muß, wird vor der Umwandlung von ungerichteten in gerichtete Kantenstückbäume diese Ausrichtung nachgeholt. Dabei wird willkürlich eine Ausrichtung von den Blättern hin zur Wurzel vorgegeben[35.]. Alle entgegengesetzt gerichteten Kantenstücke sind umzukehren. Bei isolierten Kantenstücken bleibt die ursprüngliche Richtung unverändert.

Bei der Anpassung grafischer Attribute, aber natürlich auch beim Erzeugen neuer Kantenstücke, wird, wie bei allen SPIKES–Symbolen, auf die Voreinstellungswerte der Master zurückgegriffen. Aufgrund der Überlegung, daß ein Benutzer höchstwahrscheinlich für alle innerhalb eines Planes vorkommenden Kanten, unabhängig vom Typ, dieselbe Strichart, Strichstärke oder gleich aussehende Pfeilspitzen vorgeben möchte, existieren für gerichtete und ungerichtete Kantenstücke, sowie für modifizierende Kanten innerhalb eines Plan gemeinsame Voreinstellungswerte, die von einem gemeinsam genutzten Defaultwertverwalter (Vorkommen der *bde–arc–master–class*) verwaltet werden. Die vorgebbaren Größen sind auf dem in Bild–7 gezeigten Eigenschaftenblatt des gemeinsamen Formularverwalters dargestellt.

35.. Der Vorteil dieser Ausrichtung gegenüber der umgekehrten Ausrichtung besteht darin, daß nur eine Pfeilspitze zu erzeugen ist. Da die Wahrscheinlichkeit, daß das Ergebnis den Wünschen des Benutzers entspricht, in beiden Fällen etwa gleich ist, wurde sich für die Lösung mit dem geringeren Aufwand entschieden.

2.2.3 Modifizierende Kanten

Das Symbol der modifizierenden Kante kommt ausschließlich in Blockdiagrammen vor. Aus diesem Grund soll sowohl der Aufbau, als auch die Edition modifizierender Kanten im folgenden vergleichsweise ausführlich betrachtet werden. Modifizierende Kanten werden durch zwei nebeneinander in entgegengesetzter Richtung verlaufende Kanten symbolisiert, die zueinander hingebogen sind und mit jeweils einer Pfeilspitze enden. In dem in Bild–43 gezeigten Objektmodell unterscheidet man dabei die Basisgrafik bestehend aus zwei elliptischen Bögen, eine Gruppe, die beide Pfeilspitzen umfaßt, sowie die Kontrollinie. Spezielle logische Attribute besitzt eine modifizierende Kante nicht.



Bild–43: Objektmodell für das Symbol der modifizierenden Kanten

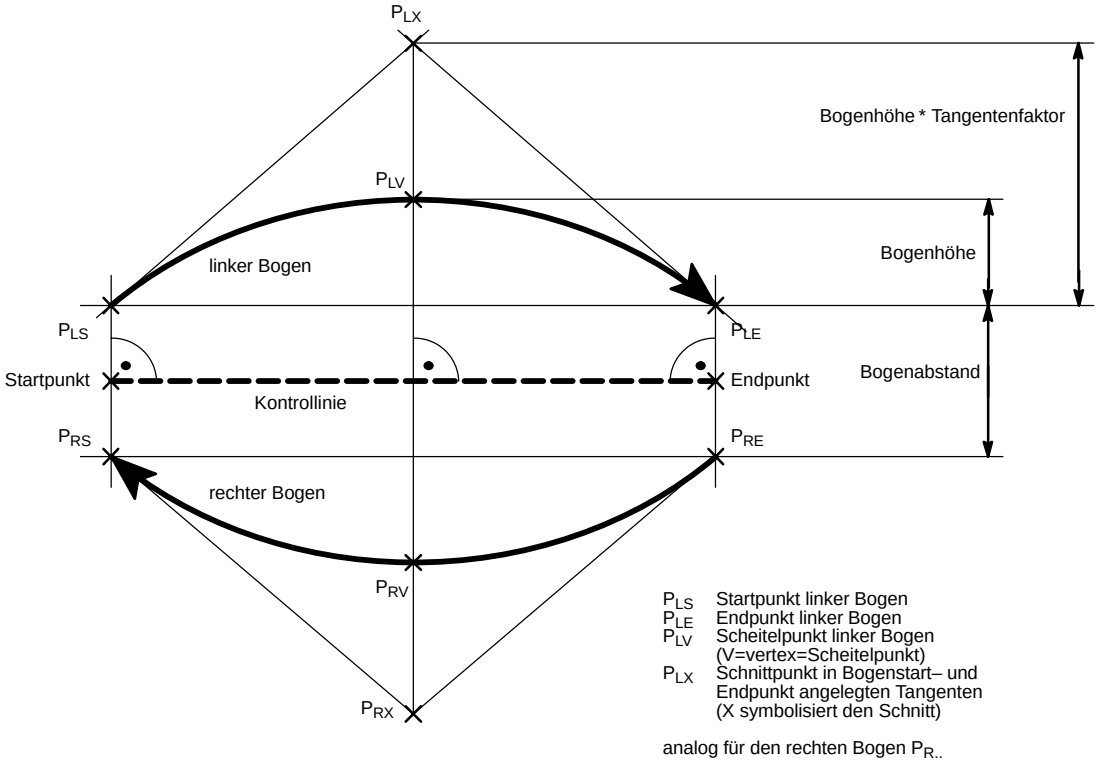
Die zwei Bögen stellen zusammen mit den beiden Pfeilspitzen die eigentliche Symbolgrafik dar. Die Entscheidung, Bögen und Pfeilspitzen in zwei getrennten Gruppen zu realisieren, hängt einerseits damit zusammen, daß Bögen und Pfeilspitzen beim Setzen der Farbe unterschiedlich behandelt werden müssen[36.], andererseits geschieht dies in Übereinstimmung mit den allgemeinen gerichteten SPIKES–Kantenstücken, bei denen die Pfeilspitzen ebenfalls nur Attributgrafiken darstellen. Die Kontrollinie ist eine im nicht ausgewählten Zustand der modifizierenden Kante unsichtbare Linie. Während der Edition einer modifizierenden Kante, dient sie dazu, Start– und Endpunkt der modifizierenden Kante festzulegen. Dabei soll gelten, daß der linke bzw. obere ihrer beiden Endpunkte den Startpunkt und der rechte bzw. untere Punkt den Endpunkt der modifizierenden Kante darstellt. Beide Punkte liegen grundsätzlich im Raster, so daß eine eindeutige Positionierung auf dem Rand eines Knotens, der ebenfalls im Raster liegt mög-

36.. Während für Pfeilspitzen, die als geschlossene Polygone realisiert sind, eine Füllung definiert ist, die beim Setzen der Farbe berücksichtigt werden muß, darf bei Bögen (Vorkommen der dg–arc–class) keine Füllung gesetzt werden, da ansonsten eine im Bezug auf das Symbol der modifizierenden Kante unsinnige Grafik entstehen würde. Durch Verwendung zweier getrennter Gruppen für Bogen und Pfeilspitzen ist eine leichte Identifizierung und schnelle Bearbeitung der beiden Gruppen als je eine Einheit möglich.

lich ist. Diese Punkte bilden zusammen mit den Attributen Bogenabstand und Bogenhöhe die individuelle geometrische Beschreibung einer modifizierenden Kante. Diese Geometrie kann sich ändern, wenn einer der Endpunkte auf einem Knoten liegt. In diesen Fällen ist eine Anziehung der Bögen auf den Rand des entsprechenden Knoten vorgesehen.

2.2.3.1 Geometrische Konstruktion

Die Kontrollinie bestimmt mit Start– und Endpunkt die Position, Länge und Richtung einer modifizierenden Kante. Die elliptischen Bögen links und rechts der Kontrollinie sind ihrerseits vollständig durch jeweils vier Punkten definiert: Start– und Endpunkt der Bogenkurve, den Scheitelpunkt, sowie den Schnittpunkt der Tangenten in Anfangs– und Endpunkt des Bogens[37]. Als Scheitelpunkt wird in diesem Zusammenhang der Punkt auf der Bogenkurve bezeichnet, der von der direkten Verbindungslinie zwischen Start– und Endpunkt des Bogens am weitesten entfernt ist.



Bild–44: Konstruktion einer modifizierenden Kante

Start– und Endpunkt eines Bogens liegen auf einer Geraden parallel der Kontrollinie, die die Mittellinie zwischen beiden Bögen bildet. Der Abstand der durch die Endpunkte beider Bögen definierten Geraden wird als Bogenabstand bezeichnet. Dieser Wert ist vorgeben und soll per Voreinstellung veränderbar sein. Die Position der Bogenendpunkte auf den Geraden kann bei Anziehung variieren. In dem Fall, daß keine Knotenanziehung erfolgt, ist ihre Position durch die Schnittpunkte dieser Geraden mit den zur Kontrollinie senkrecht verlaufenden Geraden durch

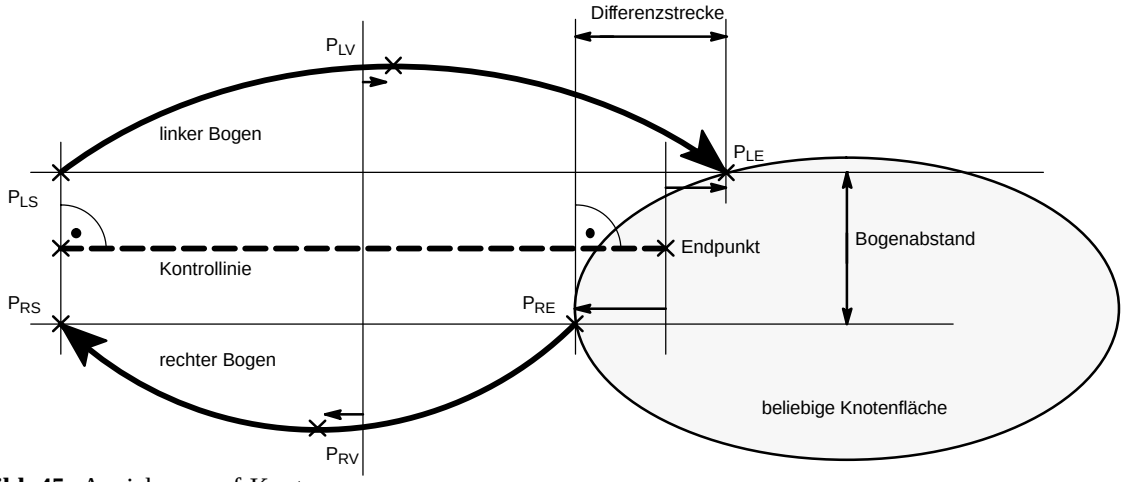
37...Unter Ausnutzung dieser Überlegung wurde eine Erweiterung der Standard–Interleaf–Klasse *dg–arc–class* vorgenommen, die eine Erzeugung elliptischer Bögen basierend auf diesen 4 Punkten zuläßt (vgl. [8])

deren Start– bzw. Endpunkt definiert. Um die verbleibenden zwei Definitionspunkte je Bogen, Scheitelpunkt und Tangentenschnittpunkt zu bestimmen, ermittelt man zuerst den Mittelpunkt zwischen Start– und Endpunkt eines Bogens. Die gesuchten Punkte liegen auf einer Geraden durch diesen Punkt senkrecht zur Kontrollinie auf der Seite des Punktes, die der Kontrollinie abgewandt ist, das heißt, in Richtung der Biegung des Bogens. Der Scheitelpunkt definiert die Höhe des Bogens. Sein Abstand von besagtem Mittelpunkt entspricht dem für die Bogenhöhe frei vorgebbaren Voreinstellungswert. Der Tangentenschnittpunkt, der in Verbindung mit der Länge der modifizierenden Kante die Neigung der Tangenten der Bögen in deren Start– und Endpunkt bestimmt, ist als konstantes Vielfaches der Bogenhöhe durch den Tangentenfaktor fest vorgegeben[38.]. Basierend auf diesen 4 Punkten lassen sich beide Bogen entsprechend der gewünschten Form erzeugen.

Um die modifizierende Kante zu vervollständigen, muß am Endpunkt des linken Bogens und am Startpunkt des rechten Bogens eine Pfeilspitze erzeugt werden. Diese willkürliche Vorgabe dient ausschließlich einem einheitlichen Erscheinungsbild der modifizierenden Kanten. Links und rechts sind dabei relativ auf die Richtung der Kontrollinie von ihrem Start– zu ihrem Endpunkt definiert. Die Richtung der Pfeilspitzen ergibt sich aus der Richtung der Tangenten in den betreffenden Bogenpunkten. Diese entspricht der Richtung des Vektors vom jeweiligen Tangentenschnittpunkt zum betroffenen Bogenendpunkt.

2.2.3.2 Anziehung auf Knoten

Liegt einer der Endpunkte der Kontrollinie einer modifizierenden Kante auf einem Knoten, genauer auf der Knotenfläche insbesondere auf dem Knotenrand, und der andere Linienendpunkt außerhalb dieser Fläche, so wird versucht, die Bögen der modifizierenden Kante an diesem Ende auf den Rand des betroffenen Knoten anzuziehen. Liegen mehrere Knoten übereinander ist selbstverständlich nur der oberste Knoten relevant.



Bild–45: Anziehung auf Knoten

38.. Der Wert wurde auf den kleinstmöglichen Wert gesetzt, für den noch eine Ellipse definiert werden kann (($\sqrt{2}$ / tan(30°)) – vgl. [8]). Mit diesem Wert wird der flachestmögliche Tangentenwinkel erreicht, was einem harmonischen Aussehen der modifizierenden Kante entgegenkommt.

Bei der Anziehung auf einen Knoten verschieben sich Start– oder Endpunkt beider Bögen parallel zur Kontrollinie. Die definierten Voreinstellungswerte Bogenhöhe und Bogenabstand bleiben von der Anziehung unberührt. Basierend auf den veränderten Endpunkten und besagten Defaultwerten können, wie im Falle ohne Anziehung, die Bögen und Pfeilspitzen bestimmt werden. Ein potentieller neuer Endpunkt wird durch den Schnittpunkt der Geraden mit dem in Frage kommenden Knotenrand definiert, auf der die Endpunkte in jedem Fall liegen müssen[39.]. Da es in der Regel jedoch mindestens zwei Schnittpunkte einer Geraden mit dem entsprechenden Knotenrand gibt, wählt man denjenigen Schnittpunkt, der dem Ende der Kontrollinie, das außerhalb der Knotenfläche liegt, am nähesten liegt. Damit Anziehung möglich ist, muß für beide Bögen ein solcher Schnittpunkt bestimmt werden können. Es wäre sinnlos, wenn ein Bogen Anziehung erfährt und der andere nicht. Damit sich durch diese Freiheiten nicht grafisch wenig ansprechende Sonderfälle ergeben können, wie dies z.B. in Bild–45 als abschreckendes Beispiel gezeigt ist, muß außerdem die Bedingung erfüllt sein, daß die bei der Anpassung möglicherweise entstehende Differenzstrecke zwischen den angepaßten Endpunkten beider Bögen in Richtung der Kontrollinie ein gewisses Maß nicht übersteigt. Wählt man den zulässigen Wert entsprechend klein, so heißt dies, daß Anziehung nur dann möglich ist, wenn beide Bögen symmetrisch zur Kontrollinie angezogen werden können[40.].

2.2.3.3 Erzeugen und Bearbeiten

Im folgenden wird das Erzeugen und Bearbeiten von modifizierenden Kanten anhand des Petrinetzes Bild–A29 geschildert. Die modifizierenden Kanten sind als Exemplare einer eigens definierten Symbol(akteur)klasse (*bde–mod–arc–class*) implementiert. Um eine modifizierende Kante zu erzeugen, wählt der Benutzer im Dialogzustand ”Bearbeitung auf Plankomponentenebene” den entsprechenden Menüeintrag aus. Dadurch wird dem Klassenakteur für modifizierende Kanten eine Botschaft zum Erzeugen eines neuen Exemplars (*mid:start–create*) geschickt, inklusive einer Richtungsvorgabe, in der die zukünftige Kante verlaufen soll. Aus Stilgründen sollen in Blockdiagrammen nur horizontale und vertikale modifizierende Kanten erlaubt sein.

Der Klassenakteur wird daraufhin den Defaultwertverwalter für modifizierende Kanten und Kantenstücke ermitteln und von diesem die Voreinstellungswerte für Bogenhöhe und –abstand erfragen. Um die geometrische Beschreibung zu vervollständigen, wird ein eigener Editionsakteur für modifizierende Kanten erzeugt. Um die Edition anzustoßen, wird diesem neben einer Startbotschaft die geometrische Teilbeschreibung einer modifizierenden Kante ohne Start– und Endpunkt aber mit Richtungsvorgabe, sowie der Klassenakteur als Auftraggeber und der Typ der Abschlußbotschaft zugesandt.

Soll eine bereits bestehende modifizierende Kante bearbeitet werden, erhält der Symbolakteur der modifizierenden Kante nach Auswählen der Kante und Menüaufruf durch den Benutzer eine

39.. Man kann dabei ausnutzen, daß sich alle Schnittpunkte eines Knotens mit einer Geraden bei dem entsprechenden Knotenakteur erfragen lassen (*mid:get–intersection–points*).

40.. Bislang (Version 1.0 des Blockdiagrammeditors) ist eine individuelle Vorgabe des Wertes nicht vorgesehen. Stattdessen ist eine feste Konstante (*bde–mod–arc–gravity–tolerance* in *bde–constants.lsp*) definiert, die gerade groß genug, um eventuelle Zeichenungenauigkeiten von Interleaf–V zu tolerieren.

entsprechende Botschaft übermittelt. In diesem Falle wird dem Editionsakteur die vollständige aktuelle geometrische Beschreibung der zu bearbeitenden modifizierenden Kante (Start– und Endpunkt sowie Bogenabstand und Höhe) übermittelt. Alle ursprünglichen Komponenten der modifizierenden Kante werden gelöscht.

Die Aufgabe des Editionsakteurs ist es, in einem interaktiven Prozeß mit dem Benutzer die geometrische Beschreibung zu vervollständigen bzw. zu verändern. Der detaillierte Ablauf ist in Bild–A30 gezeigt. Zuerst notiert der Editionsakteur zunächst alle übergebenen Werte, veranlaßt den Dialogzustandsverwalter in den Dialogzustand ”Bearbeitung modifizierende Kante” zu wechseln und beauftragt den Planeditor eine neue Untergruppe anzulegen, in der losgelöst vom verbleibenden Plan mittels von Hilfsgrafiken die neue Beschreibung durch den Benutzer definiert werden kann.

Sind keine Endpunkte gegeben, müssen diese zunächst ermittelt werden. Dazu wird an der letzten Position des Mauszeigers vor dem die Edition einleitenden Menüaufruf eine Linie erzeugt, die unmittelbar anschließend in der vorgegebenen Richtung in ihrer Größe durch den Benutzer bearbeitbar ist. Nach Festlegen der Größe durch einen abschließenden Mausklick des Benutzers werden die Endpunkte der Linie ermittelt. Nur wenn beide Endpunkte eine unterschiedliche Position haben, können beide Punkte als Start– und Endpunkt der modifizierenden Kante übernommen werden. Ansonsten wird das Ergebnis solange nicht akzeptiert, bis der Benutzer einen sinnvollen zweiten Endpunkt vorgibt. Ist dies geschehen, ist die geometrische Beschreibung vollständig. Es gibt jetzt zwei Möglichkeiten:

- Entweder kann man davon ausgehen, daß der Benutzer mit dieser Definition zufrieden ist und darauf basierend die Grafik der modifizierenden Kante erzeugt haben möchte. In diesem Falle ist die Edition nach der erstmaligen Definition zweier Endpunkte abgeschlossen.
- Oder aber man geht davon aus, daß der Benutzer möglicherweise erst einen Eindruck von dem Ergebnis haben möchte und gegebenenfalls unmittelbar anschließend Änderungen vornehmen möchte. In diesem Fall muß die Hilfsgrafik entsprechend der nun vollständigen geometrischen Beschreibung um zwei Bögen ergänzt werden und der Benutzer hätte nun dieselben Editionsmöglichkeiten, wie bei der Bearbeitung einer bereits existierenden Kante. Ist der Benutzer mit dem Ergebnis zufrieden, muß er explizit die Bearbeitung beenden. Zugunsten eines flüssigeren Editionsstiles wurde sich für das erste Konzept entschieden[41.].

Handelt es sich um die Edition einer bereits bestehenden Kante, steht dem Editionsakteur von Anfang an eine vollständige geometrische Beschreibung zur Verfügung. In diesem Fall erzeugt er eine Hilfsgrafik bestehend aus einer Mittellinie, die der Lage der Kontrollinie entspricht und zwei Bögen entsprechend dem Fall, daß keine Anziehung besteht. Zur Erzeugung der Bögen beauftragt der Editionsakteur den dafür zuständigen Operationsakteur der modifizierenden

41.. Es existiert in der Implementierung jedoch die Möglichkeit durch Umdefinieren einer internen Konstante des Editionsakteurs (*Close–After–Creation–Flag*) zwischen beiden Editionskonzepten zu wechseln.

Kante bzw. des Klassenakteurs modifizierender Kanten[42.]. Auf diesem Weg ist der Editionsakteur unabhängig von der Implementierung der Bögen. Je nachdem ob und was der Benutzer jetzt ausgewählt stehen ihm verschiedene Möglichkeiten zur Verfügung (vgl. Menübaum zum Dialogzustand ”Bearbeitung modifizierender Kante” in Anhang H):

2.2.3.3.1 Länge bearbeiten

Dabei kann die gesamte Hilfsgrafik in Richtung der Mittellinie gedehnt und gestaucht werden. Nach Beendigung der Interaktion mit dem Benutzer erfragt der Editionsakteur die neue Länge beim Mittellinienakteur. Anders als beim Erzeugen wird hierbei jedoch im Falle, daß der Benutzer versehentlich die Länge auf Null setzt, lediglich eine Warnung in der Titelzeile des Dokumentes ausgegeben und die Grafik auf den Stand vor der Längenänderung restauriert.

Für die Änderung der Linienlänge ist derselbe Operationsakteur zuständig, der auch das erstmalige Definieren von Start– und Endpunkt der modifizierenden Kante koordiniert. Die Unterscheidung, wie auf eine unzulässige Benutzervorgabe zu reagieren ist, geschieht in Abhängigkeit eines Flags, das entsprechend bei Empfang des Editionsauftrages nach dessen Auswertung durch den Initialisierungsakteur gesetzt wird.

2.2.3.3.2 Bogenabstand bearbeiten

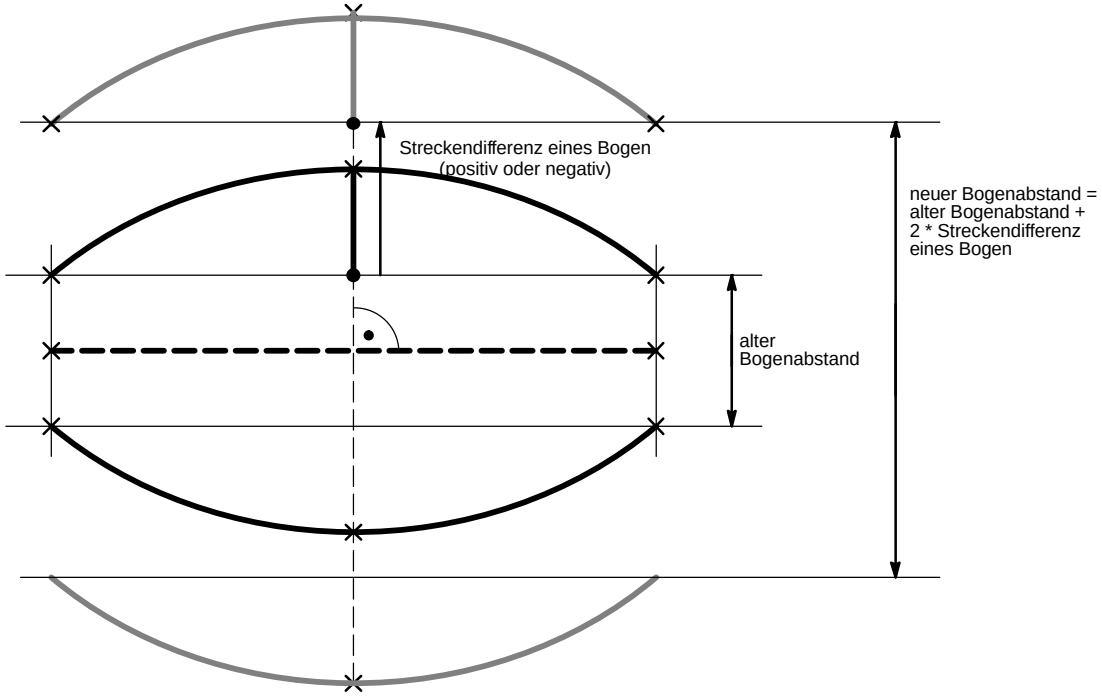
Hierbei wird der Bogenabstand durch gleichzeitiges symmetrisches Aus– bzw. Zueinanderbewegen beider Bögen links und rechts der Mittellinie festgesetzt.
Um eine Edition in der geschilderten Art und Weise zu erreichen, muß man zwei von Interleaf zur Verfügung gestellte Editionselemente verknüpfen. Enthält eine Gruppe nur Grafikobjekte deren Größenverriegelung gesetzt ist, wird sich bei einer anschließenden gemeinsamen Größenbearbeitung aller Objekte dieser Gruppe nicht deren individuelle Größe, wohl aber deren relativer Abstand zueinander ändern.

Man nutzt dies aus, indem man zunächst die Größenverriegelung beider Bögen setzt. Danach läßt man gleichzeitig die Größe beider Bögen als gemeinsame Gruppe in einer Richtung senkrecht zur Mittellinie durch den Benutzer bearbeiten. Wählt man als Fixpunkt der Animation dabei den Mittelpunkt zwischen den Bögen sind beide Bögen gleichzeitig symmetrisch senkrecht zur Mittellinie verschiebbar.

Anschließend muß die neue Bogenentfernung bestimmt werden. Die einzigen Informationen, die so zur Verfügung stehen, sind die umschreibenden Rechtecke der Bögen und die Endpunkte der Mittellinie, die sich bei den zuständigen Grafikakteuren erfragen lassen, sowie die unveränderte Bogenhöhe. Aus diesen Werten läßt sich nur mit relativ hohem mathematischen Aufwand der neue Bogenabstand allgemeingültig bestimmen. Aus diesem Grund erzeugt man vor der Interaktion zwei zusätzliche Hilfslinien, die beide vom Mittelpunkt zwischen Start– und Endpunkt eines

42.. Bei Interleaf–Lisp ist der zuständige Operationsakteur für Nachrichten, die an einzelne Vorkommen einer Klasse geschickt identisch dem zuständigen Operationsakteur des Klassenakteurs dieser Vorkommen, wenn dieser dieselbe Nachricht (mittels *tell–class*) erhält.

Bogens ausgehen und an dessen Scheitelpunkt enden. Ebenfalls setzt man die Größenverriegelung der Linie, die der Benutzer in der anschließenden Edition mitbewegen wird. Da die Linie in der Bewegungsrichtung dieselbe Ausdehnung und Position, wie der korrespondierende Bogen hat und sich die Größe bei beiden durch die Größenverriegelung nicht ändern kann, wird die Linie um dieselbe Strecke, wie der korrespondierende Bogen bewegt. Um die neue Bogenentfernung zu bestimmen, berechnet man die Entfernung vom Startpunkt der bewegten Linie zum Startpunkt der unbewegten Linie. Der neue Bogenabstand ergibt sich als Summe aus der doppelten Bewegungsstrecke eines Bogens und dem ursprünglichen Bogenabstand.



Bild–46: Edition des Bogenabstandes

Wird für den neuen Bogenabstand ein negativer Wert oder null ermittelt, beschreibt dies eine grafisch unzulässige Situation. In diesem Fall wird der Editionsakteur den ursprünglichen Zustand wiederherstellen und dem Benutzer eine Fehlerbeschreibung ausgeben. Ansonsten wird der neue Bogenabstand übernommen. In jedem Fall werden anschließend beide Hilfslinien wieder gelöscht und die Größenverriegelung der Bögen aufgehoben.

2.2.3.3.3 Bogenhöhe bearbeiten

Das anzustrebende Ziel bei der Bearbeitung der Bogenhöhe besteht sicherlich in der gleichzeitigen interaktiven Bearbeitung beider Bogenhöhen. Der dafür zuständige Standard–Interleaf–Akteur für animierte Edition läßt eine solche Edition jedoch nicht zu, ohne gleichzeitig auch den Bogenabstand mit zu beeinflussen. Um die gewünschte Funktionalität trotzdem zu erzielen, hätte der Standard–Interleaf–Akteur für animierte Edition gegen einen eigens hierfür neu (in LISP) implementierten Editionsakteur ausgetauscht werden müssen. Versuche in diese Richtung ergaben jedoch, daß die erzielbaren Ergebnisse zu zeitintensiv gewesen wären, um eine flüssige

Animation zu ermöglichen. Einen Ausweg bietet der folgende Ansatz: Beim Bearbeiten der Bogenhöhe dehnt bzw. staucht der Benutzer die Form von zunächst nur einem Bogen in senkrechter Richtung zur Mittellinie, wobei die Position der Bogenendpunkte unverändert bleibt. Nach Vorgabe der neuen Bogenhöhe durch Fixieren des ersten Bogens durch den Benutzer mittels eines Mausklicks wird auch der zweite Bogen angepaßt, wobei es nur zulässig sein soll, positive Bogenhöhen zu definieren.

Um nach der Interaktion mit dem Benutzer dazu das Ergebnis auswerten zu können, bedient man sich auch hier zweier Hilfslinien, die ebenfalls vom Mittelpunkt zwischen Start– und Endpunkt des betroffenen Bogens ausgehen und an dessen Scheitelpunkt enden. Eine Linie bleibt unverändert, während die andere in gleichem Maße wie der Bogen gedehnt bzw. gestaucht wird. Stellt man durch einen Vergleich beider Linien nach der Bearbeitung fest, daß deren Richtung zueinander nun gegenläufig ist, wurde der Bogen zur Mittellinie hin durchgebogen. In diesem Fall wird der Editionsschritt aufgehoben. Ansonsten wird die Länge der bearbeiteten Linie bestimmt, die exakt der Bogenhöhe des ebenfalls bearbeiteten Bogen entspricht. Wenn diese Länge größer null ist, bedeutet dies daß, für den Bogen eine zulässige Krümmung definiert ist. In diesem Fall wird die neue Bogenhöhe vermerkt und der bislang unbearbeitete Bogen an die neue Höhe angeglichen. Wie bereits beim anfänglichen Erzeugen der beiden Hilfsgrafikbögen wird auch hierfür der dafür zuständige Operationsakteur des Klassenakteurs modifizierender Kanten beauftragt.

2.2.3.3.4 Bewegen

Bei diesem Editionsschritt wird die Mittellinie zusammen mit beiden Bögen an eine beliebige Position innerhalb des Planes bewegt. Unerlaubte Situationen hinsichtlich der geometrischen Beschreibung können hierbei nicht entstehen.

2.2.3.3.5 Abschluß der Edition

Bei Abschluß der Edition stellt der Editionsakteur den ursprünglichen Dialogzustand wieder her, ermittelt Start– und Endpunkt der Mittellinie, sowie die vermerkte Bogenhöhe und den Bogenabstand. Danach entfernt er die gesamte Hilfsgrafik und beendet die Untergruppenbearbeitung. Abschließend sendet er dem vermerkten Auftraggeber neben der ebenfalls notierten Botschaft die ermittelte geometrische Beschreibung zu.

Ist der Auftraggeber der Klassenakteur der modifizierenden Kanten, wird dieser zunächst ein *leeres* Exemplar erzeugen, das heißt, eine modifizierende Kante ohne jede Attributgruppe. Der Klassenakteur beauftragt diese Kante basierend auf der ermittelten geometrischen Beschreibung die Symbolgrafik aus Bögen und Pfeilspitzen zu erzeugen, sowie die Beschreibung zu notieren. Sollte Anziehung auf Knoten möglich sein, wird diese durchgeführt. Die Form der Pfeilspitze, sowie Farbe und Strichart werden bei dem für modifizierende Kanten und Kantenstücke gleichermaßen zuständigen Master erfragt. Da bei Bedarf jederzeit eine neuer Editionsakteur erzeugt werden kann, wird der zunächst nicht mehr benötigte Editionsakteur für modifizierende Kanten aus dem Editorsystem entfernt.

Ähnlich sieht der Ablauf aus, wenn der Auftraggeber eine zu bearbeitende modifizierende Kante war. Diese wird selbst ihren für die Erstellung der Symbolgrafik zuständigen Operationsakteur anstoßen und die gebogenen Pfeile erzeugen lassen. Der einzige weitere Unterschied besteht darin, daß die Werte für Farbe und Strichart der neuen Grafik von der dafür eigens angelegten Kopie der ursprünglichen Basisgrafikgruppe übernommen werden. Diese Gruppe kann anschließend zusammen mit dem Editionsakteur gelöscht werden.

2.3 Benutzen des Blockdiagramm–Editors

Der Blockdiagramm–Editor (Version 1.0) stellt eine Erweiterung des Dokumentenbearbeitungssystems INTERLEAF–V dar. Um diese Erweiterung zu nutzen, müssen sowohl die SPIKES–Basis–Applikation als auch der Blockdiagrammeditor spezifische Part *installiert* sein (Applikationen installieren – vgl. [2]). Ist Interleaf gestartet, kann sowohl ein bereits bestehendes SPIKES–Dokument geöffnet werden, oder aber, falls die Applikation SPIKES mit Hilfe der Werkzeugverwaltung *eingerichtet* wurde, auch ein neues SPIKES–Dokument erzeugt werden.

Wird ein SPIKES–Dokument geöffnet, kann der Benutzer unter dem Menüpunkt ”Anpassen/...” aus der Menge aller momentan unterstützten SPIKES–Plantypen auswählen, ein Plan welchen Typs er erzeugen möchte. Ist der Blockdiagramm–Editor installiert, erscheint hier der Eintrag ”Blockdiagramm”.

Nach dem Erzeugen und Öffnen eines Blockdiagramms befindet sich der Blockdiagrammeditor zunächst im Dialogzustand ”Bearbeitung auf Plankomponentenebene”. Es ist nichts ausgewählt.. Im Falle eines neu erzeugten Planes existieren noch keinerlei Grafikobjekte. Entsprechend dem in Anhang H gezeigten Menübaum kann der Benutzer in diesem Dialogzustand Exemplare der vorgehend beschriebenen Symbolklassen erzeugen sowie gegebenenfalls bereits existierende Symbole auswählen und bearbeiten. Dabei sind entsprechend dem in Bild–A31 gezeigten Dialoggraphen sechs weitere Dialogzustände möglich, in denen der Benutzer individuell die Form der Knoten und Kanten gestalten kann, sowie Kommentarstexte hinzufügen und bearbeiten kann. Jeweils durch Anwahl des Menüpunktes ”Schließen” gelangt der Benutzer wieder in den übergeordneten Dialogzustand. Durch Anwahl des Menüpunktes ”Voreinstellungen” wird ein Eigenschaftenblatt geöffnet, in dem der Benutzer gewisse Voreinstellungswerte (vgl. Anhang I) für die Erzeugung einzelner Symbole individuell für einen Plan definieren kann. Bei den meisten geometrischen Parametern besteht dabei die zusätzliche Möglichkeit die aktuellen Werte eines ausgewählten Symboles zu übernehmen.

Darüber hinaus soll eine logische Auswertung der Planstruktur, bestehend aus Strukturanalyse, Syntaxprüfung und textueller Ausgabe der ermittelten Information möglich sein. Dieser Punkt ist nicht Bestandteil dieser Arbeit und ist im Rahmen der Implementierung eines allgemeinen SPIKES–Syntaxprüfers zu integrieren (siehe [7]).

Wird auf oberster Planebene der Menüpunkt ”Schließen” ausgewählt, wird der Plan, wie jeder andere Grafikrahmen eines Interleaf–Dokumentes für die Bearbeitung bis zum nächsten Öffnen geschlossen. Außerhalb der speziellen SPIKES–Grafikrahmen gestaltet sich die Bearbeitung des Dokumentes durch den Benutzer wie in einem normalen Standard–Interleaf–Dokument. So kann der Benutzer kann auf diesem Wege über die Dokumentenleiste, die in jedem Interleaf–Dokument normalerweise am oberen Fensterrand eingeblendet ist, beispielsweise das Dokument mitsamt des enthaltenen Planes abspeichern oder auch ausdrucken. Weitere Informationen über die Standardfunktionalität von Interleaf erhält der interessierte Leser unter [1].

Anhang A: Literaturverzeichnis

[1] Interleaf 5 User Manual

[2] Interleaf 5 Lisp Reference Manual
Volume 1+2

[3] Beschreibung des INTERLEAF–Systems als Trägersystem für die Entwicklung der Strukturplaneditoren
Projektbericht der AG Digitale Systeme
Universität Kaiserslautern / Fachbereich Elektrotechnik

[4] Implementierung der Planeditoren
W. Kleis 1994/95
Universität Kaiserslautern / Fachbereich Elektrotechnik

[5] Nichtphysikalische Grundlagen der Informationstechnik
S. Wendt
Springer–Verlag

[6] Kantenstückeditionskonzept für die SPIKES–Editoren
W. Kleis 1994
Universität Kaiserslautern / Fachbereich Elektrotechnik

[7] Entwurf und Implementierung des ERD–Editors
Diplomarbeit von R. Krauss (ab Anfang 1996 verfügbar)
Universität Kaiserslautern / Fachbereich Elektrotechnik

[8] weitere Konzepte zu SPIKES–Editoren, sowie Notizen und Vorgaben:
Dokumentensammlung von W.Kleis (inkl. Unterlagen M.Tsesis)
Universität Kaiserslautern / Fachbereich Elektrotechnik

Anhang B: Bilder & Pläne

Anhang C: Modul–Liste

Die folgende Tabelle gibt einen Überblick über im Rahmen dieser Diplomarbeit neu erstellte (alle BDE–Module), sowie erweiterte bestehende SPIKES–Module. Kleinere Moduländerungen, Module, die wegen Konzeptänderungen nicht mehr benötigt wurden und Module, die teilweise auf eigenen Konzepten aufbauen, aber nicht selbst implementiert wurden, sind nicht aufgeführt.

Modulname	Kurzbeschreibung
spk–arc–comp.lsp	Modul, das die Klasse der allg. Kantenstücksymbolakteur definiert, erweitert um Methode(n) zum – Verwalten der Kantenstückbaumrelationen – Feststellen der Verträglichkeit von Kantenstücken bzgl. der Zugehörigkeit zu gemeinsamen Kantenstückbäumen – schnellen Anpassen und Erstellen von Kantenstücken bei Kanten–stückbaumoperationen Außerdem wurde die Datenstruktur der Kantenstückgrafikgruppen im Zusammenhang mit der Kantenstückbaumproblematik leicht abgewandelt (gerichtete Basisgrafikkomponenten)
spk–dir–arc–comp.lsp	Modul, das die Klasse der Symbolakteure für gerichtete Kantenstücke definiert, erweitert um Methode(n) zum – Feststellen der Verträglichkeit von Kantenstücken bzgl. der Zugehörigkeit zu gemeinsamen Kantenstückbäumen – schnellen Anpassen und Erstellen von Kantenstücken bei Kanten–stückbaumoperationen – zur Umwandeln zwischen gerichteten und ungerichteten Kanten – zum Wechseln der Richtung von Kantenstücken
spk–dir–dg–arc.lsp	Modul (neu) zur Definition einer Klasse von Linien mit Richtungs– und Indexattribut.
spk–dir–dg–line.lsp	Modul (neu) zur Definition einer Klasse von Bogenstücken mit Richtungs– und Indexattribut
spk–messages.lsp	Modul, in dem die Nachrichtentypen aller SPIKES–Basis–Module definiert sind
spk–root–contact.lsp	Modul (neu) zur Definition einer Klasse von Datenobjekten, die Kontakte zwischen potentiellen Nachbarn innerhalb von Kantenstückbäumen beschreiben.
spk–tree–manager.lsp	Modul (neu), das die Klasse der Kantenstückbaumbearbeiter beschreibt. Ein Exemplar dieser Klasse wird benötigt zur Erzeugung und Erhaltung der Minimalform von Kantenstückbäumen, sowie einigen baumweiten Operationen.
mat.lsp	Mathematik–Bibliothek, erweitert um einige zusätzliche Funktionen

bde–agent.lsp	Modul zur Definition der Klasse der Akteurssymbole
bde–agent–master.lsp	Modul zur Definition der Klasse der Defaultwertverwalter für Akteursknoten–Symbolakteure
bde–agent–master–submenu.lsp	Modul zur Definition der Klasse der Formularakteure für Akteurdefaultwertverwalter
bde–arc–master.lsp	Modul zur Definition der Klasse der Defaultwertverwalter für die Symbolgrafiken ungerichter und gerichteter Kantenstücke, sowie für modifizierende Kanten(stücke)
bde–arc–master–submenu.lsp	Modul zur Definition der Klasse der Formularakteure für Kantenstückdefaultwertverwalter
bde–arc–submenu.lsp	Modul zur Definition der Klasse der Formularakteure zur Bearbeitung der logischen Attribute von Kantenstücken
bde–channel.lsp	Modul zur Definition der Klasse der Kanalsymbolakteure
bde–channel–master.lsp	Modul zur Definition der Klasse der Defaultwertverwalter für Kanalsymbole
bde–channel–master–submenu.lsp	Modul zur Definition der Klasse der Formularakteure für Kanaldefaultwertverwalter
bde–constants.lsp	Modul zur Definition verwendeter Konstanten z.B. für die voreingestellten Defaultwerte nach dem Erzeugen eines Blockdiagrammes
bde–dir–arc–comp.lsp	Modul zur Definition der Klasse der Symbolakteure für gerichtete Blockdiagramm–Kantenstücke
bde–editor.lsp	Modul zur Definition der BDE–Editor–Klasse. Ein Exemplar dieser Klasse kann als Eingabeauswahlverwalter (Menüakteur) auf oberstem Bearbeitungsniveau eines Blockdiagrammes angesehen werden.
bde–messages.lsp	Modul, in dem die Nachrichtentypen aller BDE–Module definiert sind
bde–mod–arc.lsp	Modul zur Definition der Klasse der Symbolakteure für modizierende Kanten
bde–mod–arc–edit–agent.lsp	Modul zur Definition der Klasse der Bearbeitungsakteure für die Symbolgrafik modifizierender Kanten
bde–mod–arc–editor.lsp	Modul zu Definition der Klasse der Eingabeauswahlverwalter für die Bearbeitungsakteure modifizierender Kanten
bde–node–submenu.lsp	Modul zur Definition der Klasse der Formularakteure zur Bearbeitung der logischen Attribute von Knoten(symbolen)
bde–plan.lsp	Modul zu Definition der Klasse der Blockdiagramm–Grafikrahmenakteure
bde–startup.lsp	Modul, daß beim Laden der Application SPIKES dafür sorgt, daß die Module für die Erstellung und Bearbeitung von Blockdiagramm geladen werden
bde–storage.lsp	Modul zur Definition der Klasse der Speichersymbolakteure
bde–storage–master.lsp	Modul zur Definition der Klasse der Defaultwertverwalter für Speichersymbole
bde–storage–master–submenu.lsp	Modul zur Definition der Klasse der Formularakteure für Speichersymbol–Defaultwertverwalter
bde–text–ui.lsp	Modul, daß den Menüeintrag zum Erstellen eines Blockdiagrammes aus einem SPIKES–Dokument im Texteditormenü anhängt.

bde–undir–arc–comp.lsp	Modul zur Definition der Klasse der Symbolakteure für ungerichtete Blockdiagramm–Kantenstücke

Anhang D: Abbildung des BDE–Objekt–modells auf die Code–Ebene

Die folgende alphabetisch geordnete Tabelle bildet im Objekte des ER–Diagramms Bild–A28 auf die realisierenden Objekte der Code–Ebene ab. Wird ein Objekt durch ein Vorkommen einer SPIKES–Klasse (spk–...–class oder bde–...–class) verkörpert, trägt das implementierende LISP–Modul in der Regel den gleichen Name, wobei die Endung "–class" entfällt.

Bezeichnung im Objektmodell	verkörpert durch
Akteurknoten(akteur)	Vorkommen der bde–agent–class
Auswahlverwalter für animierte Edition	Vorkommen der spk–animation–editor–class
Auswahlverwalter für Edition auf Blockdiagrammebene	Vorkommen der bde–editor–class
Auswahlverwalter für Edition des Umrisses von Knoten	Vorkommen der spk–node–editor–class
Auswahlverwalter für Edition freier Grafik	Operationsakteur des Standard–Interleaf–Dokumenteditionsakteurs (der zugehörige Editionsakteur ist ebenfalls Bestandteil des Standard–Interleaf–Dokumenteditionsakteurs)
Auswahlverwalter für Edition modifizierender Kanten	Vorkommen der bde–mod–arc–editor–class
Auswahlverwalter für Edition von Textattributen	Vorkommen der spk–text–attr–editor–class (der zugehörige Editionsakteur ist Bestandteil des Standard–Interleaf–Dokumenteditionsakteurs)
Auswahlverwalter für Edition zusätzlicher Pfeilspitzen	Vorkommen der spk–arrows–editor–class
Auswahlverwalter für Kantenstück-linienzugeditions	Vorkommen der spk–arc–editor–class
bereichszuständiger Dokumentedi-tionsakteur	Objekt auf das die globale Variable *doc–editor* verweist – je nach Editionszustand einer der obigen Auswahlverwalterklassen zugehörig
Blockdiagramm(akteur)	Vorkommen der bde–plan–class
Defaultwertverwalter für Akteursknoten	Vorkommen der bde–agent–master–class
Defaultwertverwalter für Kanalknoten	Vorkommen der bde–channel–master–class
Defaultwertverwalter für Kanten (Kantenstücke und mod. Kanten)	Vorkommen der bde–arc–master–class

Defaultwertverwalter für Speicherknoten	Vorkommen der bde–storage–master–class
Dokument	Vorkommen der doc–document–class (Standard–Interleaf)
Editionsakteur für animierte Edition	Vorkommen der spk–anim–edit–agent–class
Editionsakteur für Bewegen mit Kontakt	Vorkommen der spk–contact–anim–edit–agent–class
Editionsakteur für Linienzüge	Vorkommen der spk–polyline–edit–agent–class, definiert in spk–polyline–edit.lsp
Editionsakteur für modifizierende Kanten	Vorkommen der bde–mod–arc–edit–agent–class
Editionsakteur für Textattribute	Operationsakteur des Standard–Interleaf Dokumenteditionsakteurs
Editionsakteur für Umrisse von Knoten	Vorkommen der spk–node–agent–class
Editionsakteur für zusätzliche Pfeilspitzen	spk–arrows–edit–agent
Formularakteur für Akteurdefaultwerte	Vorkommen der bde–agent–master–submenu–class
Formularakteur für Kanaldefaultwerte	Vorkommen der bde–channel–master–submenu–class
Formularakteur für Kanten (Kantenstücke und mod. Kanten)	Vorkommen der bde–arc–master–submenu–class
Formularakteur für Speicherdefaultwerte	Vorkommen der bde–storage–master–submenu–class
Formularakteur für unsichtbare Planattribute	Vorkommen der spk–plan–prop–submenu–class, definiert in spk–plan–ui.lsp
Formularakteur für unsichtbare Symbolattribute	Vorkommen der spk–arc–submenu–class bzw. der spk–node–submenu–class. (beide Klassen definieren zur Zeit identische Formulare)
freies grafisches Attribut	Vorkommen der entsprechenden Standard–Interleaf–Klasse
gerichtetes Blockdiagrammkantenstück (Akteur)	Vorkommen der bde–dir–arc–comp–class
Kantenstückbaumakteur	Vorkommen der spk–tree–manager–class
Kanalknoten(akteur)	Vorkommen der bde–channel–class
modifzierende Kante (Akteur)	Vorkommen der bde–mod–arc–class
Planeditor	Vorkommen der spk–plan–editor–class
Planschriftfeld	in der Version 1 noch nicht implementiert
Speicherknoten(akteur)	Vorkommen der bde–storage–class
SPIKES–Cut+Paste–Akteur	Klassenakteur der spk–cut–and–paste–class
Syntaxprüfer	siehe [7]
ungerichtetes Blockdiagrammkantenstück (Akteur)	Vorkommen der bde–undir–arc–comp–class

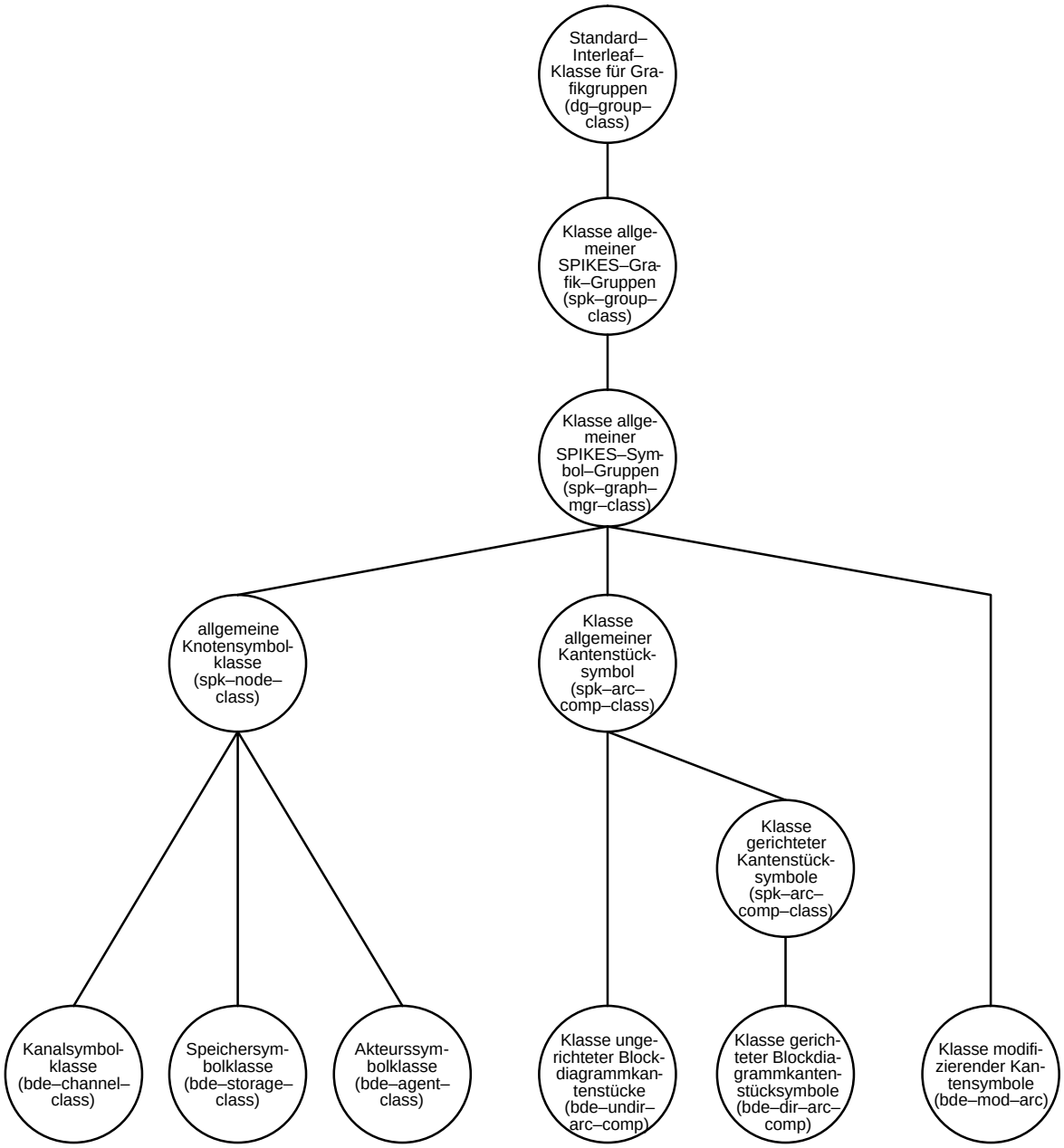
Anhang E: BDE–Dialogzustände und zuständige Editionsakteure und Auswahlverwalter

Die folgende Tabelle dient dazu zu jedem Dialogzustand des Blockdiagrammeditor entsprechend Bild–A31 den zuständigen Editionsakteur und die jeweilige Klasse des bereichzuständigen Menüauswahlverwalters zu ermitteln.

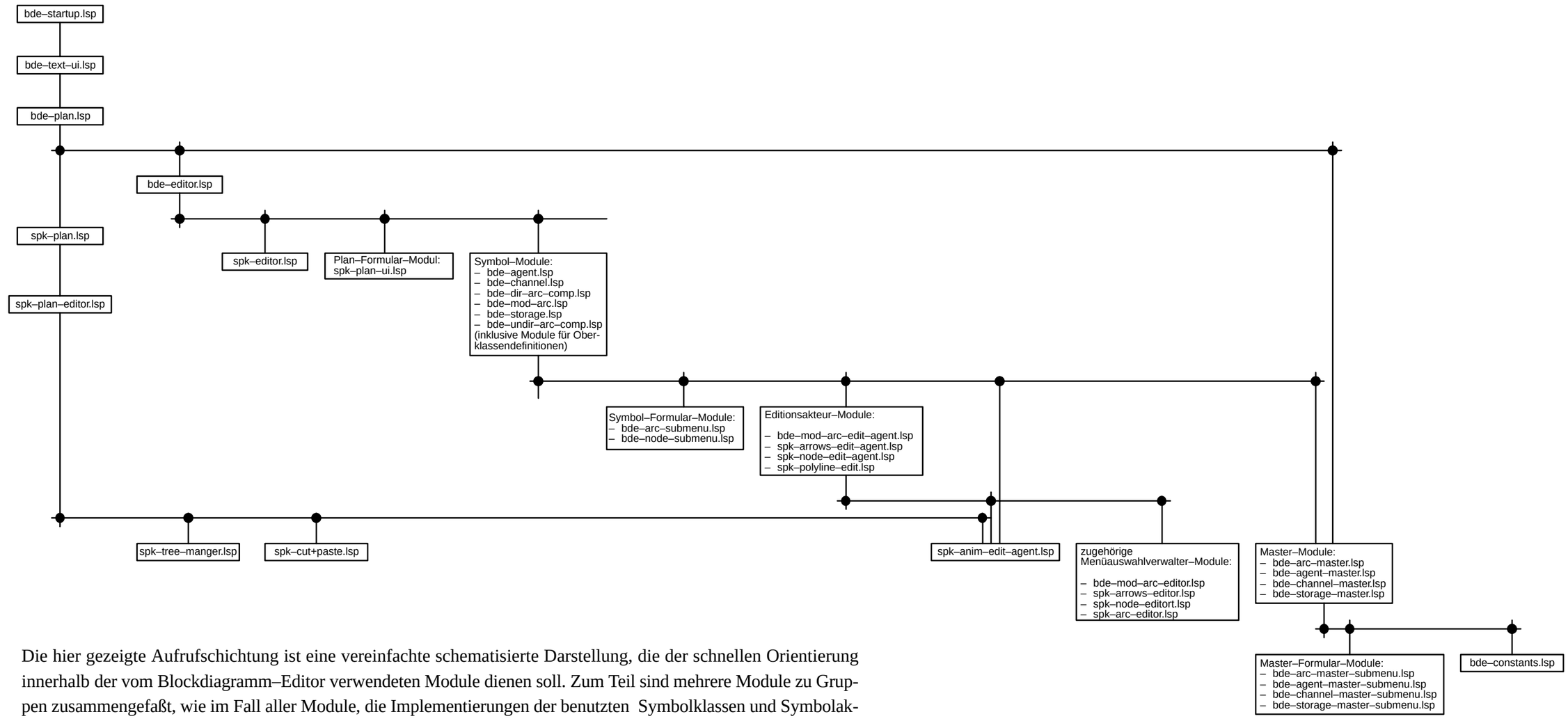
Dialogzustand	Menüauswahlverwalterklasse	Editionsakteurklasse
”Bearbeitung auf Plankomponentenebene”	bde–editor–class (Modul: bde–editor.lsp)	keine definierter Editionsakteur explizit definiert – Befehlsausführende sind der Planeditor, Symbolklassenakteure, Symbolakteure und der Operationsakteur für freie Grafik innerhalb des Interleaf–Basissystem (*dg–primivites*...)
”Linienzugbearbeitung”	spk–arc–editor–class (Modul: spk–arc–editor.lsp)	spk–polyline–edit–agent–class (Modul: spk–polyline–edit.lsp)
”Bearbeitung zusätzlicher Pfeile”	spk–arrows–editor–class (Modul: spk–arrows–editor.lsp)	spk–arrows–edit–agent–class (Modul: spk–arrows–edit)
”Bearbeitung modifizierende Kante”	bde–mod–arc–editor–class (Modul: bde–mod–arc–editor.lsp)	bde–mod–arc–edit–agent–class (Modul: bde–mod–arc–edit–agent.lsp)
”Bearbeitung Namenstext”	spk–text–attr–editor–class (Modul: spk–text–attr–editor)	Editionsakteur ist Bestandteil des Standard–Interleaf–Systems
”Bearbeitung Knotenumriß”	spk–node–editor–class ?	spk–node–edit–agent–class ?
”Bearbeitung freie Grafik”	Akteur ist Bestandteil des Standard–Interleaf–Systems	Akteur ist Bestandteil des Standard–Interleaf–Systems

Anhang F: Klassenhierarchie der Blockdiagrammsymbole

Alle in Blockdiagrammen erstellbaren Symbole sind Vorkommen einer bestimmten SPIKES–Klasse von Grafikgruppen. Das folgende Bild zeigt den dazu gehörenden Klassenbaum ausgehend von der Standard–Interleaf–Klasse *dg–group–class* für einfache Grafikgruppen als Wurzel.



Anhang G: Modulaufrufschichtung des Blockdiagramm–Editors

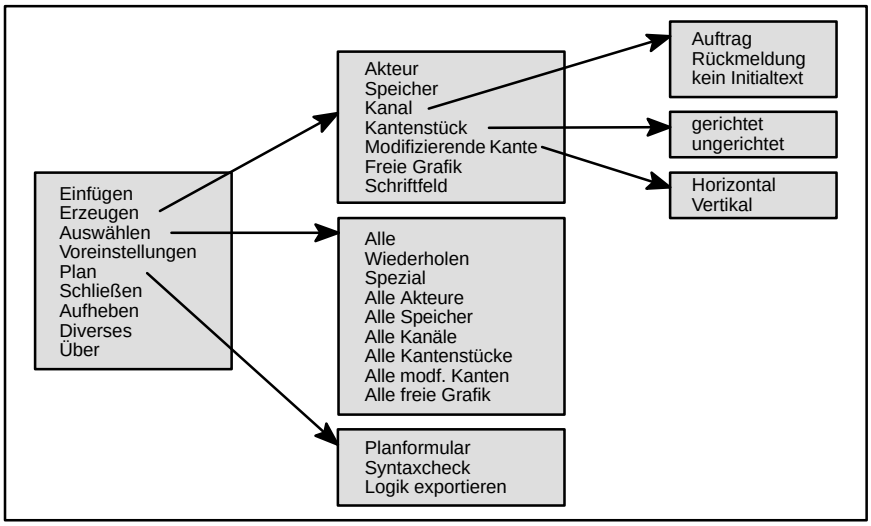


Die hier gezeigte Aufrufschichtung ist eine vereinfachte schematisierte Darstellung, die der schnellen Orientierung innerhalb der vom Blockdiagramm–Editor verwendeten Module dienen soll. Zum Teil sind mehrere Module zu Gruppen zusammengefaßt, wie im Fall aller Module, die Implementierungen der benutzten Symbolklassen und Symbolakteure beschreiben. Vernachlässigt sind Module, die Beschreibungen von Oberklassen darstellen, sowie einige Module, die von allen Modulen nutzbare Makro– und Funktionsbibliotheken definieren. Eine vollständige Darstellung würde mehrere Seiten füllen und alles andere als übersichtlich sein.

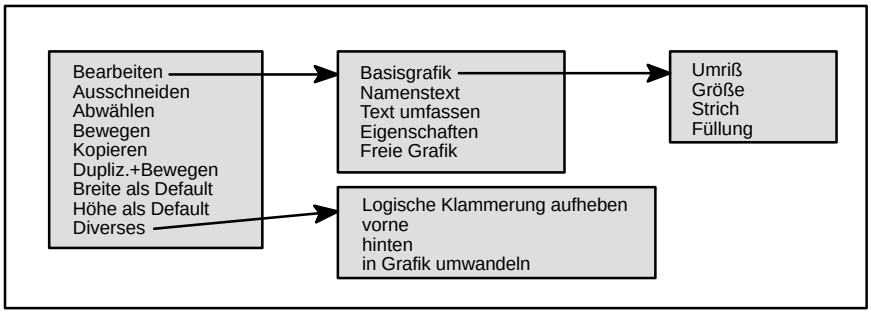
Anhang H: Blockdiagramm–Editor spezifische Menüs

Menübäume im Dialogzustand ”Bearbeitung auf Plankomponentenebene

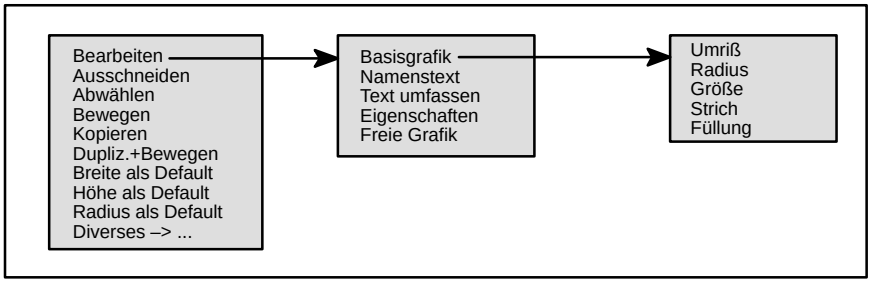
- Nichts ist ausgewählt:



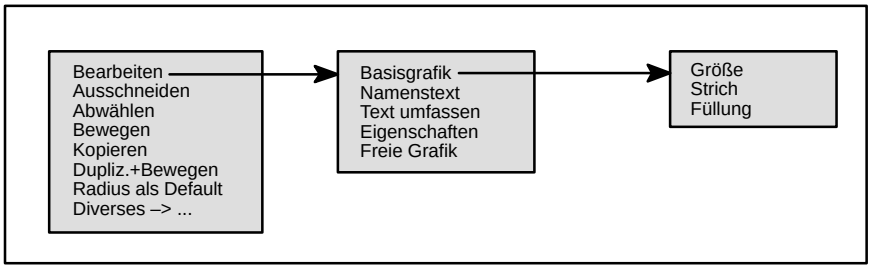
- Genau ein Akteur ist ausgewählt:



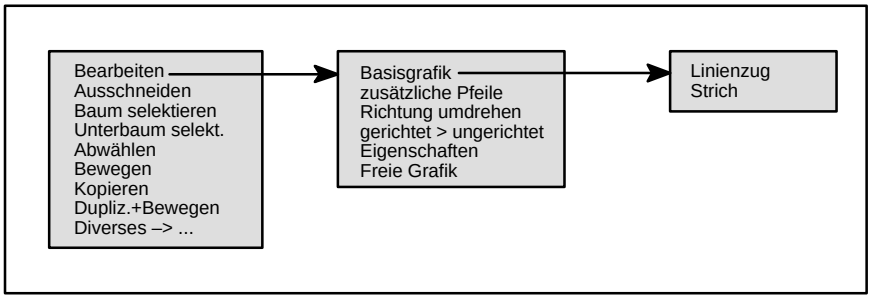
- Genau ein Speicher ist ausgewählt:



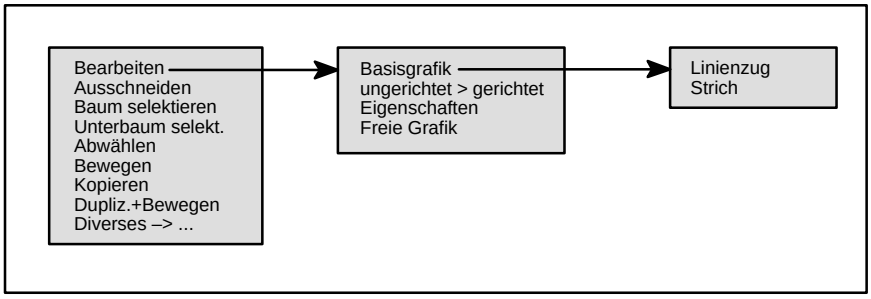
- Genau ein Kanal ist ausgewählt:



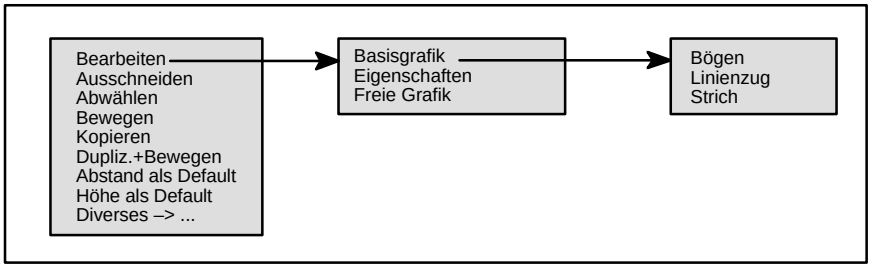
- Genau ein gerichtetes Kantenstück ist ausgewählt:



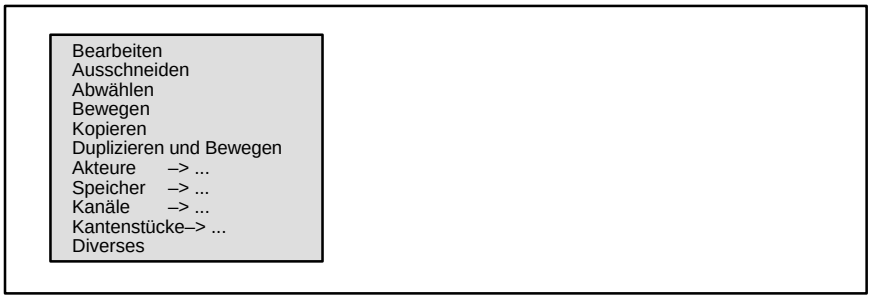
- Genau ein ungerichtetes Kantenstück ist ausgewählt:



- Genau eine modifizierende Kante ist ausgewählt:



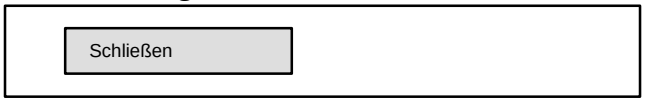
- Mehrere Komponenten des Planes, worunter mindestens ein Symbol ist, sind ausgewählt:



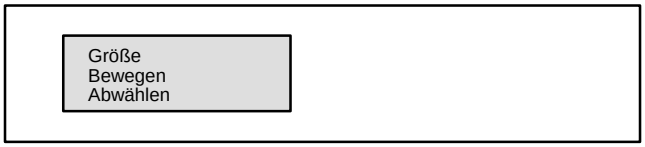
- Nur freie grafische Attribute des Planes sind ausgewählt:
  In diesem Fall steht die Standard–Funktionalität von Interleaf zur Verfügung. Das Auswahl entspricht dem Menü im Dialogzustand ”Bearbeitung freier Grafik” für den Fall, daß etwas selektiert ist.

Menübäume im Dialogzustand ”Bearbeitung modifizierender Kante”

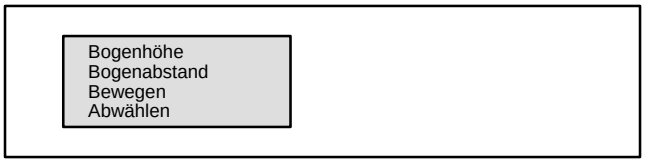
- Nichts ist ausgewählt:



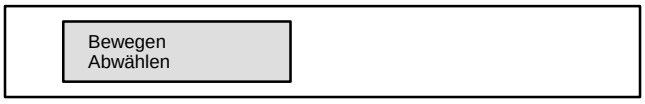
- Nur die Hilfslinie zwischen den zwei Bögen ist ausgewählt:



- Genau ein Bogen ist ausgewählt:



- Mehrere Objekte sind ausgewählt:



Anhang I: Voreinstellungsgrößen aller BDE–Symbole

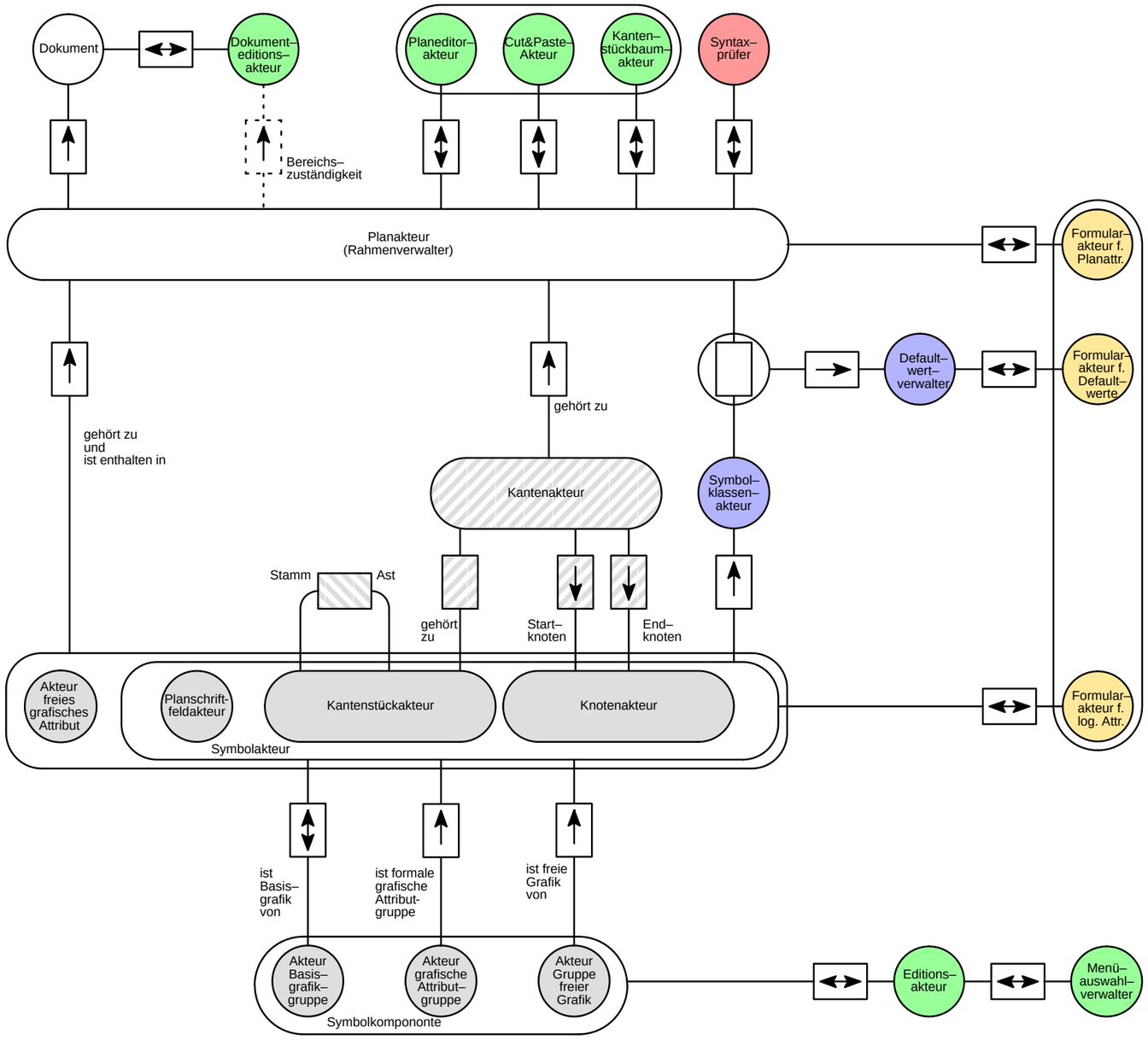
Die nachfolgenden Tabellen zeigen eine Zusammenstellung der Voreinstellungsgrößen alle in Blockdiagrammen vorkommenden Typen von Symbolen, getrennt nach Knoten und Kanten.

Voreinstellungswerte für Knoten:

verwaltete Voreinstellungswerte	Defaultwertverwalter für Speicher	Defaultwertverwalter für Kanäle	Defaultwertverwalter für Akteure
Verrundungsradius	beliebig	beliebig	0
Breite	größer 2*Radius	fest 2*Radius	beliebig
Höhe	größer 2*Radius	fest 2*Radius	beliebig
Beim Erzeugen:			
möglicher Ankerpunkt	oben–links, oben–Mitte, Zentrum	oben–links, oben–Mitte, Zentrum	oben–links, oben–Mitte, Zentrum
möglicher Kontrollpunkt	oben–rechts, Mitte–rechts, unten–rechts	oben–rechts, Mitte–rechts, unten–rechts	oben–rechts, Mitte–rechts, unten–rechts
Initialtext:			
Komponententyp	beliebiges Mikrodokument	beliebiges Mikrodokument	beliebiges Mikrodokument
oberer Randabstand	beliebig	beliebig	beliebig
Anforderungstext	nicht definiert	beliebiger String	nicht definiert
Rückmeldungstext	nicht definiert	beliebiger String	nicht definiert
Strich / Füllung	beliebig entsprechend Dokumentpalette	beliebig entsprechend Dokumentpalette	beliebig entsprechend Dokumentpalette

Voreinstellungsgrößen für Kantenstücke und modifizierende Kanten
(gemeinsam verwaltet vom Kantendefaultverwalter):

	relevant für gerichtete Kantenstücke	relevant für ungerichtete Kantenstücke	relevant für modifizierende Kanten
Pfeile:			
Halber Winkel vorne	ja	——	ja
Halber Winkel hinten	ja	——	ja
Länge	ja	——	ja
Kantenstücke:			
Verrundungsradius	ja	ja	——
modifizierende Kanten:			
Bogenhöhe	——	——	ja
Bogenabstand	——	——	ja
Strich	ja	ja	ja



Legende:

- wird nach Anforderung ermittelt
- wird falls nötig online ermittelt
- Akteur zur Verwaltung einer Grafikgruppe
- Akteur zur Edition der Grafik des Planes
- Akteur für das planspez. Symbolrepertoire
- Akteur zur Herstellung einer syntaktisch fehlerfreien Planstruktur
- Akteure zum Bearbeiten unsichtbarer Attribute des Plans und seiner Kinder

Bild–A1

Entity Relationship Diagram **SPIKES Objektmodell für Editoren**	
Entwurf des Blockdiagramm–Editors	
Autor: Knöpfel / [4]	Datum: 19.6.95

Öffne SPIKES–Dokument

sonstige Benutzeraktion

Erzeugen oder Anwählen eines SPIKES–Planes

Öffne SPIKES–Plan

Grafikrahmen bearbeiten

nichts ist ausgewählt

Erzeuge eine Plan–komponente

letzten Editions-schritt aufheben (UNDO)

...

Kopie einfügen

Bearbeitung des Plans beenden: Schließen

eine oder mehrere Plan–komponenten auswählen

mindestens 1 Plankomponente ist ausgewählt

selektionsmengenabhängiges Repertoire

Bearbeiten

Löschen

Bewegen

Kopieren

Abwählen

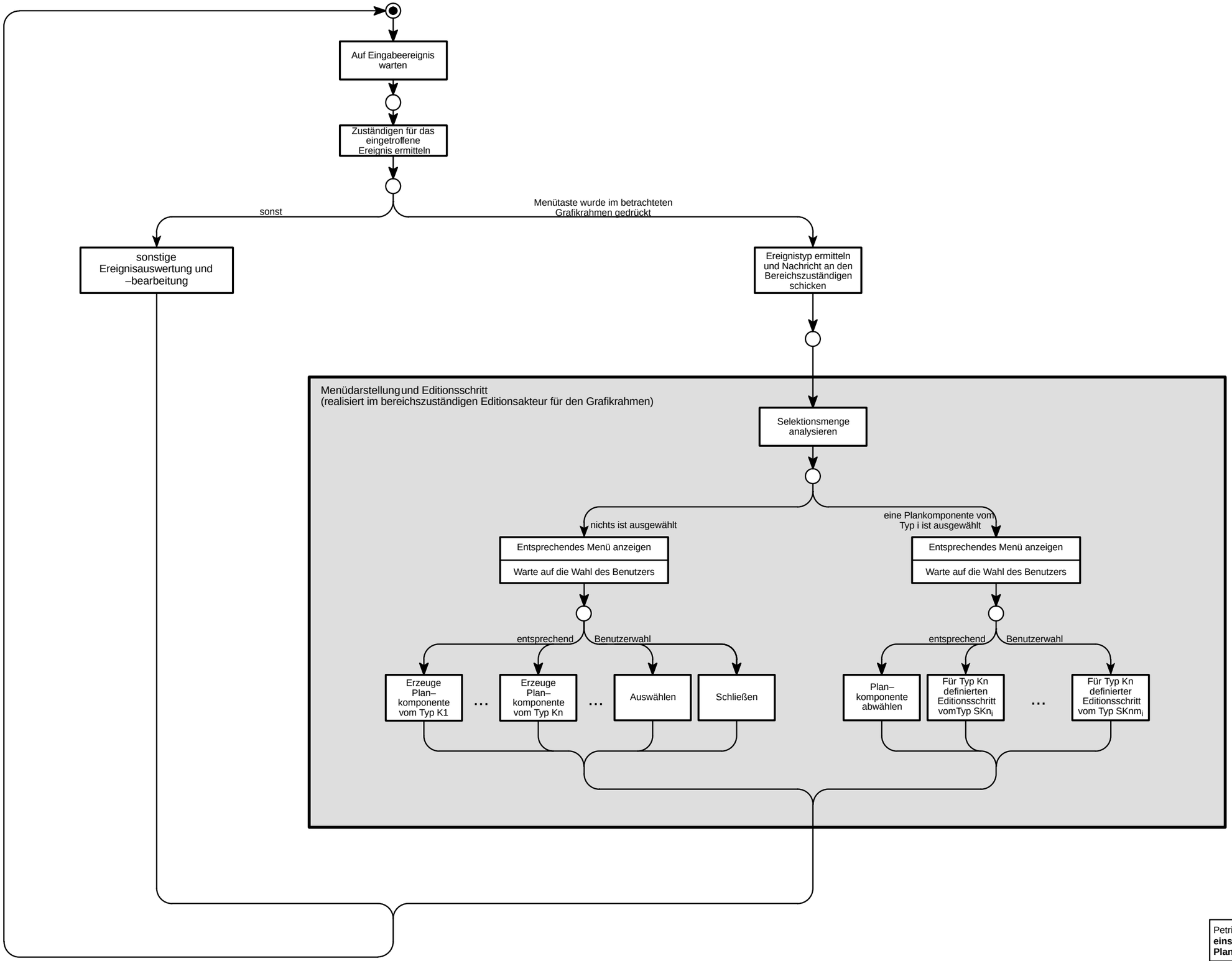
Schließe SPIKES–Dokument

Bild–A2

Petrinetz
einschrittige Edition auf oberstem Planniveau (Benutzersicht)

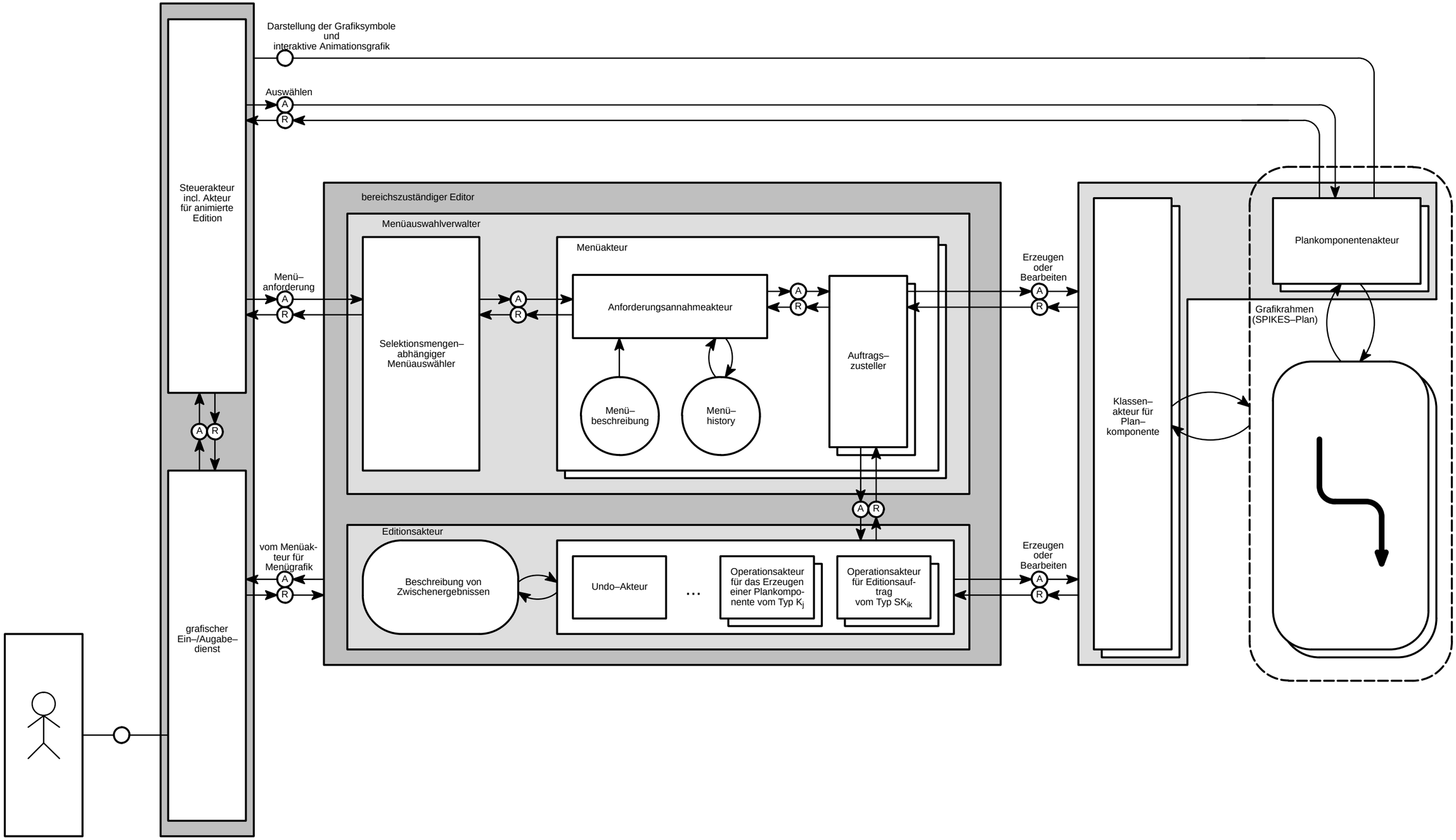
Entwurf des Blockdiagrammeditors

Autor: Knöpfel | Datum: 7/95



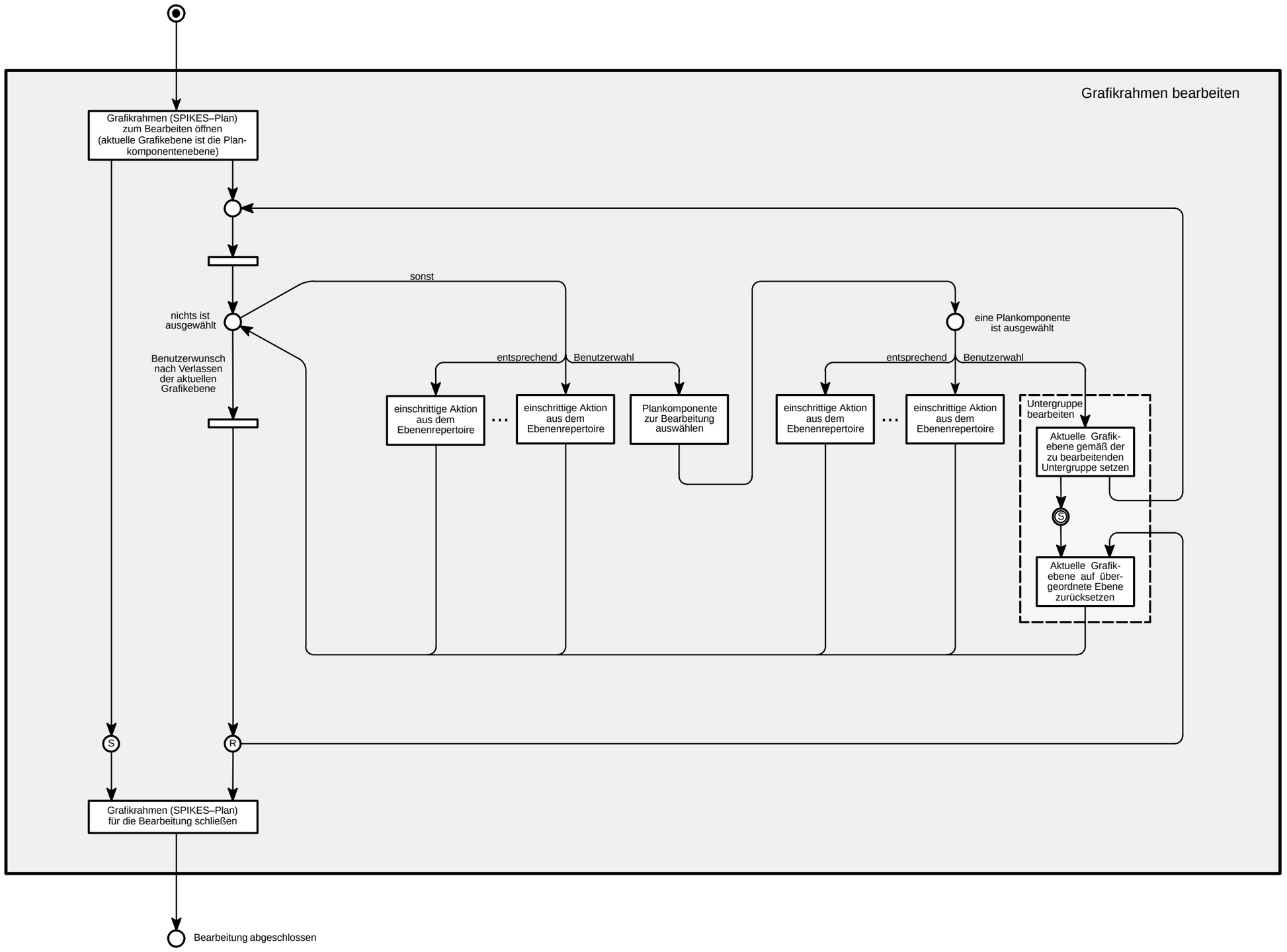
Bild–A3

Petrinetz **einschrittige Edition auf oberstem Planniveau – Systemsicht**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 7/95



Bild–A4

Aufbaubild **Editorsystem für einschrittige Edition von Plänen**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 7/95



Bild–A5

Petrinetz **Edition mit Untergruppenbearbeitung aus Benutzersicht**
Entwurf des Blockdiagrammeditors
Autor: Knöpfel | Datum: 7/95

Auf Eingabeereignis warten

Zuständigen für das eingetroffene Ereignis ermitteln

sonst

sonstige Ereignisauswertung und –bearbeitung

Menütaste wurde im betrachteten Grafikrahmen gedrückt

Menüanforderung an den Bereichszuständigen schicken

...

Aktionen des Editors für Operationen auf Plankomponentenebene

Selektionsmenge analysieren

nichts ist ausgewählt

eine Plankomponente vom Typ Ki ist ausgewählt

Entsprechendes Menü anzeigen

Warte auf die Wahl des Benutzers

...

Entsprechendes Menü anzeigen

Warte auf die Wahl des Benutzers

entsprechend

Benutzerwahl

einschrittige Aktion aus dem Ebenen–repertoire

...

einschrittige Aktion aus dem Ebenen–repertoire

Erzeuge Plan–komponente vom Typ Ki

einschrittige Aktion aus dem Ebenen-repertoire

...

einschrittige Aktion aus dem Ebenen–repertoire

Bearbeite Plan–komponente vom Typ K_j

...

Aktionen des Editors für Operationen zum Bearbeiten von Komponenten innerhalb der Plankomponente K_i

Selektionsmenge analysieren

nichts ist ausgewählt

eine Komponente der Untergruppe ist ausgewählt

Entsprechendes Menü anzeigen

Warte auf die Wahl des Benutzers

...

Entsprechendes Menü anzeigen

Warte auf die Wahl des Benutzers

entsprechend

Benutzerwahl

Schließen

...

einschrittige Aktion aus dem Ebenen–repertoire

Erzeuge komplexe Komponente vom Typ CK_j

einschrittige Aktion aus dem Ebenen–repertoire

...

einschrittige Aktion aus dem Ebenen–repertoire

Bearbeite komplexe Komponente vom Typ CK_j

...

...

...

Bild–A6

Petrinetz
Edition mit Untergruppenbearbeitung aus Systemsicht

Entwurf des Blockdiagrammeditors

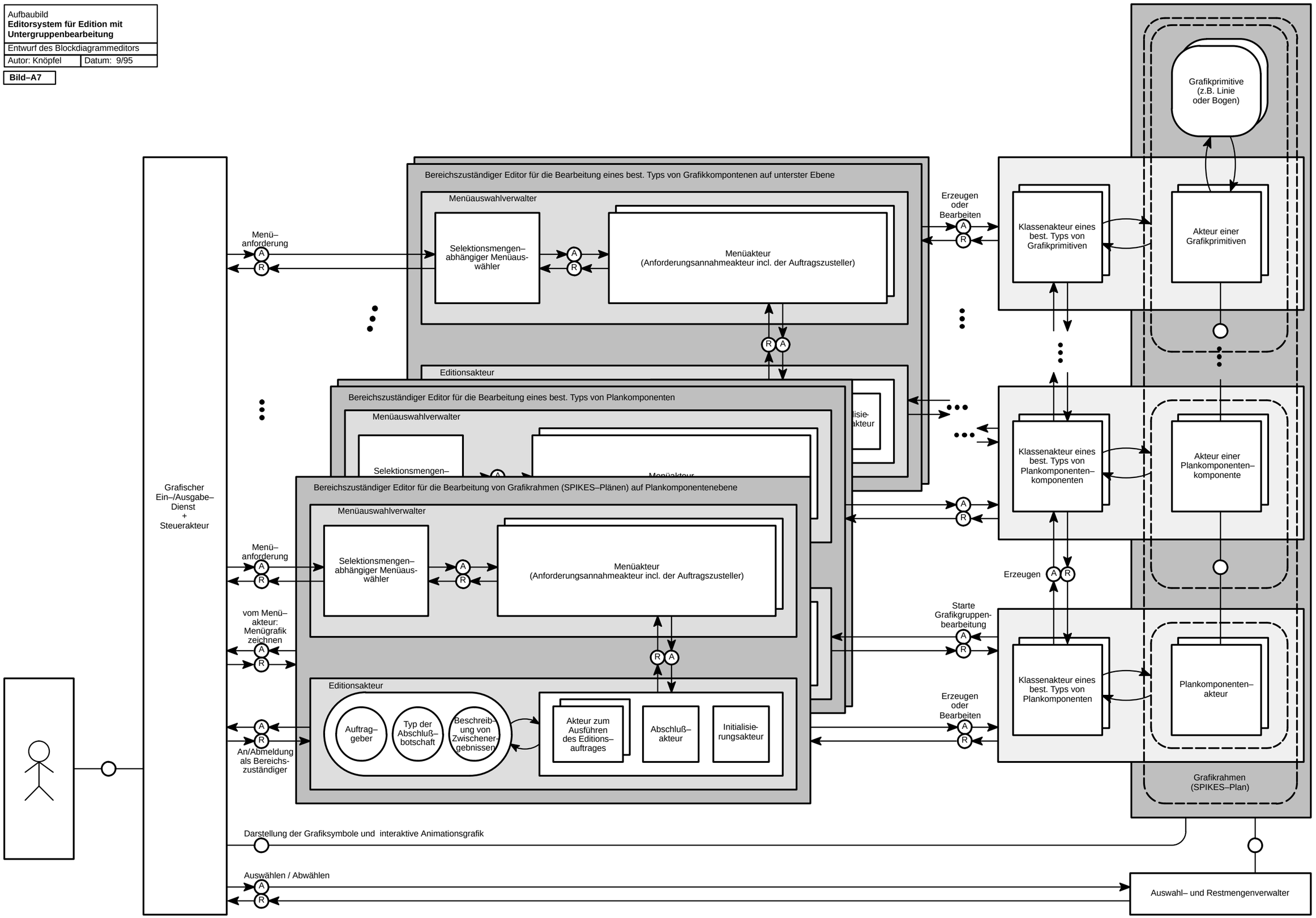
Autor: Knöpfel | Datum: 7/95

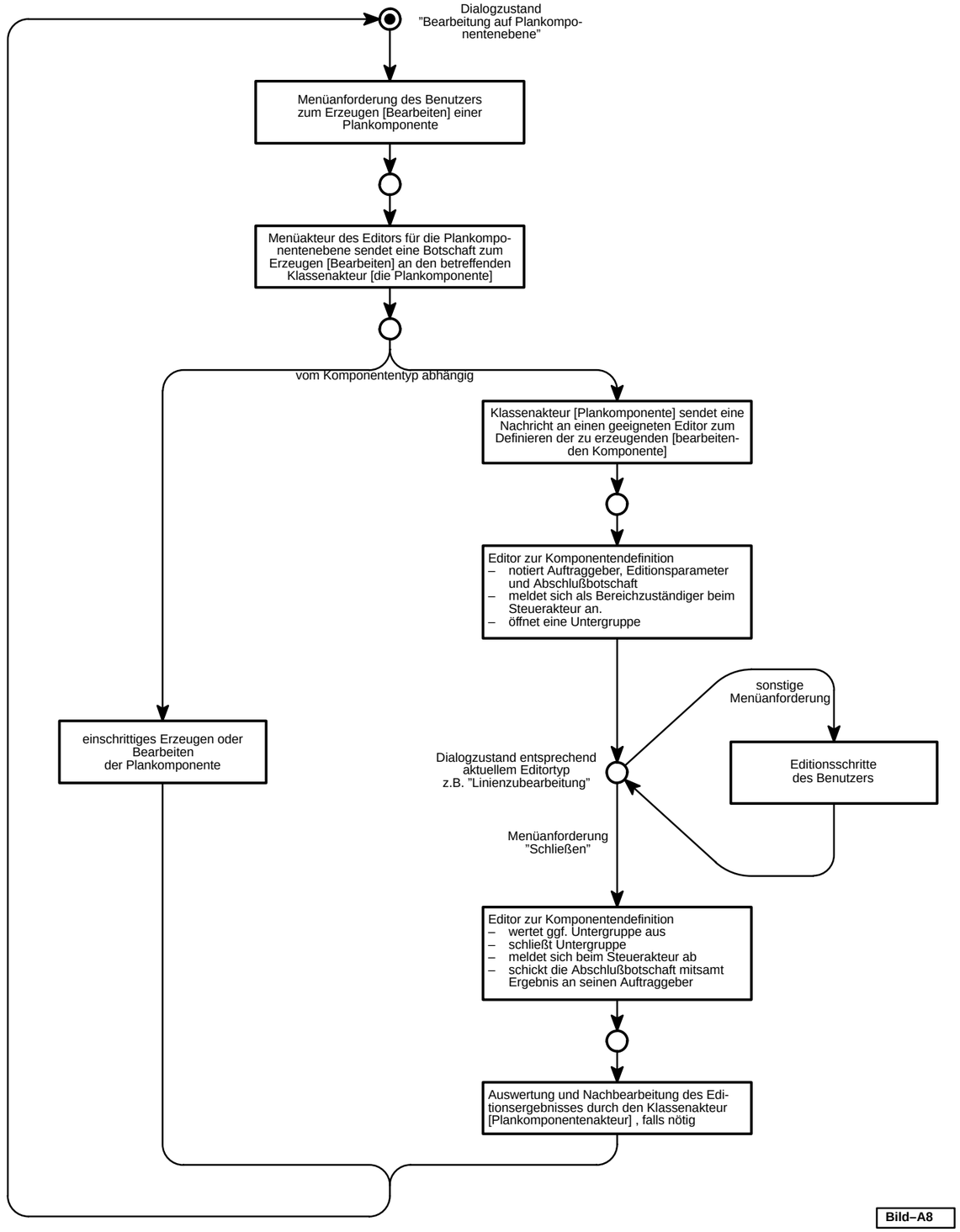
Aufbaubild
Editorsystem für Edition mit Untergruppenbearbeitung

Entwurf des Blockdiagrammeditors

Autor: Knöpfel | Datum: 9/95

Bild–A7





Bild–A8

Petrinetz
Mehrschrittiges Erzeugen und Bearbeiten aus Systemsicht

Entwurf des Blockdiagrammeditors

Autor: Knöpfel	Datum: 8/95

Aufbaubild
Implementierungsnahes Modell der SPIKES–Editoren
Entwurf des Blockdiagrammeditors
Autor: Knöpfel / [8] | Datum: 7/95

Bild–A9

Darstellung der Grafiksymbole und interaktive Animationsgrafik

Auswählen

Steuerakteur incl. Akteur für animierte Edition

Bereichszuständiger für den Grafikrahmen (Dokumenteditionsakteur)

Menüanforderungsbearbeiter

Menüauswahlverwalter

Selektions–mengen–abhängiger Menüauswähler

Menü–akteur

Menüauswahlverwalter

Selektions–mengen–abhängiger Menüauswähler

Menü–akteur

Menü–anforderung

vom Dialog–zustand abhän–giger Aus–wähler

Menügrafik darstellen

lies Tastatur–Tabelle

Dialogzustandsverwalter

Dialog–zustands–information

"PUSH–Akteur"

"POP–Akteur"

Tastatur–tabelle

grafischer Ein–/Augabe–dienst

Erzeugen oder Bearbeiten
(u.U. starte mehrschrittigen Editionsprozeß: *mid:start–edit* bzw. *mid:start–create*)

andere Editions–aufträge auf oberster Ebene

Planeditor

Undoakteur

Unter–gruppen–beauftragter

Akteur für Toplevel–editionsschritt

Verwaltungsdaten, wie z.B. Undoinformation

Erzeugen oder Bearbeiten

Selektionsebenenumschaltung zur Untergruppenbearbeitung, und Undo

Editions–aufträge zur Bearbeitung der Untergruppe

Editionsakteur

Initialisierungs–akteur

Abschluß–akteur

Akteur zum Ausführen des Editionsauftrages

Auftrag–geber

Typ der Abschluß–botschaft

Beschreib–ung von Zwischener–gebnissen

Starte Untergrup–penbear–beitung

Beende Untergrup–penbear–beitung

Erzeugen oder Bearbeiten

wechsle Dialogzustand

Defaultwert setzen

Klassen–akteur

Plankomponentenakteur

Grafikrahmen (SPIKES–Plan)

Defaultwerte lesen oder übernehmen

Defaultwertverwalter (Master)

Zugriffsakteur

Default–wert zur Erzeugung

Dialogzustand
”Bearbeitung auf Plan-
komponentenebene”

Menüanforderung
”Erzeuge gerichtetes Kantenstück”

Entsprechender Auftragszusteller schickt an den Klassenakteur für gerichtete Kantenstücke den Auftrag zum Erzeugen eines neuen Exemplares
(tell–class spk–dir–arc–comp–class mid:start–create)

start–create–Method des
Klassenakteurs für Kantenstücke

Erzeuge Editionsakteur für Linienzüge und initiiere Edition auf Basis einer leeren Punktliste
*(tell spk–polyline–edit–agent mid:start
:client self :close–message mid:close–create :points NIL)*

Interaktives Definieren einer neuen Punktliste
(vgl. Bild–A11)

close–create–Method des
Klassenakteurs für Kantenstücke

Entferne Editionsakteur für Linienzüge

Erzeuge einen Kantenstückakteur und beauftrage diesen basierend auf der Punktliste eine vollständige Kantenstückgrafik zu generieren

Versuche Anziehung auf Knoten
Kantenstück gemäß Defaultattributen anpassen
Kantenstück ggf. in Kantenbaum eingliedern

Fehlermeldung ausgeben
Kantenstückakteur löschen
Edition erneut anstoßen

Menüanforderung
”Bearbeite Linienzug des aufgewählten Kantenstückes”

Entsprechender Auftragszusteller schickt an den ausgewählten Kantenstückakteur den Auftrag eine interaktive Edition des Linienzuges der Basisgrafik einzuleiten
(tell selected–arc–component mid:start–edit)

start–edit–Method des
Kantenstückakteurs

Bearbeitung vorbereiten:
– für Undo beim Planeditor anmelden
– Kantenstück ggf. aus Baumstruktur ausgliedern
– aktuelle Kantenstückgrafik aus dem Plan entfernen

Erzeuge Editionsakteur für Linienzüge und initiiere Edition auf Basis der aktuellen Punktliste
*(tell spk–polyline–edit–agent mid:start
:client self :close–message mid:close–edit :points (...))*

Interaktives Bearbeiten der übergebenen Punktliste
(vgl. Bild–A11)

close–edit–Method des
Kantenstückakteurs

Entferne Editionsakteur für Linienzüge

Generiere basierend auf der Punktliste eine vollständige Kantenstückgrafik

Fehlermeldung ausgeben
Edition erneut anstoßen

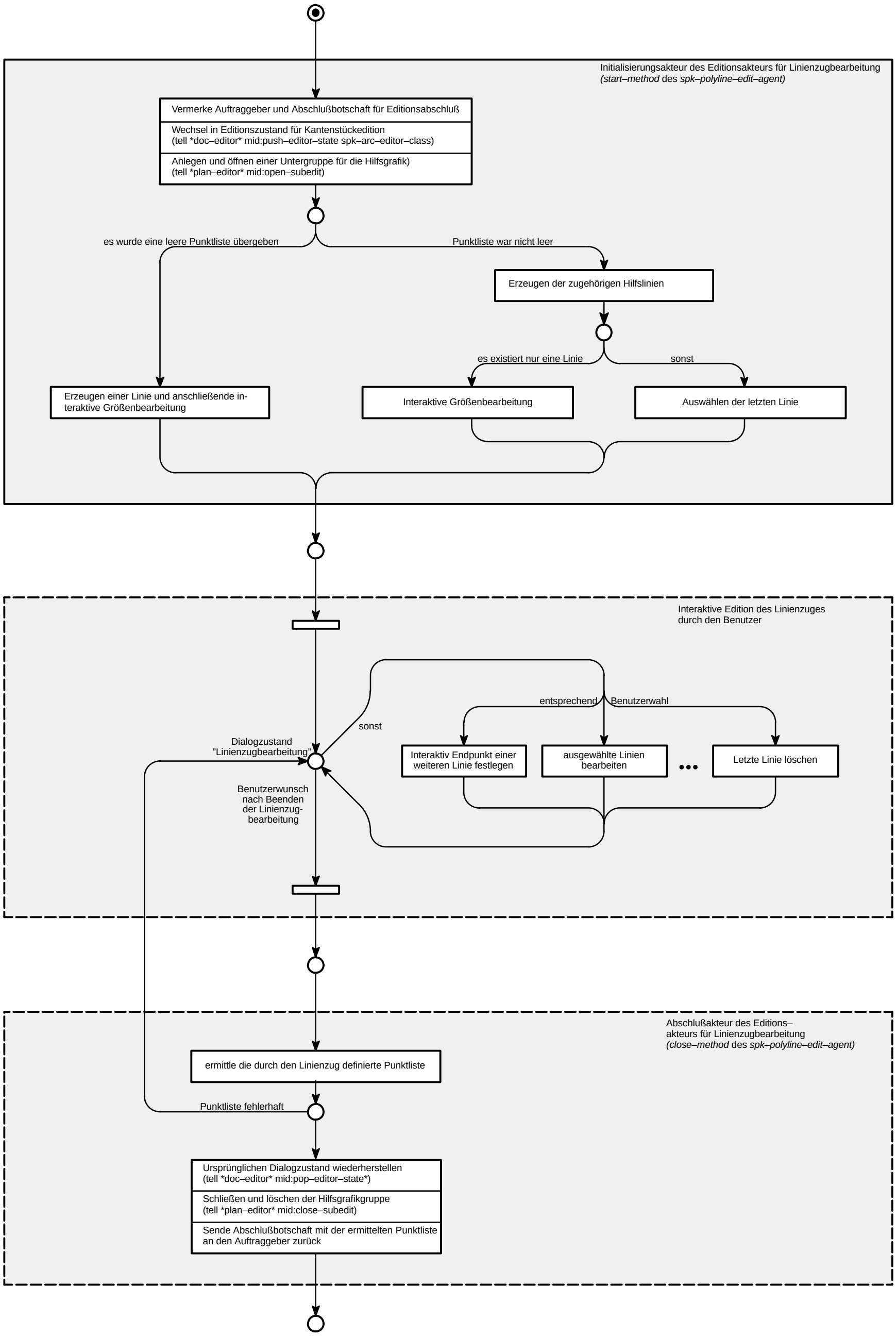
Versuche Anziehung auf Knoten
Kantenstückgrafik gemäß alter Attribute anpassen
Kantenstück ggf. in Kantenbaum eingliedern

Kantenstückakteur beim Bearbeiten bzw. Kantenstückklassenakteur beim Erzeugen

Editionsakteur für Linienzüge im Zusammenspiel mit dem Benutzer

Bild–A10

Petrinetz
Erzeugen und Bearbeiten von Kantenstücken – Überblick
Entwurf des Blockdiagrammeditors
Autor: Knöpfel | Datum: 9/95



Bild–A11

Petrinetz: **Editionsakteur für Linienzugbearbeitung**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 9/95

Petrinetz
Eingliedern von Kantenstückbäumen

Entwurf des Blockdiagrammeditors

Autor: Knöpfel | Datum: 8/95

Bild–A12

eine beliebige Menge von Plankomponenten wurde in den Plan eingefügt

Plankomponenten in den Plan eingliedern
(integrate–Methode des *spk–tree–managers)*

Prüfe, ob das nächste Element der Menge ein Kantenstück ist

sonst

Plankomponente ist ein Kantenstück

sonst

Bestimme ggf. die übrigen Kantenstücke des zugehörigen Kantenstückbaumes

Bestimme alle diesen Baum betreffenden Wurzelkontakte

Kantenstückbaum in den Plan eingliedern
(integrate–tree–Methode des spk–tree–managers)

Prüfe, ob der nächste Wurzelkontakt eine Verbindung der beiden beteiligten Bäume erlaubt

sonst

Verbindung ist zulässig

Vereinige die durch den Wurzelkontakt bezeichneten beiden Bäume

alle Wurzelkontakte bearbeitet

Alle neu eingefügten Plankomponenten wurden berücksichtigt.

alle Kantenstücke unter den neuen Plankomponenten sind in den ursprünglichen Plan eingegliedert

alle Wurzelkontakte zwischen dem einzugliedernden Baum und den Bäumen des ursprünglichen Planes bestimmen
*(tell *spk–tree–manager* mid:find–root–contacts)*

Nacheinander für alle Kantenstücke des einzugliedernden Baumes

alle in Frage kommenden berührenden Kantenstücke ermitteln
(tell spk–ar–comp mid:get–touching–arc–comps)

Bestimme alle Liniensegmente des Kantenstücklinienzuges

Nacheinander für alle Liniensegemente

Bestimme alle Plankomponenten, die von dem Liniensegment berührt werden und die von dem selben Kantenstücktyp sind, wie das Kantenstück zu dem das Liniensegment gehört

alle Linienelemente behandelt

alle Kantenstücke behandelt

Ermittelte Kantenstücke in einer Liste als Kontaktkandidaten vermerken

Bereinige die Kandidatenliste von Doppel–einträgen und Kantenstücken, die dem einzufügenden Baum selbst angehören

Bestimme das Wurzelkantenstück des einzugliedernden Baumes

Bestimme alle bestehenden Wurzelkontakte zwischen dem Wurzelkantenstück und den zuvor bestimmten Kandidaten lege für jeden gefundenen Kontakt eine Kontaktbeschreibung an

nacheinander für alle Kantenstücke der zuvor bestimmten Kandidatenliste

Bestimme alle bestehenden Wurzelkontakte zwischen dem Kantenstück und den angegebenen Testkandidaten, in diesem Falle den Ästen des einzugliedernden Baumes
(tell spk–arc–comp mid:find–root–contacts)

sonst

Kantenstück ist isoliert

Kantenstück ist Wurzel eines Baumes

Teste nacheinander alle Kantenstücke des neu einzugliedernden Baumes ob der STARTpunkt des Wurzelkantenstück eines dieser Kantentstücke berührt.

Wenn ja, vermerke in der Wurzelkontakteliste eine Beschreibung dieses Wurzelkontaktes
*(tell *tree–manager mid:add–root–contact ...)*

Bestimme mit hilfe der beim Wurzelkantenstück abgelegten Wachstumsinformation den Wurzelpunkt des Baumes

Teste nacheinander alle Kantenstücke des neu einzugliedernden Baumes ob der ENDpunkt des Wurzelkantenstück eines dieser Kantentstücke berührt.

Wenn ja, vermerke in der Wurzelkontakteliste eine Beschreibung dieses Wurzelkontaktes
*(tell *tree–manager mid:add–root–contact ...)*

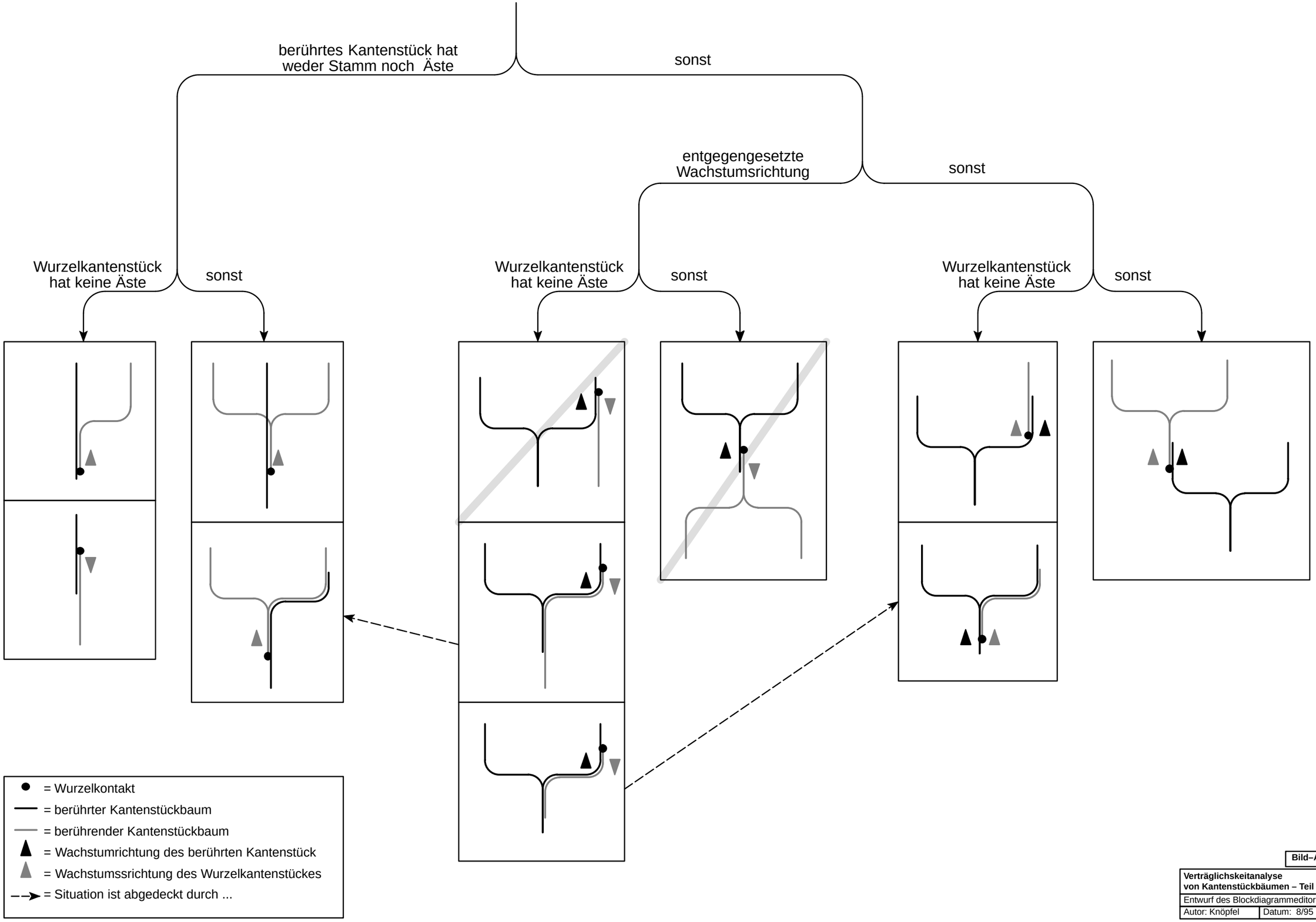
Teste nacheinander alle Kantenstücke des neu einzugliedernden Baumes ob der Wurzelpunkt des Wurzelkantenstück eines dieser Kantentstücke berührt.

Wenn ja, vermerke in der Wurzelkontakteliste eine Beschreibung dieses Wurzelkontaktes
*(tell *tree–manager mid:add–root–contact ...)*

alle Kandidaten behandelt

Petrinetz **Bestimmen von Wurzelkontakten**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95

Bild–A13



Bild–A14

Verträglichkeitanalyse von Kantenstückbäumen – Teil 1	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95

gemeinsamer Endpunkt von Wurzelkantenstück und berührtem Kantenstück

sonst

entgegengesetzter Verlauf

Verlauf beider Kantenstücke bezogen auf den Wurzelpunkt ist gleich

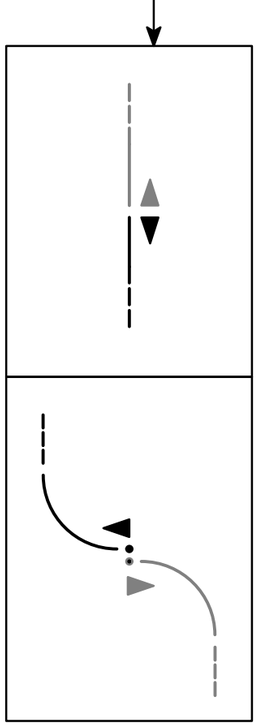
nur eines der beiden Liniensegmente ist zu einem Punkt entartet

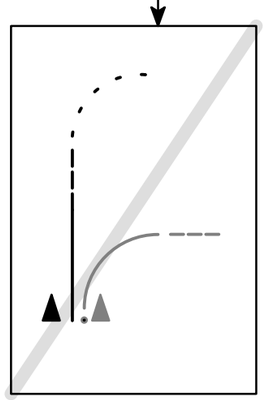
beiden Liniensegmente sind zu einem Punkt entartet

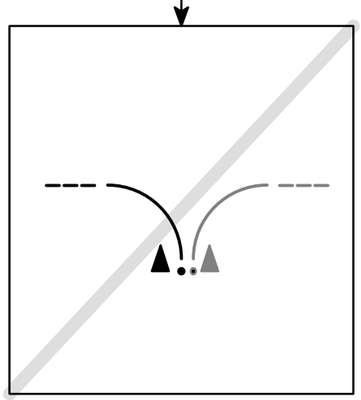
keines der beiden Liniensegmente ist zu einem Punkt entartet

sonst

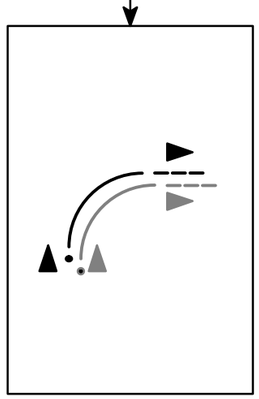
der Verlauf beider Kantenstücke ist zumindest bis nach der ersten Biegung identisch

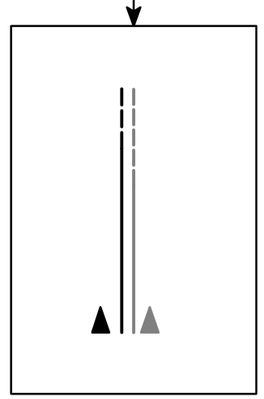


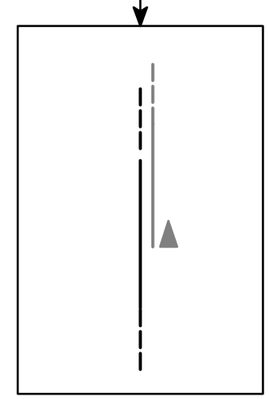




nicht zulässige Situationen







zulässige Situationen

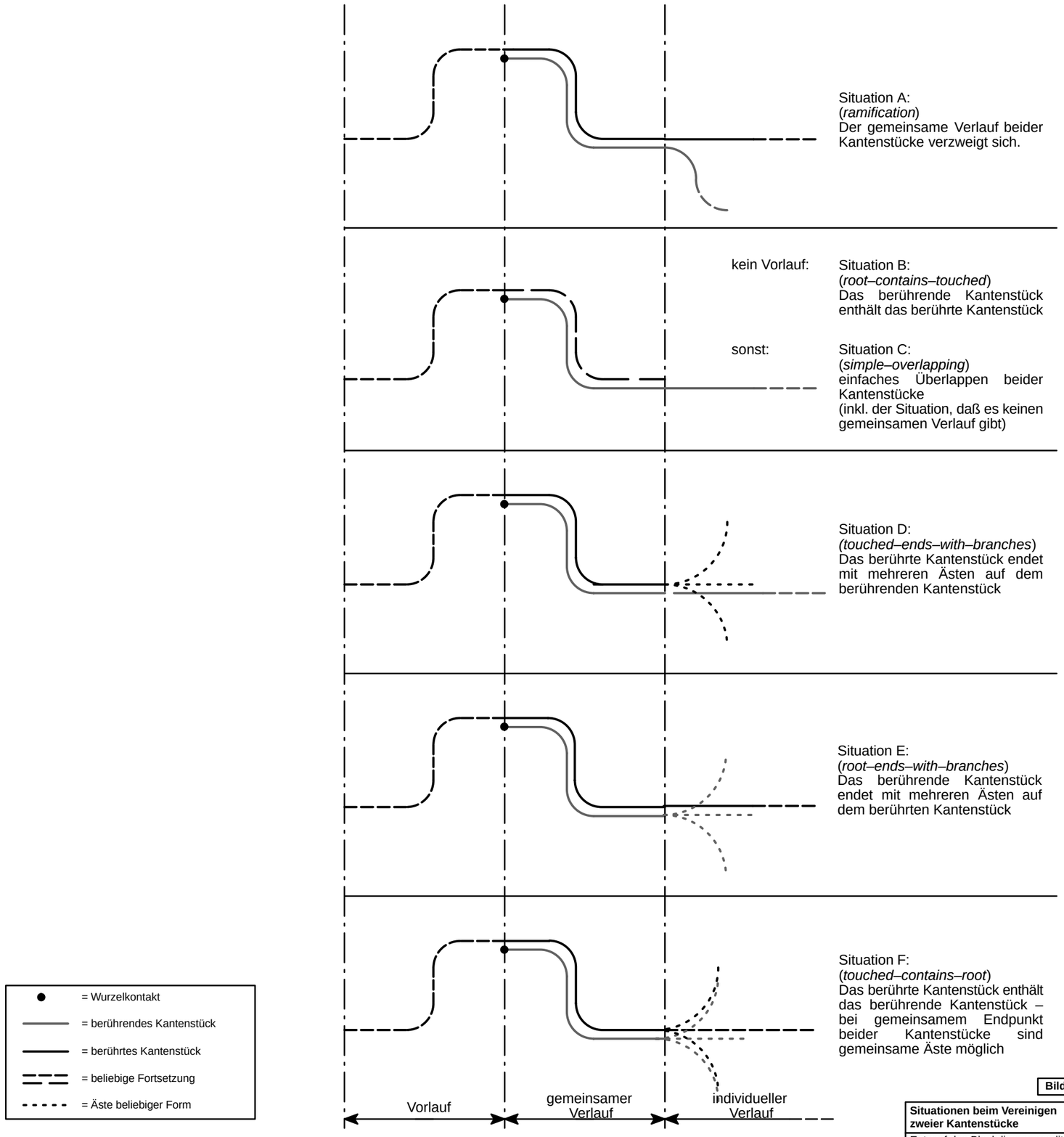
zulässige Situation:
gemeinsamer Startpunkt,
entgegengesetzter Verlauf

= Verlauf des berührten Kantenstück ausgehend vom Kontaktpunkt

= Verlauf des Wurzelkantenstückes ausgehend vom Kontaktpunkt

Bild–A15

Verträglichkeitanalyse von Kantenstückbäumen – Teil 2	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95



Bild–A16

Situationen beim Vereinigen zweier Kantenstücke	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95

= für den gemeinsamen Teil notierte Segmente
= für das berührte Kantenstück notierte Segmente
= für das berührende Kantenstück notierte Segmente
= beliebige möglicherweise leere Fortsetzung
= Trennstrich zwischen zwei Segmenten

Segmentliste und Punktliste für das berührte und das berührende Kantenstück, sowie für den gemeinsamen Teil anlegen

sonst

es exisitiert kein Vorlauf aus unberührten Segmenten

unberührte Segmente des Vorlaufes uebernehmen

Übernahme unberührter Segmente des Vorlaufers

sonst

gemeinsamer Startpunkt von berührtem und berührendem Segment

berührtes und berührendes Liniensegment stoßen stumpf aneinander

berührendes Liniensegment ist ein Punkt am Ende des berührten Liniensegments

Kontaktpunkt liegt irgendwo zwischen Start– und Endpunkt des berührten Segmentes

Übernahme des berührten Segmentes in den gemeinsamen Teil

Übernahme des berührten Segmentes, Löschen des Punktsegments

Übernahme vorderer Teil des berührten Segments bis Kontaktpunkt

keine Anpassung nötig

Anpasung sich überlappender Segmentabschnitte am Segmentende

sonst

es exisitiert kein gemeinsamer Verlauf grafisch identischer Segmente

Übernahme deckungsgleicher Segmente

Übernahme deckungsgleicher Segmente

sonst

eine der beiden oder beide Kantenstücksegmentlisten enthalten keine weiteren Einträge

sonst

die beiden Segmente des nächsten Abschnittes verlaufen in unterschiedliche Richtungen

berührtes Segment endet vor berührendem Segment

berührndes Segment endet vor berührtem Segment

berührndes Liniensegment hat die Länge NULL

berührndes Liniensegment ist eine echte Linie

berührtes Liniensegment hat die Länge NULL

berührtes Liniensegment ist eine echte Linie

berührtes Punkt–segment entfernen

Übernahme in den gemeinsamen Teil bis Überdeckungsende

berührendes Punkt–segment entfernen

Übernahme in den gemeinsamen Teil bis Überdeckungsende

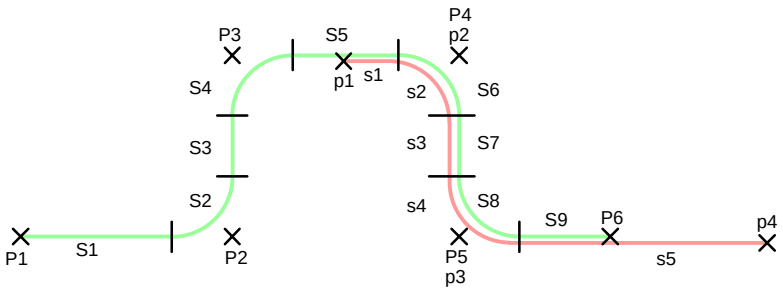
keine Anpassung nötig

Anpasung sich überlappender Segmentabschnitte am Segmentanfang

Bild–A17

Petrinetz: **Vereinigen des gemeinsamen Teiles zweier Kantenstücke**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95

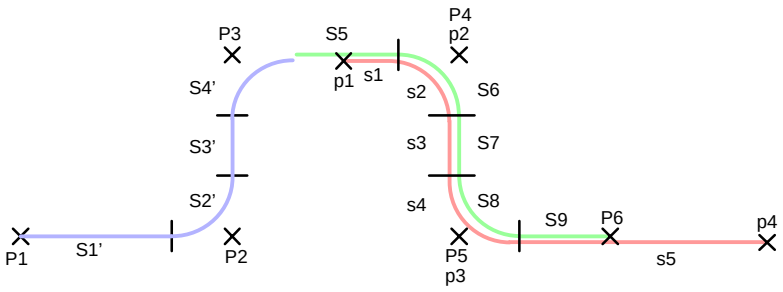
nach Initialisierung der Punkt und Segmentlisten



Segmentliste berührtes Kantenstück: (S1, S2, S3, S4, S5, S6, S7, S8, S9)
Segmentliste berührendes Kantenstück: (s1, s2, s3, s4, s5)
Segmentliste gemeinsamer Teil ()

Punktliste berührtes Kantenstück (P1, P2, P3, P4, P5, P6)
Punktliste berührendes Kantenstück (p1, p2, p3, p4)
Punktliste gemeinsamer Teil ()

nach Übernahme unberührter Segmente des Vorlaufes

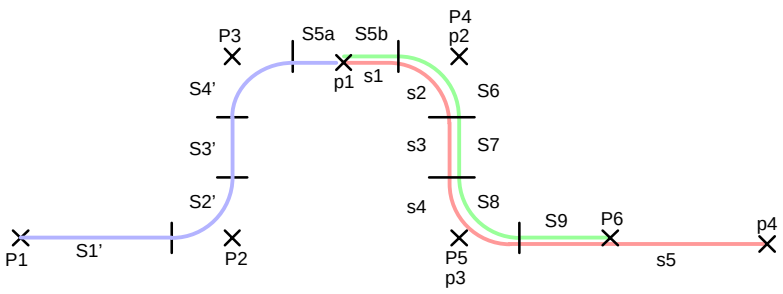


Segmentliste berührtes Kantenstück: (S5, S6, S7, S8, S9)
Segmentliste berührendes Kantenstück: (s1, s2, s3, s4, s5)
Segmentliste gemeinsamer Teil (S1', S2', S3', S4')

Punktliste berührtes Kantenstück (P3, P4, P5, P6)
Punktliste berührendes Kantenstück (p1, p2, p3, p4)
Punktliste gemeinsamer Teil (P1, P2,)

Anmerkung: (X' = Kopie(X))

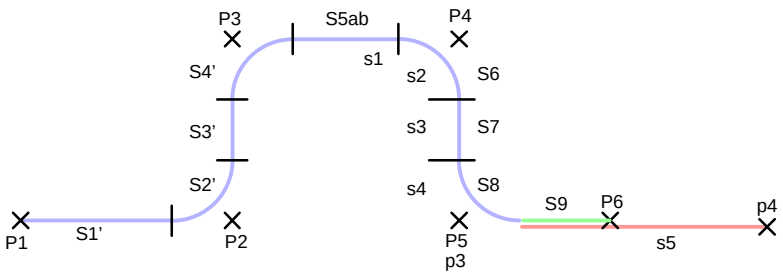
nach Anpasung sich überlappender Segmentabschnitte am Segmentende



Segmentliste berührtes Kantenstück: (S5a, S6, S7, S8, S9)
Segmentliste berührendes Kantenstück: (s1, s2, s3, s4, s5)
Segmentliste gemeinsamer Teil (S1', S2', S3', S4',S5b))

Punktliste berührtes Kantenstück (P4, P5, P6)
Punktliste berührendes Kantenstück (p1, p2, p3, p4)
Punktliste gemeinsamer Teil (P1, P2, P3)

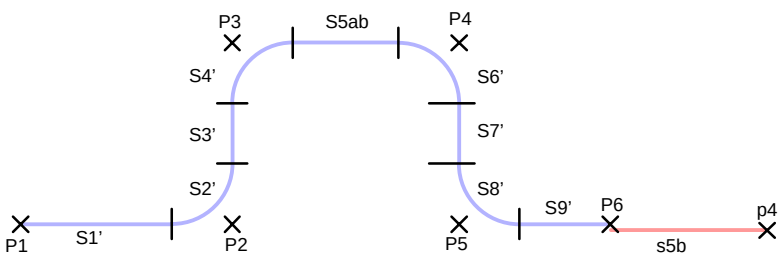
nach Übernahme deckungsgleicher Segmente



Segmentliste berührtes Kantenstück: (S9)
Segmentliste berührendes Kantenstück: (s5)
Segmentliste gemeinsamer Teil (S1', S2', S3', S4',S5ab, S6', S7', S8')

Punktliste berührtes Kantenstück (P5, P6)
Punktliste berührendes Kantenstück (p3, p4)
Punktliste gemeinsamer Teil (P1, P2, P3, P4 (=p2))

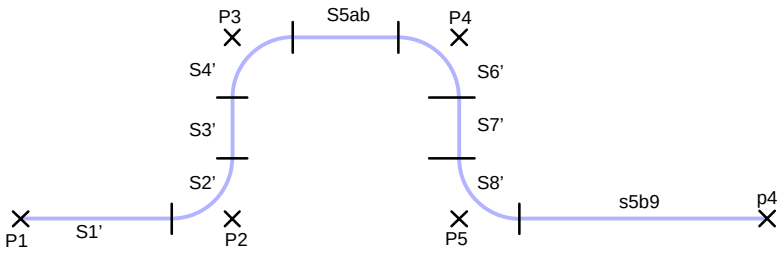
nach Anpasung sich überlappender Segmentabschnitte am Segmentanfang



Segmentliste berührtes Kantenstück: ()
Segmentliste berührendes Kantenstück: (s5b)
Segmentliste gemeinsamer Teil (S1', S2', S3', S4',S5ab, S6', S7', S8', S9')

Punktliste berührtes Kantenstück (P6)
Punktliste berührendes Kantenstück (p4)
Punktliste gemeinsamer Teil (P1, P2, P3, P4, P5 (=p3))

nach individueller Behandlung (hier: Übernahme des verbleibenden Linienzuges)



Segmentliste gemeinsamer Teil (S1', S2', S3', S4',S5', S6', S7', S8', s5b9)
Punktliste gemeinsamer Teil (P1, P2, P3, P4, P5, p4)

Zuordnung der Segmente entsprechend der Segmentlisten:
- Segmente berührtes Kantenstück
- Segmente berührendes Kantenstück
- Segmente gemeinsamer Teil

Bild–A18

Vereinigungbeispiel zweier sich überlappender Kantenstücke	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95

gegeben: Liste mit Verweisen auf zu lösende Plankomponenten

Referenzen zwischen zurückbleibenden und zu lösenden Bäumen lösen

Nimm die nächste noch unberücksichtigte zu lösende Plankomponente

sonst

sonst

die Plankomponente ist ein Kantenstück

Vermerke die Plankomponente als berücksichtigt und trage sie in der Liste der gelösten Plankomponenten ein

Für alle Kantenstücke des Baumes, den dieses Kantenstück spezifiziert:

– lösche alle Stamm–Ast–Referenzen zwischen zurückbleibenden und zu lösenden Kantenstücken
– vermerke alle auftretenden gelösten und zurück–bleibenden Wurzelkantenstücke
– vermerke alle Kantenstücke als berücksichtigt

(Verfeinerung siehe Bild–A20)

alle zu lösenden Plankomponenten berücksichtigt

Struktur der entstandenen Bäume in Minimalform überführen

Nimm das nächste Wurzelkantenstück aus der Liste zu lösender Wurzelkantenstücke

Passe gegebenenfalls dessen von der Lage im Baum abgängigen Attribute an

Erkläre das Wurzelkantenstück zum aktuellen Kantenstück

sonst

Überführe den Kantenstückbaum, den das aktuelle Kantenstück spezifiziert,
– in einen Baum korrekter Minimalformstruktur und
– vermerke alle Kantenstücke der endgültigen Baumstruktur

(Verfeinerung siehe Bild–A21)

alle vermerkten zu lösenden Wurzelkantenstücke bearbeitet

Übernimm die Elemente der endgültigen Baumstruktur in die Liste der gelösten Plankomponenten

Nimm das nächste Wurzelkantenstück aus der Liste zurückbleibender Wurzelkantenstücke

Passe gegebenenfalls dessen von der Lage im Baum abgängigen Attribute an

Erkläre das Wurzelkantenstück zum aktuellen Kantenstück

sonst

Überführe den Kantenstückbaum, den das aktuelle Kantenstück spezifiziert,
– in einen Baum korrekter Minimalformstruktur und
– vermerke alle Kantenstücke der endgültigen Baumstruktur

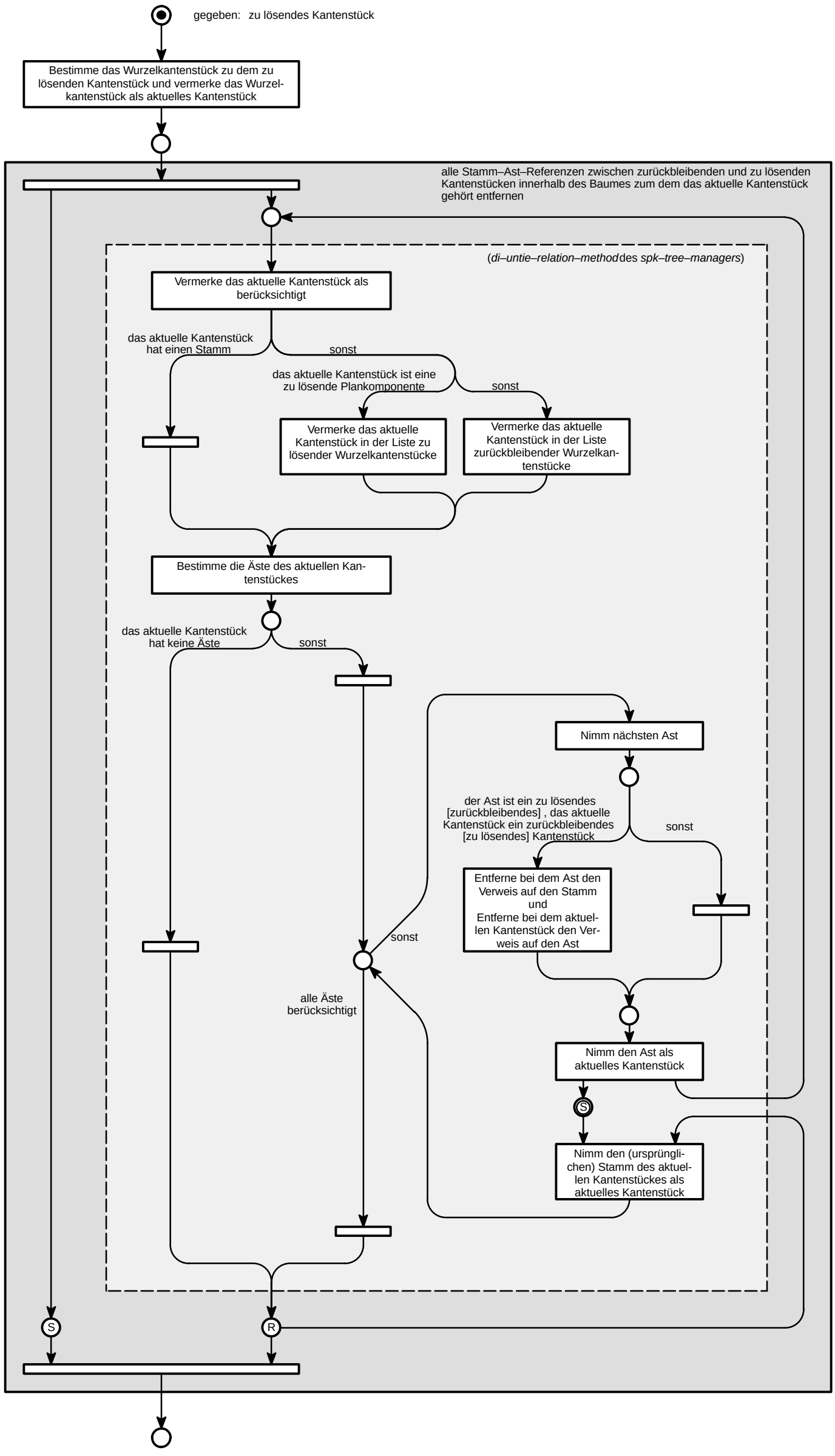
(Verfeinerung siehe Bild–A21)

alle vermerkten zurückbleibenden Wurzelkantenstücke bearbeitet

Wähle zur anschließenden Bearbeitung alle gelösten Plankomponenten aus

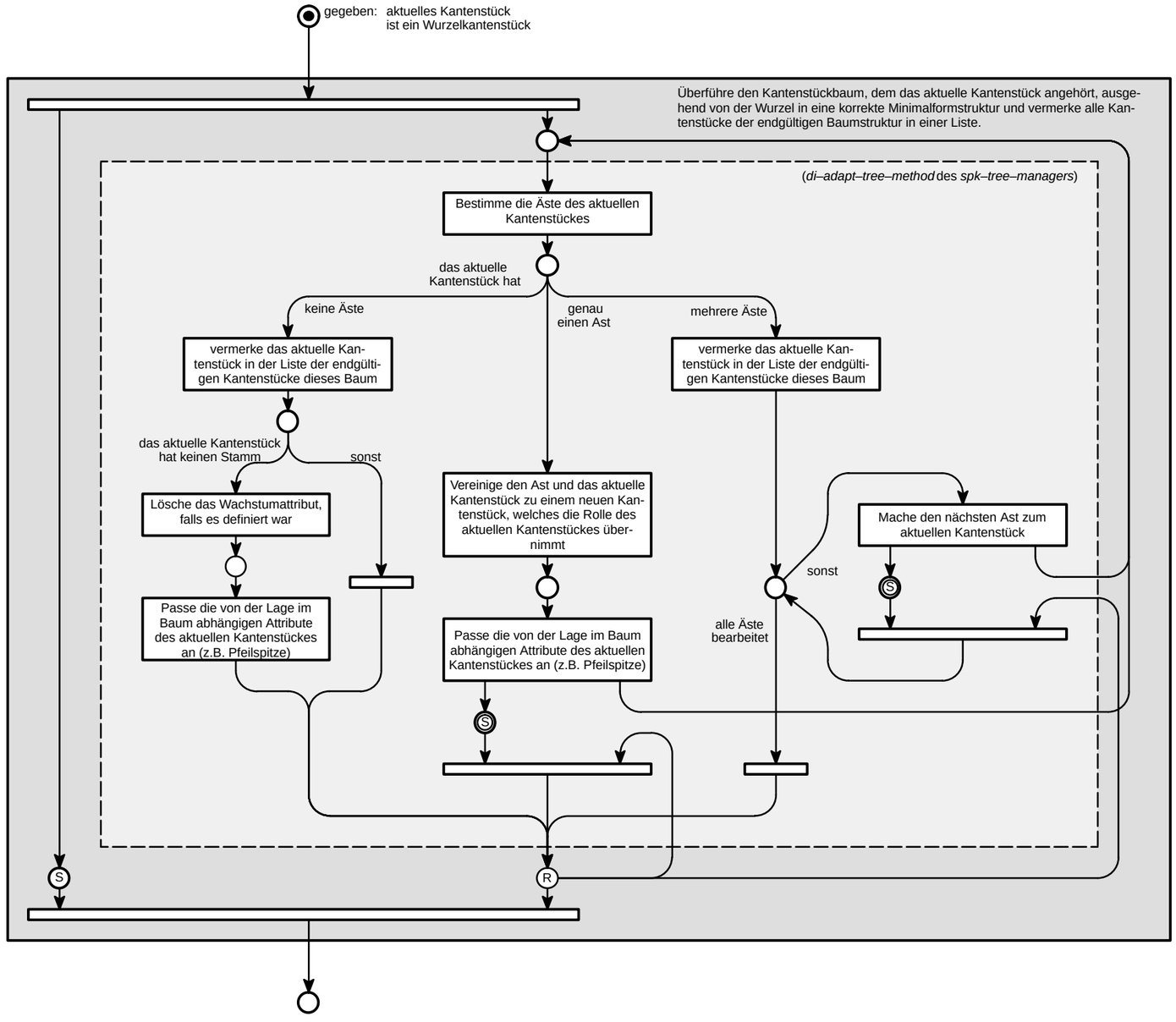
Bild–A19

Petrinetz: **Heraustrennen von Kantenstücken Überblick**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95



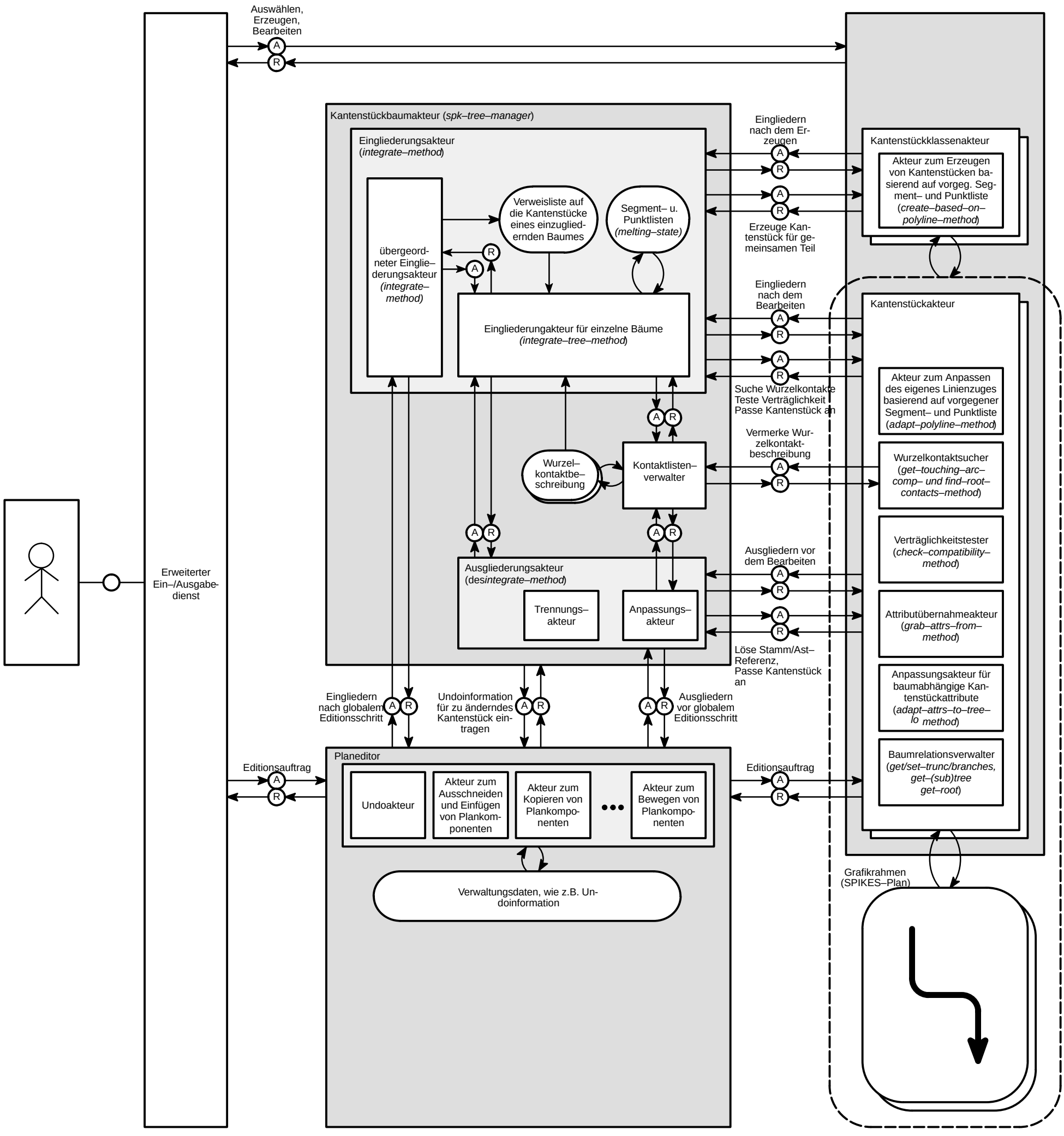
Bild–A20

Petrinetz: **Heraustrennen von Kantenstücken Trennen der Referenzen**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95



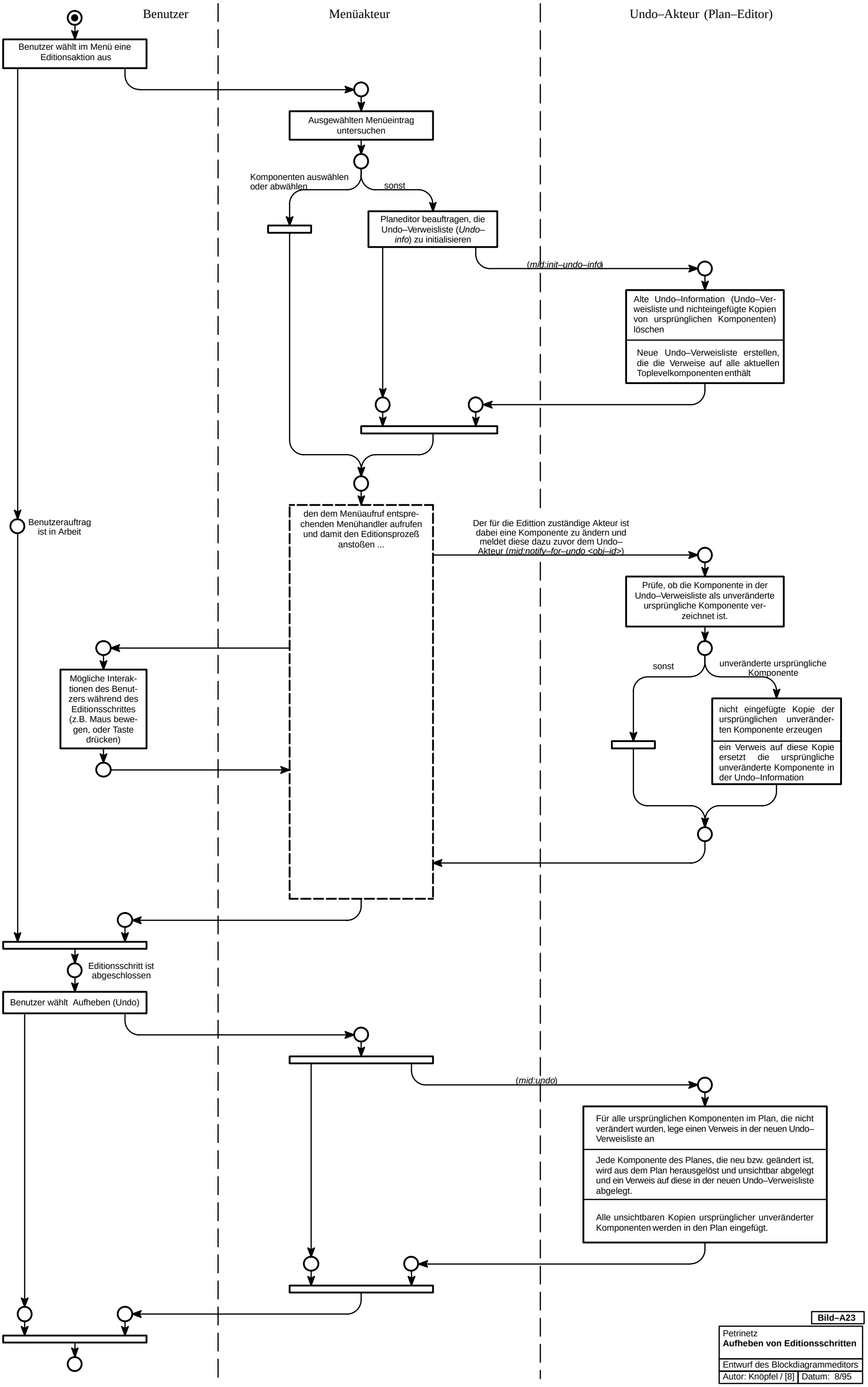
Bild–A21

Petrinetz: **Heraustrennen von Kantenstücken Baumstrukturanpassung**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95



Bild–A22

Aufbaubild **Der Kantenstückbaumakteur im Editor–System**
Entwurf des Blockdiagrammeditors
Autor: Knöpfel — Datum: 7/95

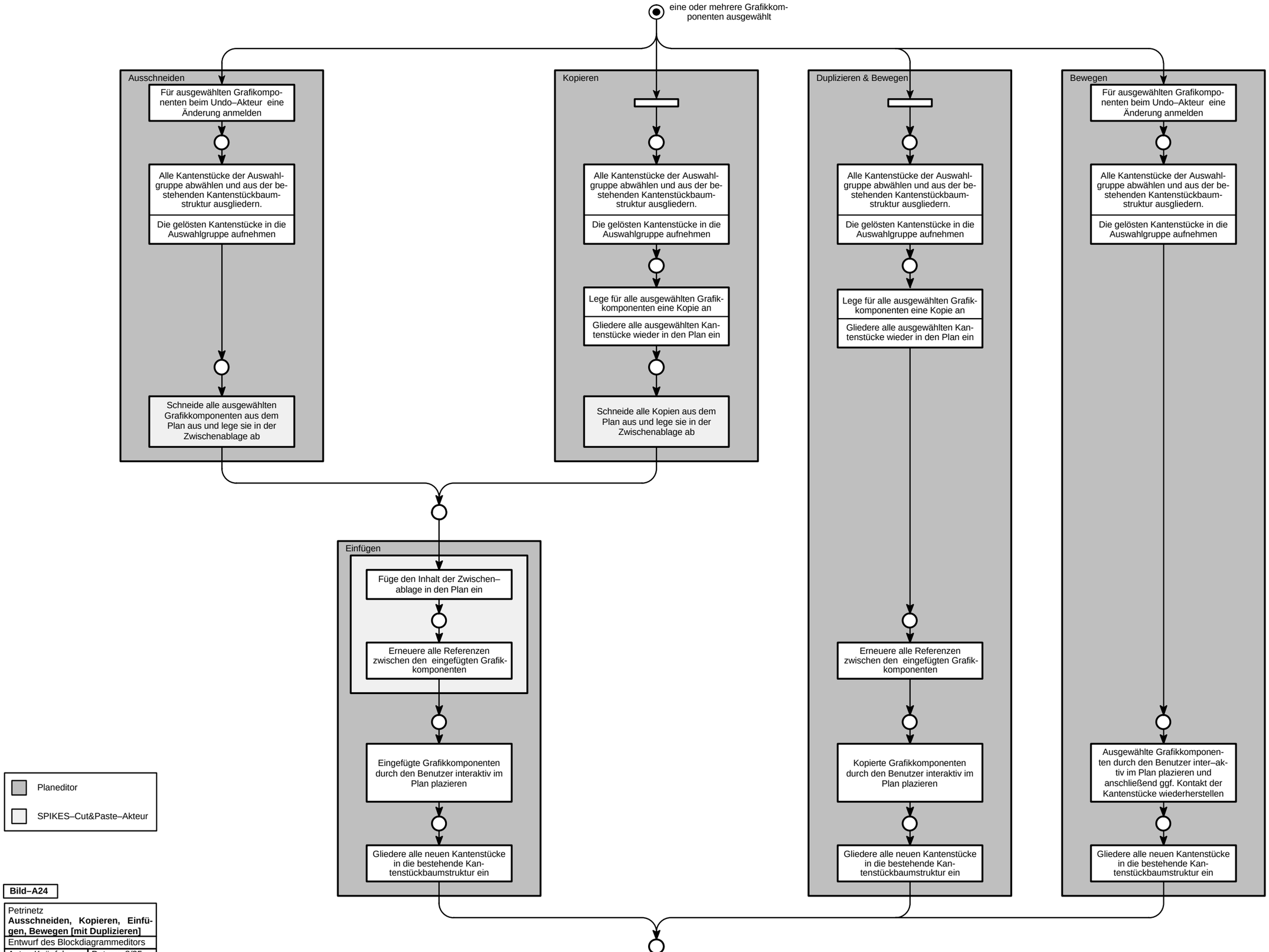


Bild–A23

Petrinetz
Aufheben von Editionsschritten

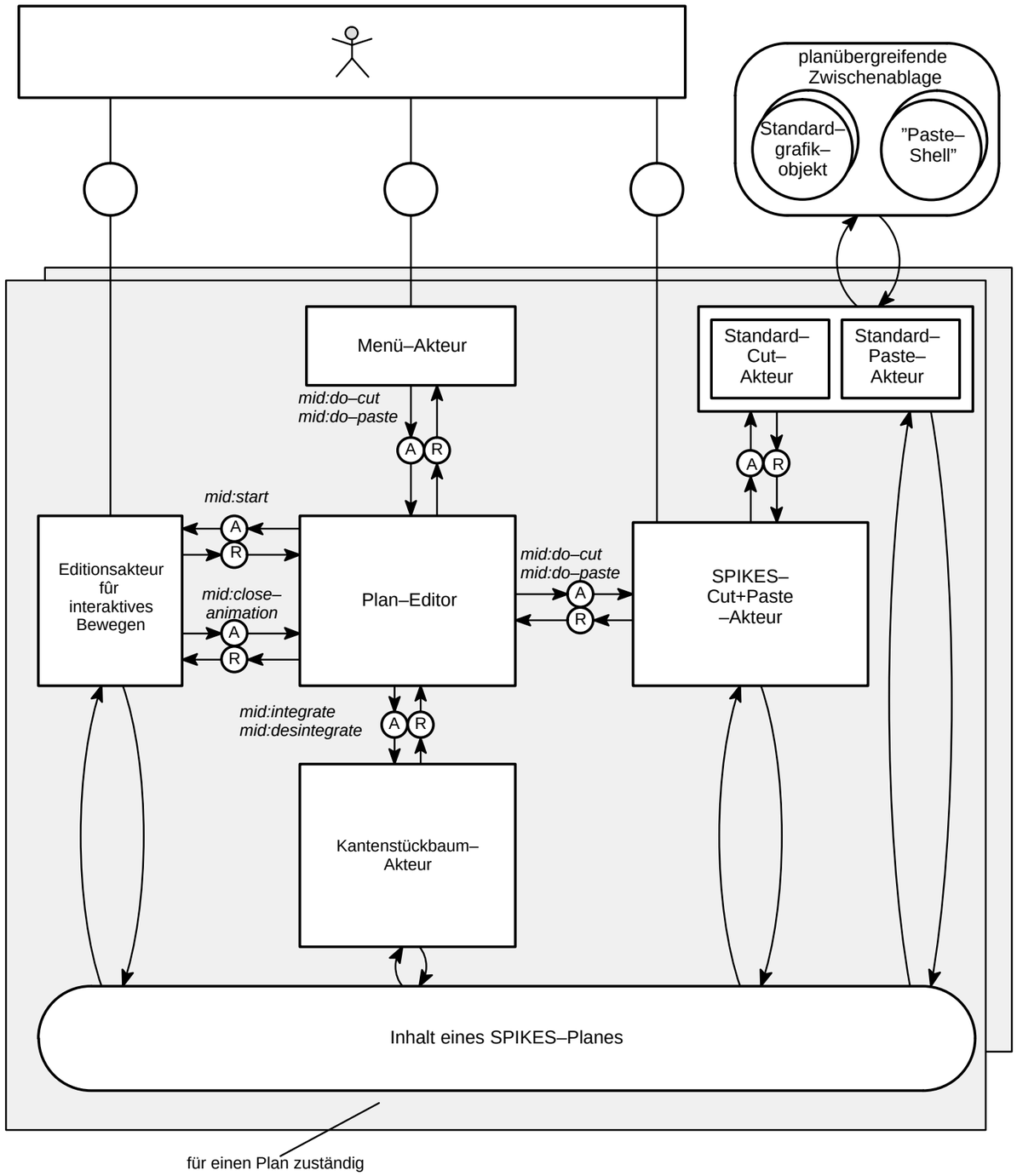
Entwurf des Blockdiagrammeditors

Autor: Knöpfel / [8] | Datum: 8/95



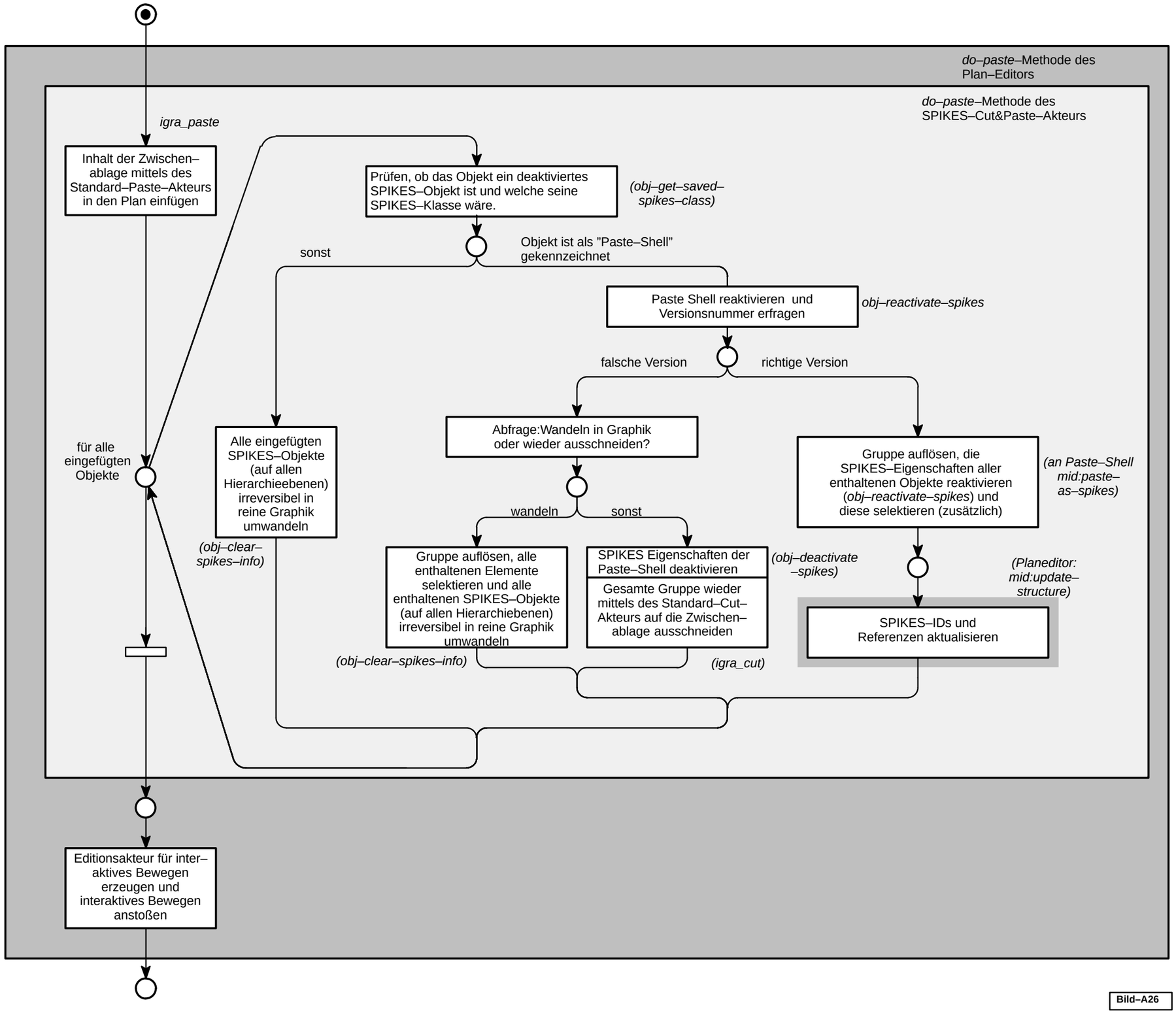
Bild–A24

Petrinetz **Ausschneiden, Kopieren, Einfügen, Bewegen [mit Duplizieren]**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 8/95



Bild–A25

Aufbaubild **Der SPIKES–Cut&Paste–Akteur**	
Entwurf des Blockdiagrammeditors	
Autor: Kleis (aus [8])	Datum: 6/95



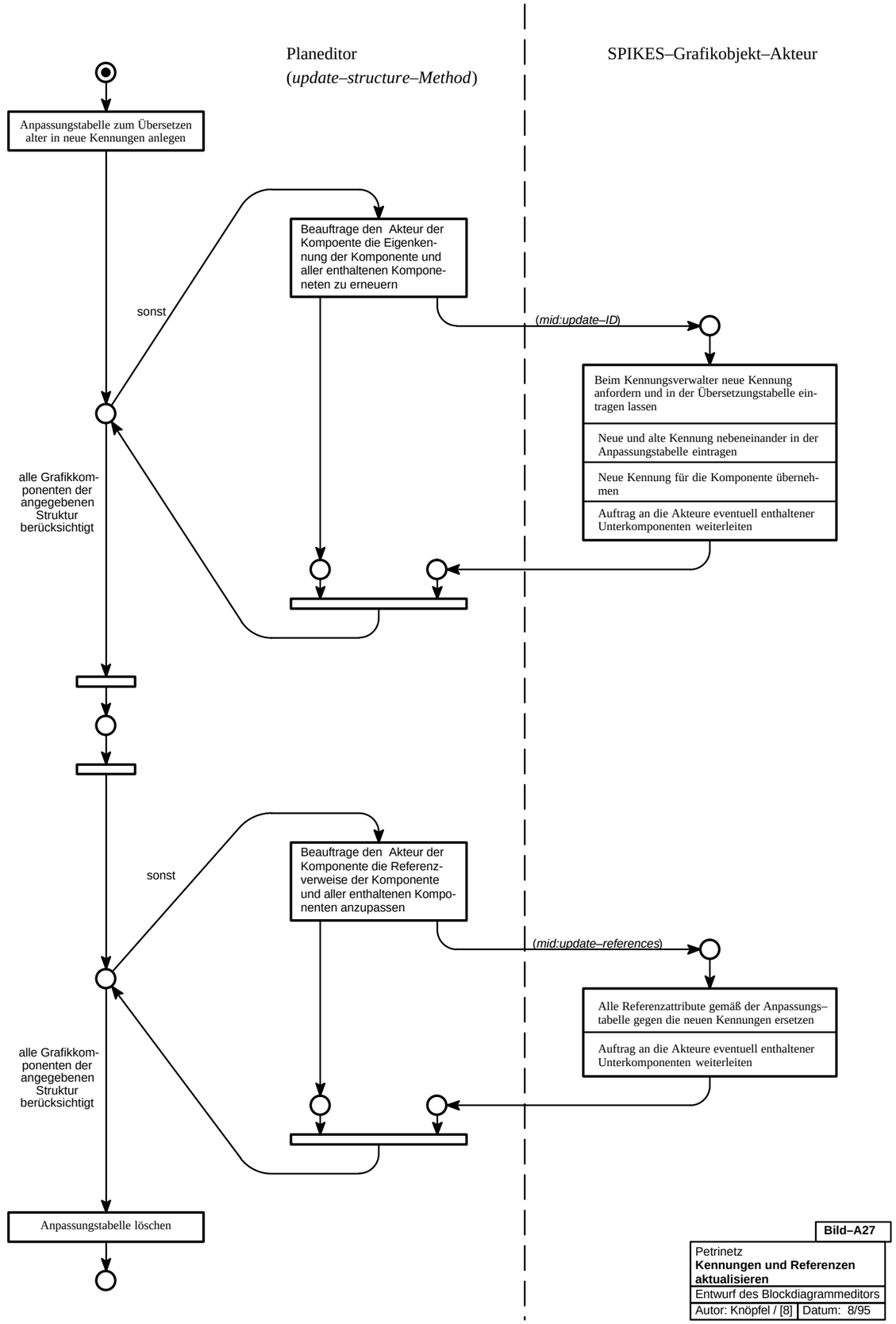
Bild–A26

Petrinetz
Einfügen aus der Zwischenablage

Entwurf des Blockdiagrammeditors

Autor: Kleis (aus [8]) | Datum: 6/95

(leicht modifiziertes Originalbild)

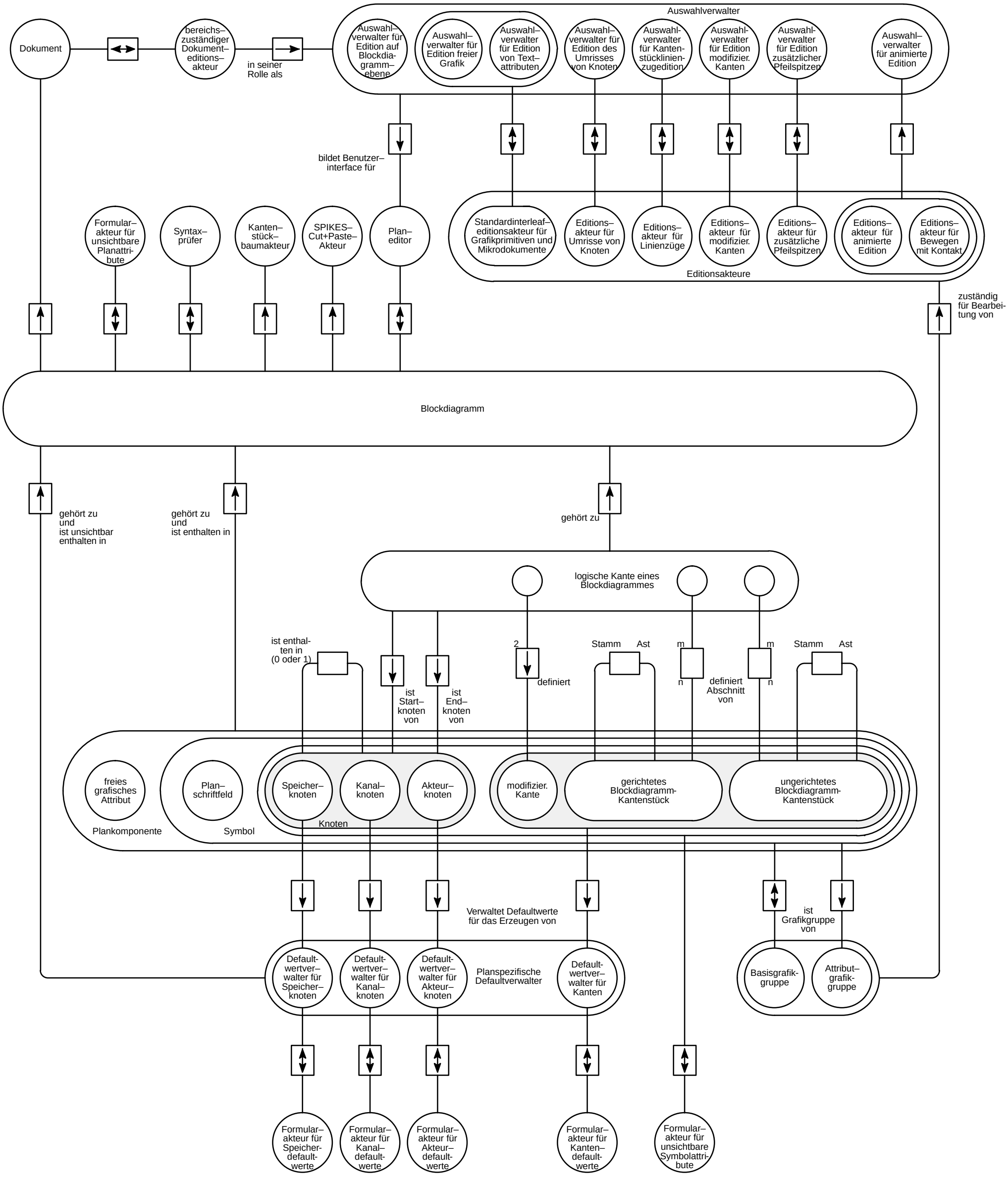


Bild–A27

Petrinetz
Kennungen und Referenzen aktualisieren

Entwurf des Blockdiagrammeditors

Autor: Knöpfel / [8] | Datum: 8/95



Bild–A28

Entity Relationship Diagram **Objektmodell des Blockdiagramm–Editors**	
Entwurf des Blockdiagramm–Editors	
Autor: Knöpfel	Datum: 19.6.95

Dialogzustand
”Bearbeitung auf Plan-
komponentenebene”

Menüanforderung
”Erzeuge modifizierende Kantenstück”

Entsprechender Auftragszusteller schickt an den Klassenakteur für modifizierende Kanten den Auftrag zum Erzeugen eines neuen Exemplares
(tell–class bde–mod–arc–class mid:start–create :direction ...)

start–create–Method des
Klassenakteurs für mod. Kanten

Erzeuge Editionsakteur für modifizierende Kanten und initiiere Edition auf Basis der Defaultwerte für Bogenabstand und Höhe
(tell bde–mod–arc–edit–agent mid:start
:client self :close–message mid:close–create ...)

Interaktives Definieren von mindestens Start– und Endpunkt
ggf. auch Bogenhöhe und –abstand (vgl. Bild–A30)

close–create–Method des
Klassenakteurs für mod. Kanten

Entferne Editionsakteur für modifizierende Kanten

Erzeuge einen Kantenstückakteur und beauftrage diesen basierend auf der geometrischen Beschreibung die Grafik einer modifizierenden Kante zu generieren

Passe die Farbe und Linienparameter der erzeugten Grafik an die planspezifischen Defaultwerte für modifizierende Kanten an

Menüanforderung
”Bearbeite die Bögen der ausgewählten modifizierenden Kante”

Entsprechender Auftragszusteller schickt an den ausgewählten Kantenakteur den Auftrag eine interaktive Edition der Bögen einzuleiten
(tell selected–bde–mod–arc mid:start–edit)

start–edit–Method des
Kantenakteurs

Bearbeitung vorbereiten:
– für Undo beim Planeditor anmelden
– aktuelle Symbolgrafik aus dem Plan entfernen

Erzeuge Editionsakteur für modifizierende Kanten und initiiere Edition auf Basis der aktuellen geometrischen Parameter
(tell spk–mod–arc–edit–agent mid:start
:client self :close–message mid:close–edit ...)

Interaktives Bearbeiten der geometrischen Parameter
(vgl. Bild–A30)

close–edit–Method des
Kantenakteurs

Entferne Editionsakteur für modifizierende Kanten

Generiere basierend auf der ermittelten geometrischen Beschreibung die Grafik einer modizizierenden Kante

Passe die Farbe und Linienparameter der erzeugten Grafik gemäß den Werten der ursprünglichen Grafik an

Kantenakteur beim Bearbeiten bzw. Klassenakteur für mod. Kanten beim Erzeugen

Editionsakteur für modifizierende Kante im Zusammenspiel mit dem Benutzer

Bild–A29

Petrinetz
Erzeugen und Bearbeiten von modifzierenden Kanten – Überblick

Entwurf des Blockdiagrammeditors

Autor: Knöpfel | Datum: 9/95

Initialisierungsakteur des Editionsakteurs für die Bearbeitung modifizierender Kanten
(start–method des *bde–mod–arc–edit–agent)*

Vermerke Auftraggeber, Abschlußbotschaft, sowie die zu bearbeiteten Größen: Bogenhöhe u. –abstand und ggf. Start– und Endpunkt

Wechsel in den Dialogzustand "Bearbeitung modifizierende Kante" (tell *doc–editor* mid:push–editor–state bde–mod–arc–editor–class)

Anlegen und Öffnen einer Untergruppe anzulegen für die Hilfsgrafik (tell *plan–editor* mid:open–subedit)

es wurden zwei Endpunkte übergeben

Erzeuge aufgrund der geometrischen Beschreibung als Hilfsgrafik:
– ein Paar Bögen
– eine Mittellinie (zwischen diesen Bögen)

es wurden keine Endunkte übergeben

Erzeuge an der aktuellen Rasterposition eine Linie der Länge *Null*

sonst

Starte Größenbearbeitung der Linie gemäß vorgegebenem Richtungsparameter

Länge der Linie ungleich Null

sonst

Erzeuge als zusätzliche Hilfsgrafik zwei Bögen

Close–After–Creation–Flag gesetzt (Default)

Interaktives Bearbeiten der geometrischen Größen Start– und Endpunkt sowie Bogenabstand und –höhe mittels der Hilfsgrafik

entsprechend Benutzerwahl

Länge bearbeiten

Bogenabstand bearbeiten

Bogenhöhe bearbeiten

Bewegen der gesamte Grafik

Ermitteln der bearbeiteten geometrischen Größe

Wert unzulässig

sonst

Editionsschritt aufheben

Fehlermeldung ausgeben

neue Werte vermerken

ggf. Hilfsgrafik anpassen

sonst

Dialogzustand "Bearbeitung modifizierender Kante"

Benutzerwunsch nach Beenden der Bearbeitung

Abschlußakteur des Editions–akteurs für mod. Kanten *(close–method)*

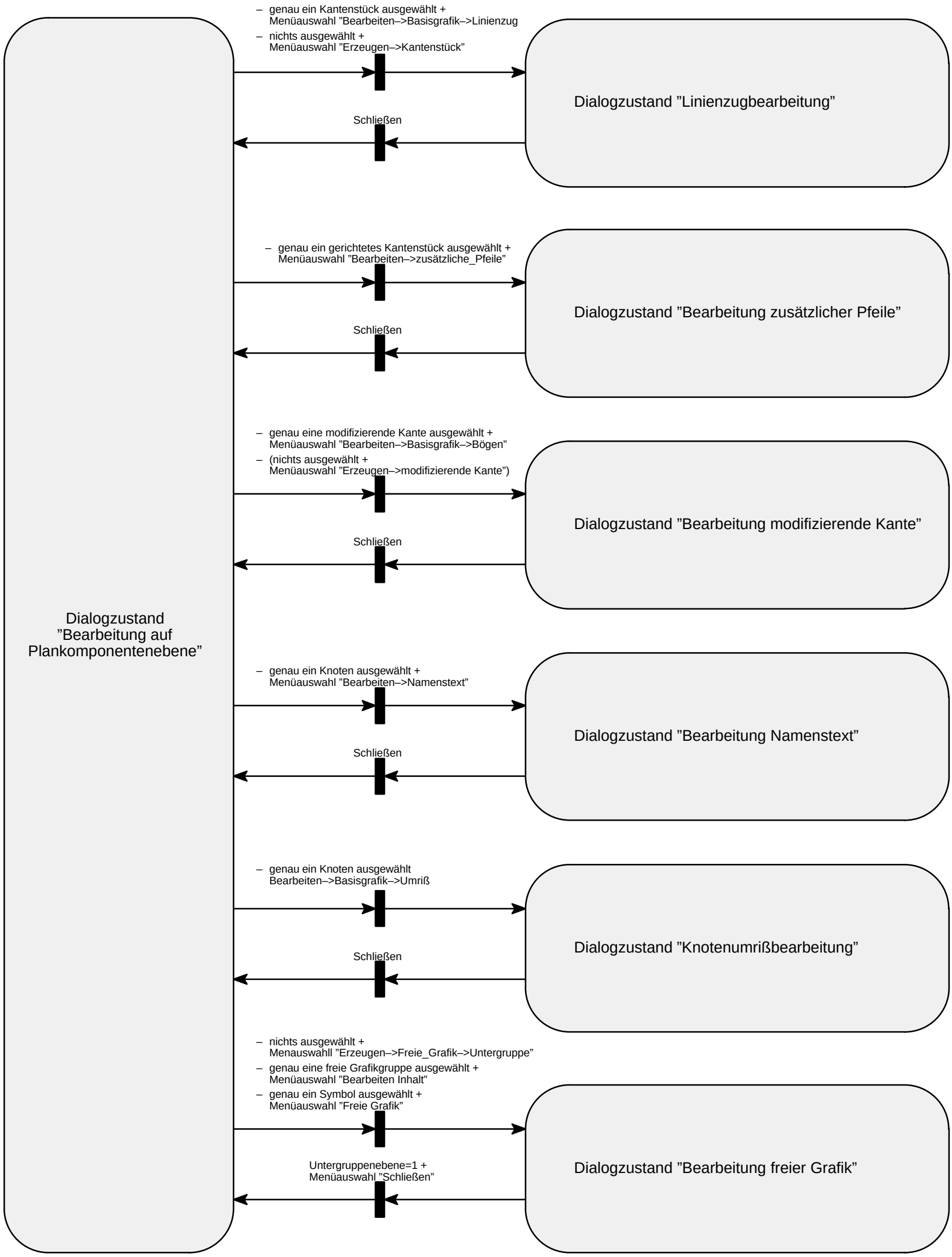
Ursprünglichen Dialogzustand wiederherstellen *(tell *doc–editor* mid:pop–editor–state*

Schließen und Löschen der Hilfsgrafikgruppe *(tell *plan–editor* mid:close–subedit)*

Sende Abschlußbotschaft inklusive Start– und Endpunkt, Bogenabstand, sowie Bogenhöhe an den Auftraggeber zurück

Bild–A30

Petrinetz: **Editionsakteur für die Bearbeitung modifizierender Kanten**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 9/95



Bild–A31

Dialoggraph **Dialogzustände des Blockdiagrammeditors Version 1.0**	
Entwurf des Blockdiagrammeditors	
Autor: Knöpfel	Datum: 11/95