LFCX:Diagrammtypen

1. UML für die Netzwerkerinnen [→ ZP:Sheet:2]

(I) UML . . .

- steht für *Unified Modeling Language* und "[. . .] ist eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Systeme" [→ Oestereich u. Scheithauer: UML-Kurzreferenz, 2014, S.5]
- ist keine Markup-Language (wie HTML, XML, LaTeX), sondern
- bietet eine Sammlung von Anleitungen und Symbolen und
- dient der Konstruktion von Diagrammen mit unterschiedlichem Zweck

(II) Systematik [→ ZP:Sheet:3]

Die Literatur zu UML klassifiziert Diagramme in zwei Gruppen, nämlich in die

- **Strukturdiagramme:** *(Structural Diagrams)*
  - **Klassendiagramm** *(Class Diagram)* := beschreibt Klassen von Objekten, ihre Eigenschaften, Operationen und Beziehungen untereinander (Welcher Typ subspezifiziert wen und vererbt seine Eigentschaften an ihn?)
  - **Objektdiagramm** *(Object Diagram)* := dokumentiert Beziehungen von Instanzen einer Klasse = ein zur Laufzeit vorhandenes Exemplar einer Klasse (Welches Objekt ist Komponente von welchem anderen?)
  - Komponentendiagramm
  - Kompositionsstrukturdiagramm
  - Verteilungsdiagramm
  - Paketdiagramm
  - Profildiagramm (= Stereotypen)
  - Anwendungsfalldiagramm *(Use Case Diagram)* (gelegentlich verwirrenderweise auch *Verhaltensdiagramm* genannt) := dokumentiert Systeme und Nutzer, zeigt Akteure, Anwendungsfälle und deren Beziehungen untereinander. (= Wer tut was für wen?)
- **Verhaltensdiagramme** *(Behavioral Diagrams)*
  - **Aktivitätsdiagramm** *(Activity Diagram}* := dokumentiert das Tun von beteiligten Komponenten, beschreibt einen Ablauf mittels Knoten und Kontrollflüssen (Wer tut was, wenn vorher wer was getan hat?)
  - Interaktionsdiagramme
    * Interaktionsübersicht
    * Kommunikationsdiagramm

    * **Sequenzdiagramm** *(Sequence Diagram)* := dokumentiert den Nachrichtfluss = zeigt eine Reihe von Nachrichten, die von den Beteiligten in einer bestimmten Reihenfolge ausgetauscht werden. (Wer sagt was zu wem wann?)
    * Zeitdiagramm
  - *Zustandsdiagramm (State Diagram)* := dokumentiert Zustände und Übergange von Systemkomponenten.
    * Protokollautomat

[vgl. → Oestereich u. Scheithauer: UML-Kurzreferenz, 2014, S. 10, 65, 91, 118, 141 et passim] /* Bibliographische Angaben im FI-Literaturverzeichnis */

Ironie:

- Das *(Use Case Diagram)* wird ja gelegentlich auch als *Verhaltensdiagramm* bezeichnet, weil es die verschiedenen Möglichkeiten eines Users mit einem System zu interagieren als Set von verschiedenen *Use Case Diagrammen* erfasst.
- Die *Verhaltensdiagramme* sind dagegen vom Typ 'Verhaltensdiagramm', weil sie den Ablauf eines Verhaltens in einem Diagramm dokumentieren.

[→ ZP:Sheet:4]

Betrachtet man nur die wichtigen Diagrammtypen (die fett markierten), wird das Gewirr durchsichtig:

- drei *Strukturdiagramme*, nämlich das *Klassendiagramm*, das *Objektdiagramm* und das *Usecase-Diagramm*
- zwei/drei *Behavioral Diagrams*, nämlich das *Aktivitätsdiagramm*, das *Sequenzdiagramm* und vielleicht noch das *Zustandsdiagramm*.

(III) Aktivitätsdiagramm

(A) Zulässige Symbole und Erläuterung s. **[→ ZP:Sheet:5]**

Ein Aktivitätsdiagramm

- beginnt immer mit **einem** Startknoten. Sind gleich von Anfang an mehrere Akteure aktiv, folgt dem Startknoten direkt das Forksymbol.
- kann mit **mehreren** Endknoten enden [vgl. → Oestereich u. Scheithauer: UML-Kurzreferenz, 2014, S.118]
- kann einen Akteur etwas tun lassen und trotzdem kein Resultat an andere weiterreichen lassen, z.B., weil er es ignorieren soll. Oder weil die Reaktion außerhalb des Prozesses bleibt.Das kann man sprachlich explizieren: `liest(A,x -> ignore(A,inhalt(X))` - wenn man es will.

(B) Beispiel s. **[→ ZP:Sheet:6]**

Kommentar:

- **Grundidee UML**: Viele verschiedene Diagramme beschreiben einen Gegenstand.
- **Praxis UML**: Die Symbolmenge wird um Zeichen aus anderen Diagrammtypen erweitert.
- **Empfehlunge**: Meiden Sie solche ‘Erweiterungen’. (Im Beispieldiagramm sind das etwa die Knoten `send signal` oder `trigger`)

(C) Rezept zum Erstellen eines Aktivitätsdiagramms

Üblicherweise erhalten Sie einen Text, der das zu dokumentierende Aktivitätsgeflecht beschreibt. Um das in und mit einem *Activity Diagram* zu dokumentieren, gehen Sie so vor:

1. Markieren / unterstreichen Sie alle Akteure im Text.
2. Markieren / unterstreichen Sie alle Handlungen im Text.
3. Ordnen Sie auf einem Papier jedem Akteur jede Handlung zu, die er ausführt.
4. Legen Sie für jeden Akteur im Diagramm eine ‘Lane’ an, sofern der Diagrammtyp das vorsieht.
5. Tragen Sie alle Handlungen mit entsprechendem Symbol als Aktivität bzw. Prozessschritt in das Diagramm ein. Beginnen die Beschreibung stets mit einem Verb!
6. Arrangieren Sie die Prozessschritte / Aktivitäten (und Zustände) entsprechend der zeitlichen und logischen Reihenfolge, die der Text vorgibt. Integriere ggfls. die Symbole (*JOINT, FORK*)
7. Verknüpfen Sie die Schritte / Aktionen (und ggfls. Zustände) mit Pfeilen.

Hinweise

1. Vermeiden Sie unbedingt passive Formulierungen und substantivierte Verben. Nutzen Sie stattdessen Aktivsätze. Die explizieren die Handelnde und ihre Tat.
2. Das Verfahren zum Erstellen eines Aktivitätsdiagramms gilt sinngemäß auch für Prozess- und Flussdiagramme. Manche von denen kennen das Konzept *Lane* nicht. Dann markieren Sie Zusammenhänge farblich.

Faustregel: **Pfeile immer zuletzt!**

(IV) Sequenzdiagramm

(A) Zulässige Symbole und Erläuterung s. **[→ ZP:Sheet:7]**

Ein Sequenzdiagramm

- hat für jeden Akteur eine Lebenslinie.

- repräsentiert die Reihenfolge der versendeten Nachrichten und Antworten samt Sendern und Empfängern.
- bietet trotz der intendierten linearen Folge die Möglichkeit, Bedingungen () und Schleifen () zu erfassen.

(B) Beispiel s. **[→ ZP:Sheet8]**

Hinweis

Am schwierigsten zu verstehende ist der Aktivitätsbalken:

- *Wikipedia* sagt, "die schmalen Rechtecke, die auf den Lebenslinien liegen, seien Aktivierungsbalken, die den Focus of Control anzeigen, also jenen Bereich, in dem ein Objekt über den Kontrollfluss verfügt, und aktiv an Interaktionen beteiligt ist". → https://de.wikipedia.org/wiki/Sequenzdiagramm
- *Lucidchart* meint, ein Aktivitätsbalken "[…] repräsentiere die Zeit, die ein Objekt zum Abschließen einer Aufgabe benötigt" und "je länger die Ausführung der Aufgabe dauere, desto länger (sei) der Aktivitätsbalken". → https://www.lucidchart.com/pages/de/uml-sequenzdiagramme
- Oestereich u. Scheithauer betonen, dass "beim Sequenzdiagramm […] der zeitliche Verlauf der Nachrichten im Vordergrund stehe" und dass Rollen (Akteure) nur ".[…] mit senkerechten Lebenslinie gezeigt (würden)". Würden Lebenslinie durch "nicht ausgefüllte (oder graue) Balken" überlagert, so "[…] symbolisieren diese den Steuerungsfokus". Der sei "**optional**" und gäbe bloß an, "[…] welche Rolle gerade die Programmkontrolle (besitze) […] [vgl. → Oestereich u. Scheithauer: UML-Kurzreferenz, 2014, S. 142f]:

> *"Die Länge des Steuerungsfocus [= Aktivitätsbalken, KR] hat keinerlei Aussagekraft, denn das Sequenzdiagramm zeigt nur die Reihenfolge der Ereignisse [= Kommunikationsschritte, KR], nicht die dazwischenliegen Zeit." [vgl. → Oestereich u. Scheithauer: UML-Kurzreferenz, 2014, S. 143]*

> *"Wenn während des Ausführungsfokus nichts passiert, sollte man der Übersichtlichkeit halber den Ausführungsfokus weglassen." [dies., ebda.]*

- Kecher, Hoffmann-Elbern u. Will interpretieren sprechen nicht vom *Aktivitätsbalken*, sondern vom "Ausführungsbalken": Für sie spezifiziert "die Lebenslinie" die "passive Lebenszeit (eines) Teilnehmers", der "Ausführungsbalken" dagegen seine "aktive Zeit". [vgl. → Kecher, Hoffmann-Elbern u. Will: UML 2.5, 2021, S. 353f]. Wir sehen, dass hinter dieser Deutung das Konzept eines Softwareprozesses steht, die in einem Rechner vom Betriebssystem 'nach vorne', will sagen in die Aktion geholt werden.
- Geht man in der Zeit zurück, sieht man, dass die Sequenzdiagramme weniger komplex waren, aber auch schon da auf den Programmierkontext auslegt waren. Der Zweck von Sequenzdiagrammen sei es, "(to) emphasize the time-based flow of events". Die "lifeline" wird auch hier

schon erwähnt, nicht aber der Aktivitätsbalken. Der wird nur - wie selbstverständlich - genutzt. [vgl. → Pilone: UML 2.0, 2006, S. 76ff]

Dieser inneren 'Widersprüche wegen' die *Faustregel*: **Aktivitätsbalken weglassen, Pfeile immer zuletzt!**

(C) Rezept zum Erstellen eines Sequenzdiagramms

Üblicherweise erhalten Sie einen Text, der das zu dokumentierenden Nachrichtengeflecht beschreibt. Um das in und mit einem *Sequence Diagram* zu dokumentieren, gehen Sie so vor:

- Markieren Sie alle Sender u. Empfänger im Text.
- Markiere alle Nachrichten im Text.
- Ordnen Sie jedem Sender alle Nachrichten zu, die er sendet, und vermerken Sie den Empfänger dazu.
- Legen für jeden Sender/Empfänger im Diagramm eine Lebenslinie an.
- Tragen Sie alle Nachrichten als Pfeil von Sender zu Empfänger ein. Markieren Sie den Typ durch den Linientyp.
- Arrangieren die Abfolge Nachrichten entsprechend der faktischen Reihenfolge top-down/left-right
- Markiere Sie die Nachrichten inhaltlich.
- Markiere ggf. durch Balken, ob ein Sender auf eine Antwort wartet (ohne die Zeit weiter anderwärtig zu nutzen [synchron]) bzw. ob er prozessual im Lead ist.

2. Sonstige Diagramme für Netzwerkerinnen

(V) Flussdiagramm

(A) Zulässige Symbole und Erläuterungen s. **[→ ZP:Sheet:9]**

Flussdiagramme (**Flow Charts**) sollten anfänglich (bedingte) Datenflüsse in einem Computerprogramm dokumentieren - zu einer Zeit, als

- als nebenläufige Arbeitsschritte in einem Programm eher 'Out-Of-Scope' waren
- als Entscheidungen JA/NEIN-Entscheidungen waren.

Deshalb fehlten in der *Sprache* 'Flow-Chart' ursprünglich:

- nebenläufig (parallele) Prozessschritte
- additive (AND) Nachfolgeschritte
- freie Auswahlschritte

Darum die später in die Definition eingefügten Symbole und Erläuterungen s. **[→ ZP:Sheet:10]**

(B) Beispiel s. **[→ ZP:Sheet11]**

(C) Rezept zum Erstellen eines Flussdiagramms

Üblicherweise erhalten Sie einen Text, der das zu dokumentierenden Nachrichtengeflecht beschreibt. Um das in und mit einem *Flussdiagramm* zu dokumentieren, gehen Sie so vor:

- Markieren Sie die Prozessschritte im Text
- Markieren Sie die Dateneingabeschritte im Text
- Labeln Sie die Prozess- und Dateneingabeschritte möglichst einfach
- Bringen Sie die Schritte optisch in die Reihenfolge, in der sie abgearbeitet werden
- Markieren Sie die Verzweigungen und Entscheidungen
- Verbinden Sie die Elemente in der intendierten Reihenfolge.
- Setzen Sie nötigenfalls Kommentare.

(VI) BPMN-Diagramm

Historischer Kontext:

- BPMN wurde seit 2001 v. Stephen A. White bei IBM entwickelt.
- Erstveröffentlichung 2004 durch *Business Process Management Inititative*
- Seit 2005 von der Object Management Group (OMG) weiterentwickelt. [OMG ist auch für UML Weiterentwicklung zuständig]
- Veröffentlichung BPMN-2.0 2011
- Seit 2013 ist BPMN-2.0.1 ISO-Standard
- (vgl. dazu Kersken: Daten- und Prozessanalyse für Fachinformatiker*innen, 2021, S. 429f)
- ISO/IEC 19510:2013: https://www.iso.org/standard/62652.html: *"ISO/IEC 19510:2013 is identical to OMG BPMN 2.0.1"*
- OMG BPMN Betreuung: https://www.omg.org/spec/BPMN/2.0.1/About-BPMN
- OMG BPMN Definition: Business Process Model and Notation (BPMN) Version 2.0.1 https://www.omg.org/spec/BPMN/2.0.1/PDF: (Bei Unklarheiten ist dieses Dokument mithin das entscheidende.)
  - *Basic BPMN Modeling Elements* = S.26 - S.30
  - *BPMN Extended Modeling Elements* = S.31 - S.39

(A) Zulässige Symbole und Erläuterungen s. **[→ ZP:Sheet:12]**

- Prozessschritten in der *Business Process Model and Notation*-Sprache stehen in der Wirklichkeit Aktionen (mit Zustandsänderungen) gegenüber.
- Als Schritte sind die Prozessschritte in erster Linie **Vorgänger** und **Nachfolger**
- Ein Schritt kann mehrere Nachfolger haben, aus denen bei einem Prozessdurchlauf **einer ausgewählt** wird.
  - Per 'Entscheidung' auszuwählende Prozessschritte werden durch ein OR-Symbol markiert, das in der Bedeutung von xOR genutzt wird.
- Gelegentlich will man auch es **nebenläufig auszuführende Prozessschritte** dokumentieren. Diese werden per definitionem 'gleichzeitig' ausgeführt.
  - Nebenläufig auszuführende Prozessschritte werden durch ein AND-Symbol markiert.

Randbemerkungen:

- *ZP:Sheet:12* listet
  1. die Symbole auf, die auch Sascha Kersken - Autor des Standardwerkes 'IT-Handbuch für Fachinformatikerinnen' - in seinem Buch zur Daten- und Prozessanalyse bespricht. (vgl. Kersken: Daten- und Prozessanalyse für Fachinformatiker*innen, 2021, S. 431f [Seitenangaben für OMG BPMN-Standard jeweils hinzugefügt]
  2. einige der Symbole auf, die der BPMN-Standard anbietet (Seitenangaben für OMG BPMN-Standard jeweils hinzugefügt)
- Verbindungspfeile führt Kersken nur implizit in und mit seinen Beispielen ein. Verschiedene Pfeiltypen verwendet er nicht (vgl. ders., a.a.O., S. 433 - 439)
- Tatsächlich bietet der OMG BPMN-Standard auch andere Pfeile an: zwei sind aufgelistet.
- 'Verzweigungen' (OR, XOR) und Parallelisierungen (AND) lässt Kersken mit dem entsprechenden Symbol beginnen. Deren Ende markiert er nicht. Will sagen: Anders als bei anderen Autoren gelegentlich zu sehen, nutzt der die Symbole **nicht** im Sinne einer Klammer.
- Der OMG BPMN-Standard bietet für den Fork – that "*refers to the dividing of a path into two or more parallel paths (also known as an AND-Split)*" – zwei Formen an, einen impliziten und einen expliziten mit AND-Gate [S.34]
- Der OMG BPMN-Standard bietet außerdem Symbole für die Zusammenführung: *Join* and *Merging* [S.35 und S. 36]
- Das Merging gibt es ebenfalls in zwei Formen, einer impliziten und einer expliziten. Bei der expliziten wird das allgemeine, unspezifizierte Gate-Symbol genutzt [S.36]

1. Die **Pflicht**, mit einem Gate geöffnete Strukturen an ihrem Ende mit demselben Gate-Symbol zu schließen, gibt die ISO-/OMG-Spezifikation nicht her, weder textuell noch durch die Beispiele.

2. Entsprechend heißt es in der jüngsten deutschsprachigen Einführung von 2025: *“Für die Zusammenführung ist in EPK ein Konnektor Pflicht, **in BPMN nicht** […]”.* [vgl. Freund u. Rücker: Praxishandbuch BPMN, 2025, S.101]
3. BPMN Merges etc. ermöglichen aber die Zusammenführung, wenn man deren Definition etwas weicher interpretiert.

(B) Beispiel s. **[→ ZP:Sheet13]**

(C) Rezept zum Erstellen eines BPMN-Diagramms:

[→ Rezept:Flussdiagramm]

(VII) EPK-Diagramm

Historischer Kontext:

- Initiale Idee zur Verwendung ereignisorientierte Prozessketten stammt aus 1992 [vgl. Keller, Nüttgens u. Scheer: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“, 1992]
- Eine formale Spezifikation dazu wurde von einem der Erstautoren 2003 ‘nachgereicht’ [vl. Nüttgens u. Rump: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK), in J. Desel und M. Weske [Hrsg.]: Promise, 2002, S.64 - 77]
- eine moderne Zusammenfassung der Nutzung stammt von kvp: → https://www.kvp.de/wp-content/uploads/2017/07/methodenblatt-epk-ereignisgesteuerte-prozesskette.pdf
- Eine scharfe Zusammenstellung der Nutzungsregeln liefert von Studydrive: https://www.studydrive.net/de/flashcards/die-10-wichtigsten-epk-regeln/18887
- Eine formale Standardisierung (wie bei BPMN als ISO-Norm) gibt es bei EPK - so weit ich erkennen kann - bisher nicht.
- Später sind Zusatzinformationen zur Verantwortung (ORG-Einheit), Informationsobjekten und Anwendungssystemen symbolisch erfasst worden. Damit entstanden die erweiterten Prozessketten, (eEPK bzw. eEPC) → https://de.wikipedia.org/wiki/Ereignisgesteuerte_Prozesskette

(A) Zulässige Symbole und Erläuterungen s. **[→ ZP:Sheet:14]**

Ereignisgesteuerte Prozessketten (EPK)

- sind als Modellierungssprache 1992 in Deutschland erfunden worden.
- haben darum international weniger Bedeutung.
- sind zentraler Bestandteil des ARIS-Konzeptes, einem “Ordnungsrahmen”, der “[…] von einer Aufteilung des Modells in Beschreibungssichten und -ebenen ausgeht”, um die “[…] Beschrei-

bung der einzelnen Elemente durch dafür speziell vorgesehene Methoden (zu) ermöglichen". → https://de.wikipedia.org/wiki/ARIS
- werden englisch mit dem Begriff *Event-driven-Process-Chains (EPC)* referenziert.

In der Konsequenz stellt `draw.io` von sich aus keinen Symbolpool dafür zur Verfügung. Auf Github gibt es aber eine freie Bibliothek in XML-Form samt Installationsanleitung:

> → https://github.com/AxelStieglbauer/drawio-lib-epc

Besondere Regeln zur Erstellung eines EPK-Diagramms:

- **Regel 1**: Jeder Aktion, Aufgabe, Prozesswegweiser geht ein Ereignis / Zustand voraus.
- **Regel 2**: Jeder Aktion, Aufgabe, Prozesswegweiser folgt mindestens ein Ereignis Zustand nach.
- **Regel 3**: Folgen Aktionen, etc. mehrere Zustände, werden sie per Konnektoren 'aufgespalten'.
- **Regel 4**: Ereignisse und Aktionen haben grundsätzlich nur einen Eingang und einen Ausgang (1 Pfeil rein und 1 Pfeil raus).

Besondere Symbole bei Erstellung eines EPK-Diagramms:

- **AND**-Konnektor: alle Nachfolger werden ausgeführt
- **OR**-Konnektor: 1-n aus n Nachfolgern werden ausgeführt
- **XOR**-Konnektor: 1 aus n Nachfolgern wird ausgeführt

(B) Beispiel s. **[→ ZP:Sheet15]**

(C) Rezept zum Erstellen eines BPMN-Diagramms:

[→ Rezept:Flussdiagramm]

Randbemerkungen:

Die **Pflicht**, mit einem Gate geöffnete Strukturen an ihrem Ende mit demselben Gate-Symbol zu schließen, ergibt sich spätestens aus der formalen Spezifikation und wird (partiell) in die Lehrtradition übernommen:

1. In der initialen Spezifikation von 1992 wird zumindest in der Beispielgrafik das 'schließende' Oder-Symbol eingeführt. [→ https://web.archive.org/web/20190109111105/http://www.uni-saarland.de/fileadmin/user_upload/Fachrichtungen/fr13_BWL/professuren/PDF/heft89.pdf, S. 13]
2. In der formalen Spezifikation von 2002 wird explizit festgelegt, "Funktionen (besäßen) genau eine eingehende und genau eine ausgehende Kontrollflusskante" und "Ereignisse hätten genau eine eingehende und/oder genau eine ausgehende Kontrollflusskante" [→ https://web.archive.org/web/20170121031104/https://www.wiso.uni-hamburg.de/fileadmin/wiso_fs_wi/EPK-Community/Promise2002_Nuettgens_Rump.pdf, S. 69]

> - Mithin müssen die durch Konnektoren aufgesplitteten Wege am Ende wieder zusammengeführt werden. Dies geschieht durch dasselbe Symbol, das den Split markiert.

3. Dem entspricht die Anweisung aus dem Studydrive-Dokument “Ereignisse und Funktionen (hätten) grundsätzlich nur einen Eingang und einen Ausgang (1 Pfeil rein und 1 Pfeil raus). [→ https://www.studydrive.net/de/flashcards/die-10-wichtigsten-epk-regeln/18887]
4. Die kvp als “Gesellschaft für Beratung und Weiterbildung” nimmt diesen Aspekt in ihrer Erklärung gerade NICHT explizit auf, obwohl die Diagramme das visuell nahezulegen scheinen: KVM geht es darum, “[…] dass mehrere Ereignisse einer Funktion bevorstehen bzw. folgen, oder, dass einem Ereignis mehrere Funktionen bevorstehen, von denen eins oder mehrere zum betrachteten Zeitpunkt ihre Gültigkeit haben” (können) → https://www.kvp.de/wp-content/uploads/2017/07/methodenblatt-epk-ereignisgesteuerte-prozesskette.pdf.

Randbemerkungen:

R1: zur Beendigung parallel ausgeführter Prozessschritte:

Sie finden gelegentlich die Aussage, bei einem BPMN- oder EPK-Diagramm würden, sofern diese den “[…] Sequenzfluss (aufteilen), […] am Ende der gleichzeitig ausgeführten Aufgaben ein zusammenführendes Gateway modelliert.” → https://www.cwa.de/bpmn-software-prozessmodellierung/

Zu Begründung heißt es, dieses Schließen …

- werde in alten Prüfungen auch so gemacht!
- “[…] kennzeichne, dass der Prozessablauf erst fortgesetzt wird, wenn alle davor parallel angestoßenen Aufgaben abgeschlossen wurden.”
- bewirke, dass die nächste Aktivität unabhängig von der gewählten Bedingung einheitlich fortgesetzt werde.
- verdeutliche das Verzweigungsende und mache den Prozess übersichtlicher.
- sei üblich, da dies den Prozess technisch und logisch korrekt mache, offen gebliebene Gateways könnten (von Tools) falsch verstanden werden.

Tatsächlich sind die Dinge schärfer geregelt:

1. In EPK geht jeder Aktion ein Zustand voraus und es folgt ihr ein anderer Zustand. Zustände und Aktionen haben je höchstens einen Eingang und/oder Ausgang. Mehrere Nachfolger und mehrere Vorgänger müssen mithin durch Konnektoren zusammengeführt werden. Die ‘Klammerungsregel’ ist mithin obligatorisach.
2. BPMN erlaubt die Verzweigung und Zusammenführung ohne Konnektoren. Die Klammerung ist bestenfalls optional.

R2: zur Öffnung parallel ausgeführter Prozessschritte:

1. EPK erlaubt den öffnenden Fork nur über einen Konnektor. Das ergibt sich aus der 1-In-1-Out-Regel.
2. BPMN erlaubt explizit den Fork (that "*refers to the dividing of a path into two or more parallel paths (also known as an AND-Split)*" OMG BPMN Standard S.34) ohne ein Gate.
3. Das Flussdiagramm lässt es grundsätzlich offen.

ABER:

- Ohne eine explizite Regelung (wie im OMG BPMN Standard, dass ein nicht symbolisch markierter Fork als parallele Ausführung = AND zu lesen ist) bleibt eine Verbindung von einer Vorgängeraktivität zu mehreren Nachfolgeraktivitäten ambigue: Soll das als *AND*, *OR*, oder *xOR* gelesen werden.
- Deshalb ist es grundsätzlich gut (wenn auch formal nicht zwingend), den Übergang mit einem Symbol (Gate) entsprechend auszuweisen.

ALSO **FAUSTREGEL**: *Öffnende Verzweigungen immer mit einem Symbole explizieren*!!

R3: Probleme der Reihenfolgen:

- Ein öffnendes AND-Gate meint implizit immer, dass die folgenden Aktionen parallel ausgefüllt werden.
  - ein öffnendes AND-Gate für dieselbe Person (Akteurin) ist empirisch inadäquat, weil dieselbe Akteurin nicht zeitgleich 2 Dinge tun kann.
  - ein öffnendes AND-Gate für dieselbe Abteilung (Akteurin) ist modellierungstechnisch unzureichend, weil es verschleiert, welcher Teil der Abteilung was in welcher Reihenfolge tut.
- Ein öffnendes explizites oder implizites OR-Gate als die Option zu verstehen, dass die Akteurin sich die Reihenfolge aussuchen kann, würde formal die Grafenforderung verletzen, weil bei einem Prozessablauf damit 'aus der Reihe' zu einem Knoten gesprungen werden müsste.

So ergibt sich folgende Zeichensynopse **[→ ZP:Sheet16]**