



# Workflow Mining

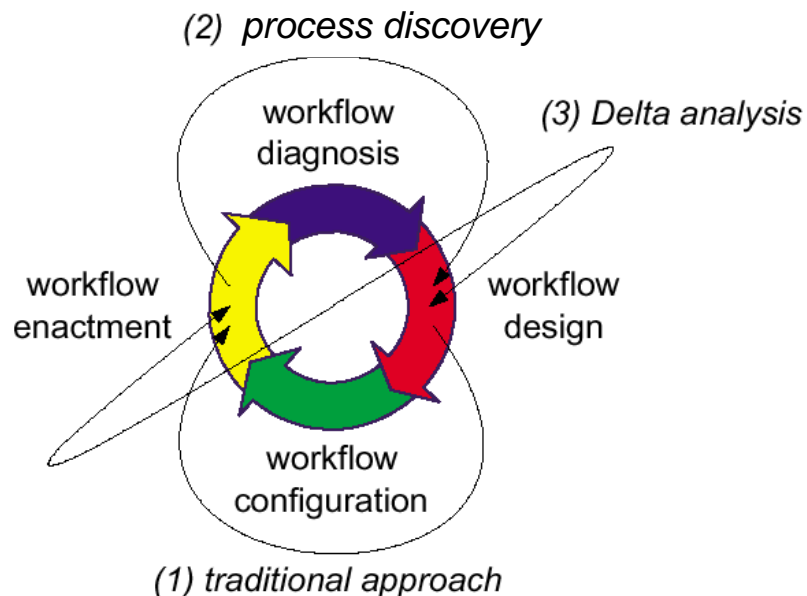
**Vorlesung:  
Workflow-Management-Systeme**

**Frank Eichinger**



# Workflow Mining

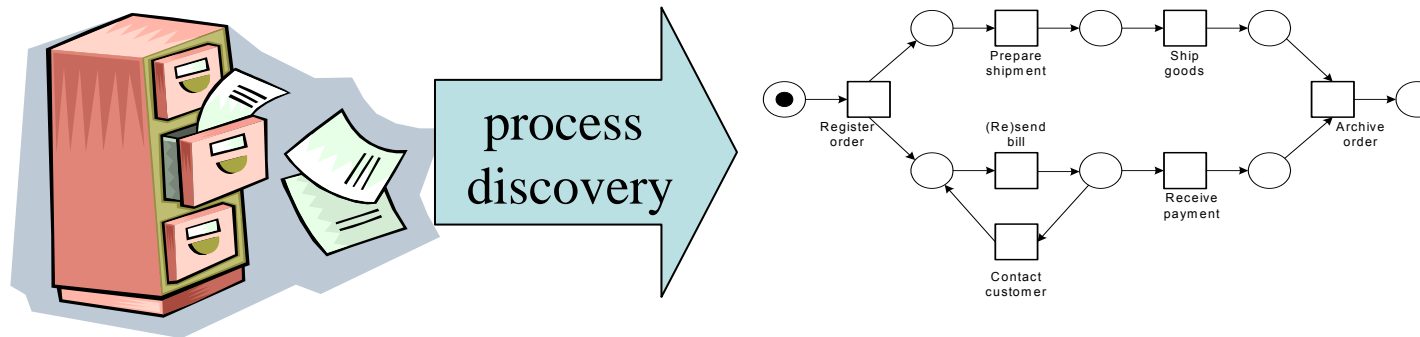
- Analyse-Techniken für bestehende Systeme.
- Ziel: Verbesserungs-Potentiale aufdecken.
- Kann als Input für BPR (Business Process Reengineering) und CPI (Continuous Process Improvement) genutzt werden.



- Verschiedene Aspekte:
  - Process Discovery (2) – Wie sieht der Prozess (wirklich) aus?
  - Delta Analysis (3) – Passiert das, was spezifiziert wurde?
  - Performance Analysis – Wie können wir den Workflow verbessern?

# Process Discovery

- Idee: Umdrehen des traditionellen Ansatzes

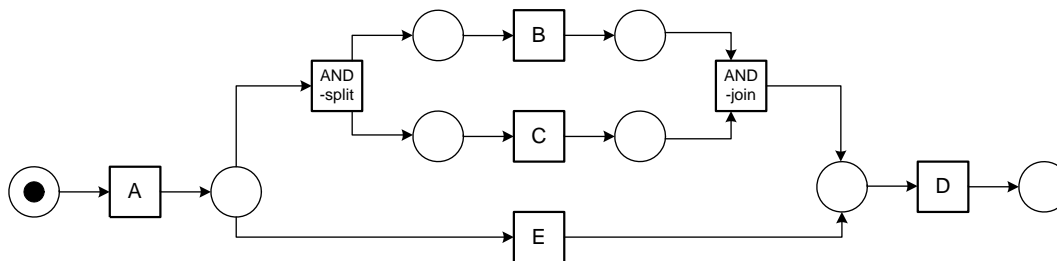


- Ausgangspunkt: Logfile
  - Minimal: Fall-Nummer, Task, zeitliche Ordnung
  - Optional: Genaue Zeit, User, assoziierte Daten etc.
  - „Rauschfrei“
  - „Vollständig“
- Ergebnis: Workflow-Schema als Petri-Netz



## Beispiel-Logfile

- Das Beispiel enthält die folgenden Sequenzen:
  - ABCD (case 1, case 3)
  - ACBD (case 2, case 4)
  - AED (case 5)
- Vereinfachte Darstellung:
  - {ABCD, ACBD, AED}
- Vorab: das Petri-Netz dazu:



```
case 1, task A
case 2, task A
case 3, task A
case 3, task B
case 1, task B
case 1, task C
case 2, task C
case 4, task A
case 2, task B
case 2, task D
case 5, task A
case 4, task C
case 1, task D
case 3, task C
case 3, task D
case 4, task B
case 5, task E
case 5, task D
case 4, task D
```



## Ordnungs-Relationen

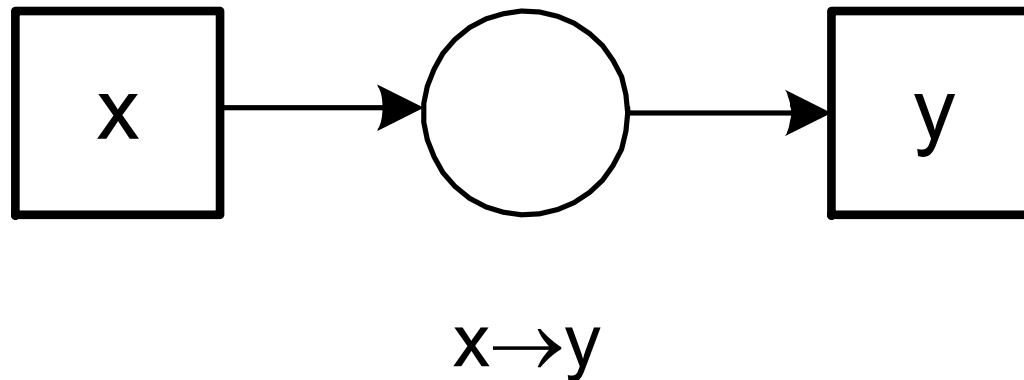
- Direkte Nachfolge:  $x > y$ 
  - Gdw.  $y$  folgt direkt auf  $x$
- Kausalität:  $x \rightarrow y$ 
  - Gdw.  $x > y$  und nicht  $y > x$
- Parallelität:  $x || y$ 
  - Gdw.  $x > y$  und  $y > x$
- Unabhängigkeit:  $x \# y$ 
  - Gdw. nicht  $x > y$  und nicht  $y > x$  und  $x \neq y$

- Im Beispiel  $\{ABCD, ACBD, AED\}$ :
- $A > B, B > C, C > D, A > C, C > B, B > D, A > E, E > D$
- $A \rightarrow B, A \rightarrow C, A \rightarrow E, B \rightarrow D, C \rightarrow D, E \rightarrow D$
- $B || C, C || B$
- $A \# D, B \# E, C \# E$



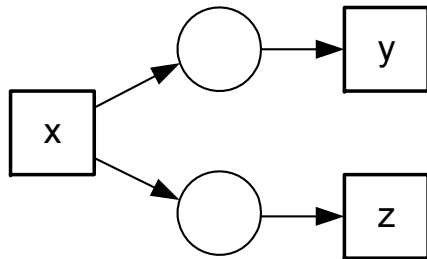
## Idee: Erzeugung eines Petri-Netzes (1)

- Wie kann aus den Relationen ein Workflow-Schema abgeleitet werden?
  - Einfachster Fall: Kausalität

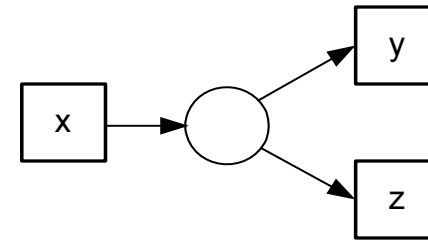




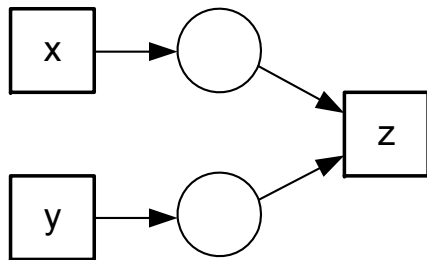
## Idee: Erzeugung eines Petri-Netzes (2)



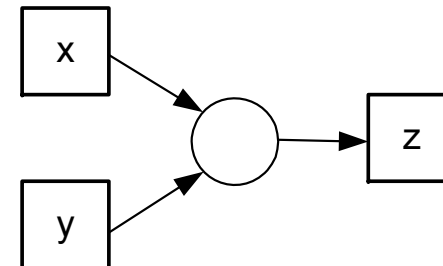
$x \rightarrow y, x \rightarrow z, y \parallel z$



$x \rightarrow y, x \rightarrow z, y \# z$



$x \rightarrow z, y \rightarrow z, x \parallel y$



$x \rightarrow z, y \rightarrow z, x \# y$



## Formalisierung: Der $\alpha$ -Algorithmus

Gegeben: Workflow-Log  $W$

Gesucht: Workflow-Schema als Petri-Netz

$$\alpha(W) = (\text{Stellen } P_W, \text{ Transitionen } T_W, \text{ Verbindungen } F_W)$$

1.  $T_W = \{t \in T \mid \exists \sigma \in W t \in \sigma\}$ , Transitionen
2.  $T_I = \{t \in T \mid \exists \sigma \in W t = \text{first}(\sigma)\}$ , Start-Transition
3.  $T_O = \{t \in T \mid \exists \sigma \in W t = \text{last}(\sigma)\}$ , End-Transition
4.  $X_W = \{(A, B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall a \in A \forall b \in B a \rightarrow_W b \wedge \forall a_1, a_2 \in A a_1 \#_W a_2 \wedge \forall b_1, b_2 \in B b_1 \#_W b_2\}$ , „Kausalitätsrelationen“
5.  $Y_W = \{(A, B) \in X_W \mid \forall (A', B') \in X_W A \subseteq A' \wedge B \subseteq B' \implies (A, B) = (A', B')\}$ , „Maximale Kausalitätsrelationen“
6.  $P_W = \{p_{(A,B)} \mid (A, B) \in Y_W\} \cup \{i_W, o_W\}$ , Stellen
7.  $F_W = \{(a, p_{(A,B)}) \mid (A, B) \in Y_W \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_W \wedge b \in B\} \cup \{(i_W, t) \mid t \in T_I\} \cup \{(t, o_W) \mid t \in T_O\}$ , and Verbindungen
8.  $\alpha(W) = (P_W, T_W, F_W)$ . Petri-Netz



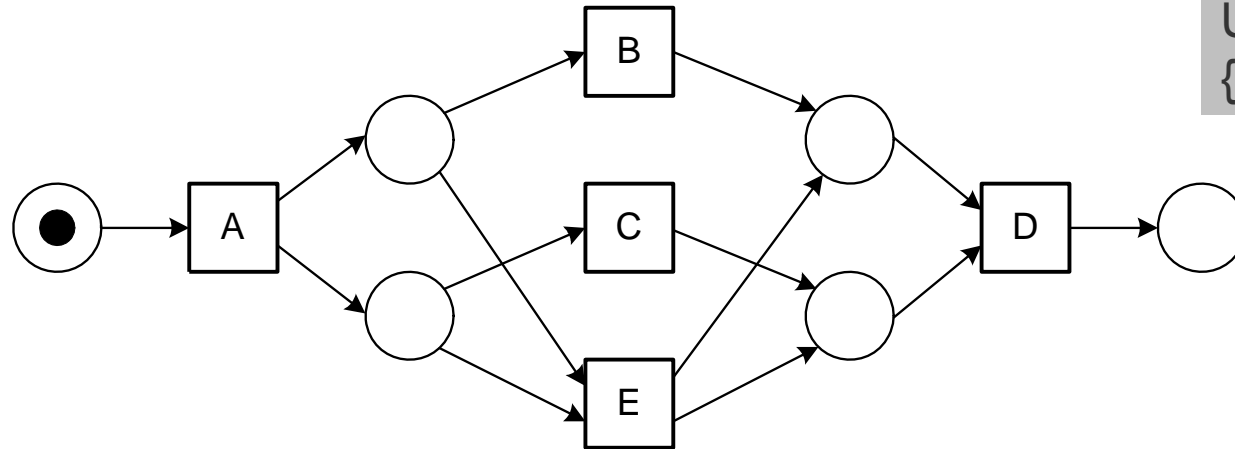
## $\alpha$ -Algorithmus – Beispiel

1.  $T_w = \{A, B, C, D, E\}$
2.  $T_I = \{A\}$
3.  $T_O = \{D\}$ ,
4.  $X_w = \{(\{A\}, \{B\}), (\{A\}, \{C\}), (\{A\}, \{E\}), (\{B\}, \{D\}), (\{C\}, \{D\}), (\{E\}, \{D\}), (\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\})\}$
5.  $Y_w = \{(\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\})\}$
6.  $P_w = \{i_w, o_w, p(\{A\}, \{B, E\}), p\{A\}, \{C, E\}), p(\{B, E\}, \{D\}), p(\{C, E\}, \{D\})\}$ ,
7.  $F_w = \{(i_w, A), (A, p(\{A\}, \{B, E\})), (p(\{A\}, \{B, E\}), B), \dots, (D, o_w)\}$
8.  $\alpha(W) = (P_w, T_w, F_w)$

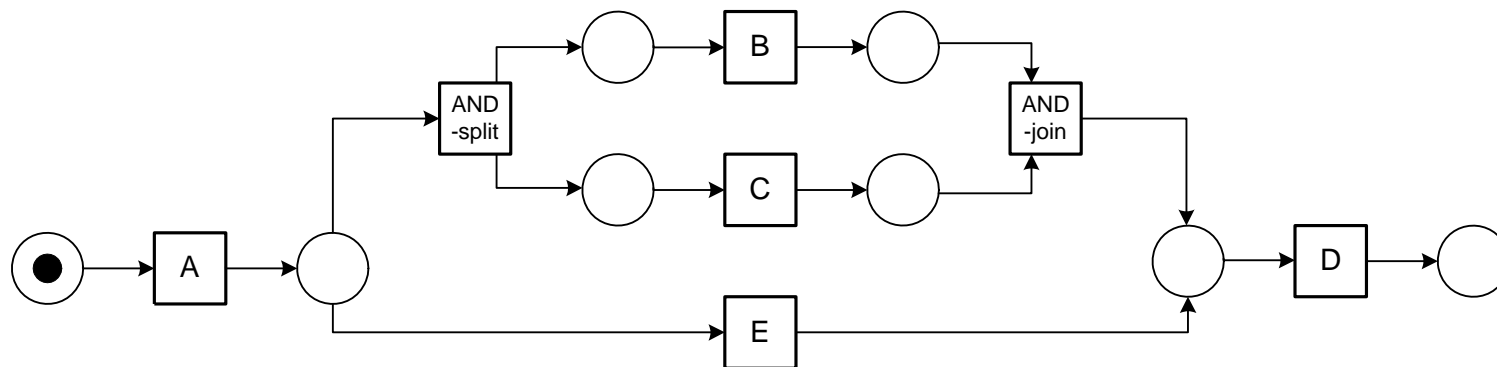
Ursprungs-Log:  
{ABCD, ACBD, AED}



# $\alpha$ -Algorithmus – Beispiel-Netz



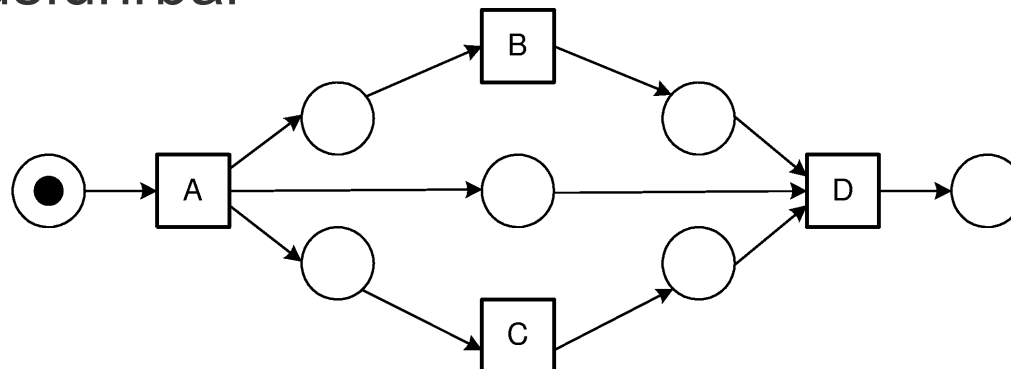
Ursprungs-Log:  
{ABCD, ACBD, AED}





## $\alpha$ -Algorithmus – Unzulänglichkeiten (1)

- Es können nur „beobachtbare“ Netzwerke erzeugt werden.
  - Splits und Joins nicht immer direkt beobachtbar.
  - Beobachtetes Verhalten wird aber korrekt wiedergegeben.
- Anderes Problem: ‚E‘ nicht sichtbar
  - {AD} nicht ausführbar



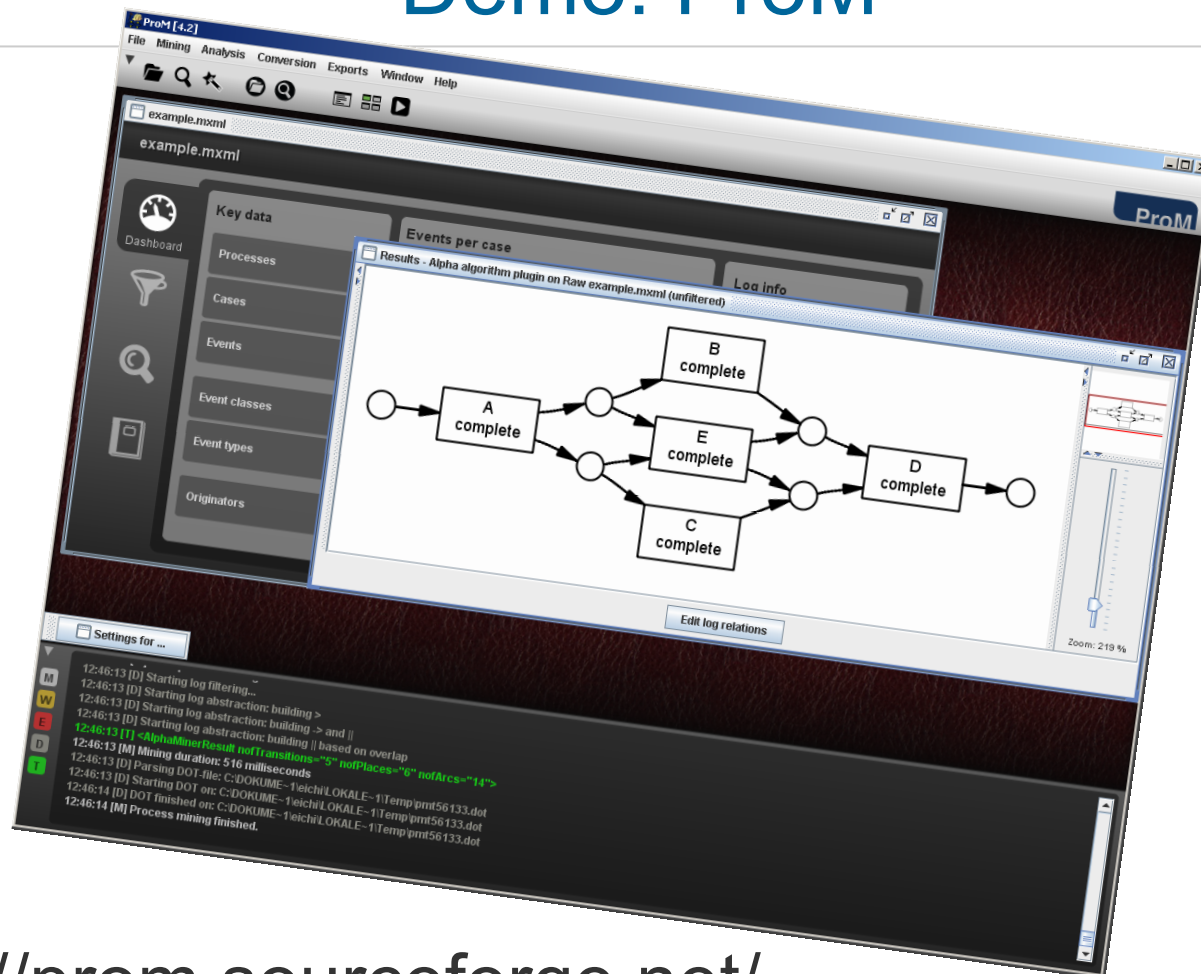


## $\alpha$ -Algorithmus – Unzulänglichkeiten (2)

- Mehrfach auftretende Aktivitäten
  - Schwierig, der  $\alpha$ -Algorithmus konstruiert nur eine Transition pro Aktivität.
- Iterationen
  - Führen zu mehrfach auftretenden Aktivitäten.
  - Müssen ggf. extra erkannt werden.
    - Mitunter sehr schwierig!
- Verrauschte Daten
  - Lösung z.B. durch Betrachtung der Häufigkeit.
- ...



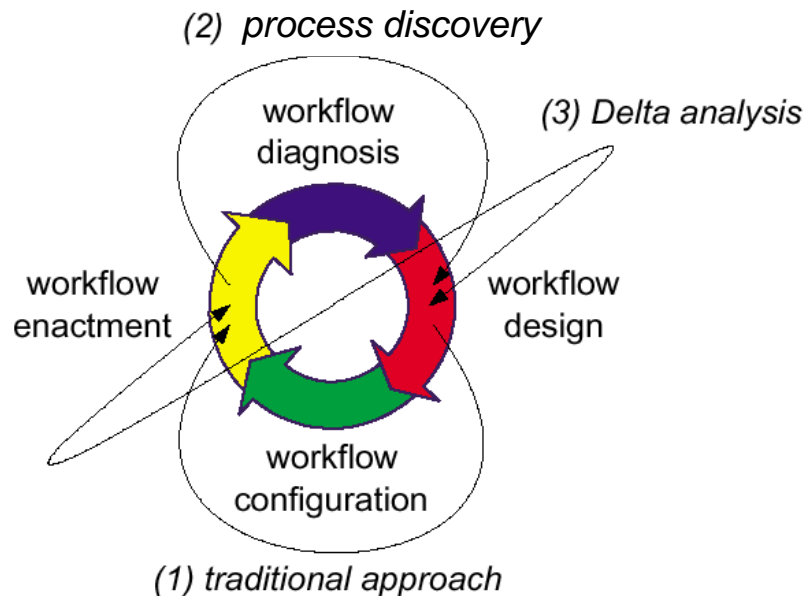
## Demo: ProM



<http://prom.sourceforge.net/>



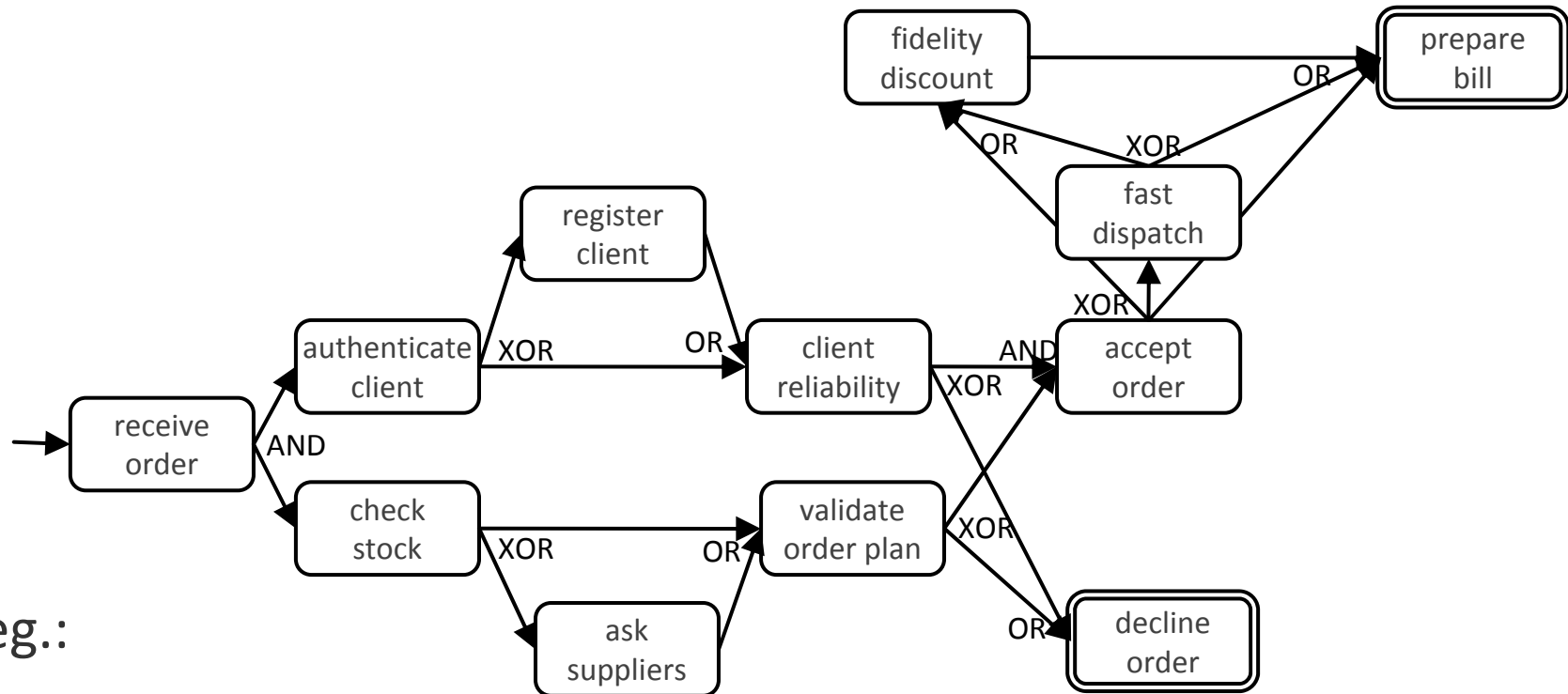
# Workflow Mining



- Verschiedene Aspekte:
  - Process Discovery (2) – Wie sieht der Prozess (wirklich) aus?
  - Delta Analysis (3) – Passiert das, was spezifiziert wurde?
  - Performance Analysis – Wie können wir den Workflow verbessern?
- Nächster Schritt („workflow diagnosis“):
  - Welche Pfade sind häufig?
  - Welche Instanzen werden wahrscheinlich abgebrochen?



## Workflow-Diagnose mit Graph Mining (1)

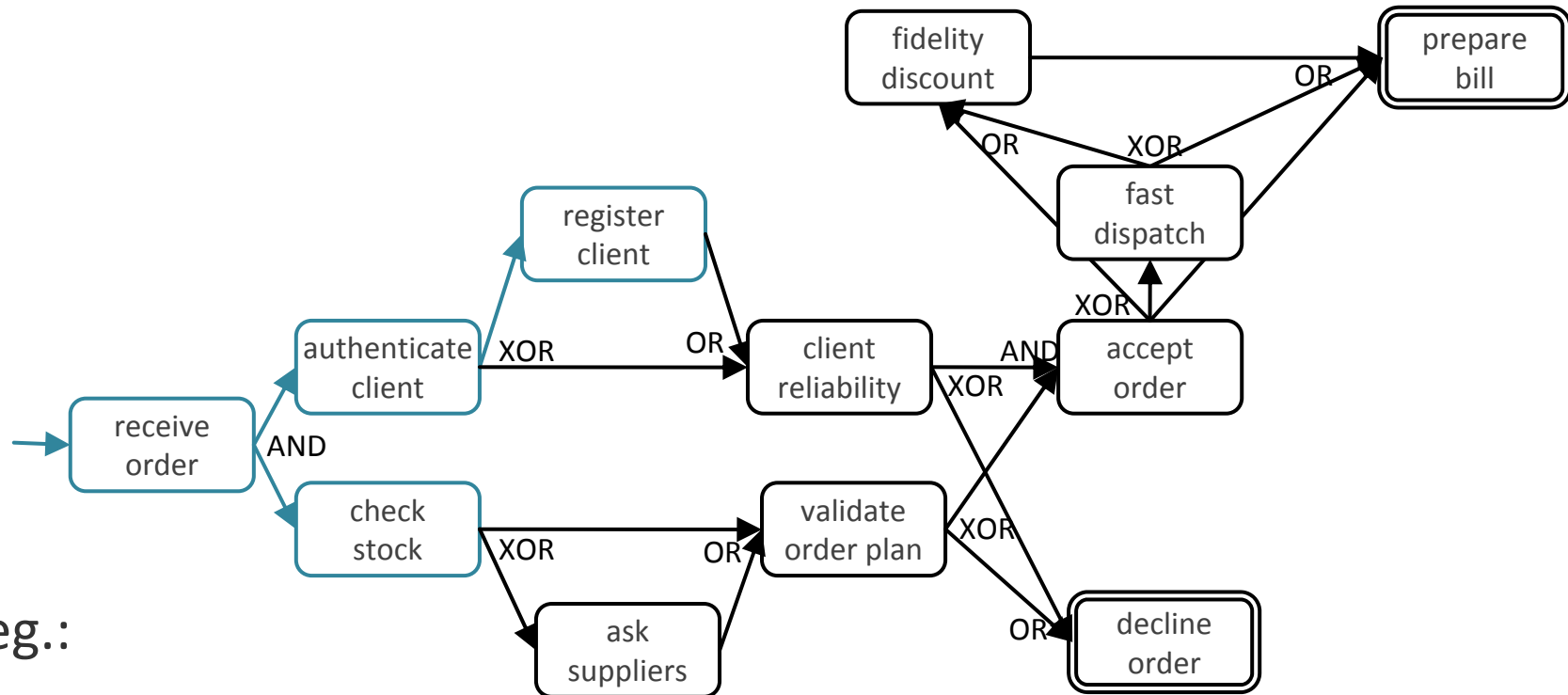


Geg.:

Workflow-Schema



## Workflow-Diagnose mit Graph Mining (2)



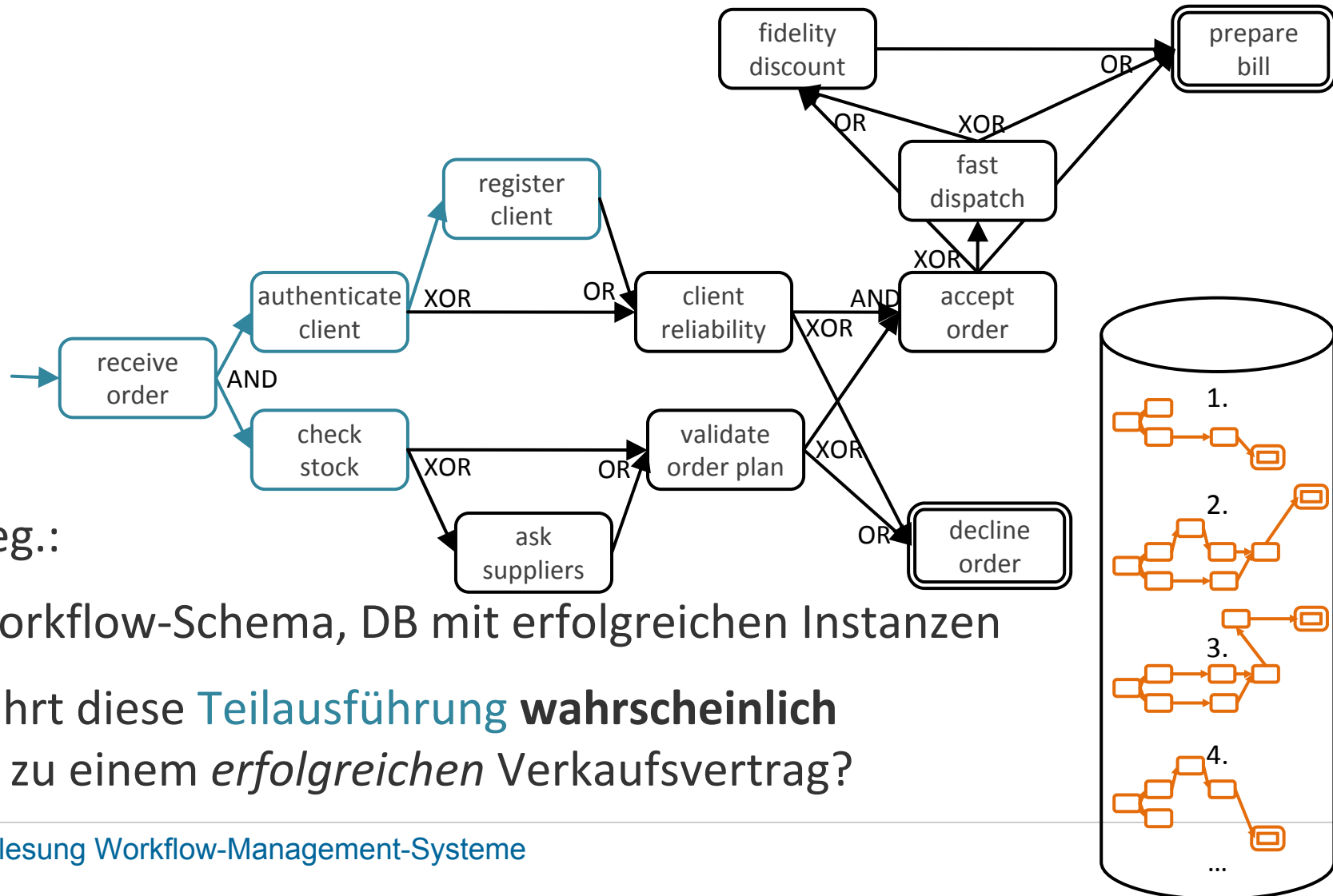
Geg.:

Workflow-Schema

Führt diese **Teilausführung** möglicherweise  
zu einem *erfolgreichen* Verkaufsvertrag?



## Workflow-Diagnose mit Graph Mining (3)



Geg.:

Workflow-Schema, DB mit erfolgreichen Instanzen

Führt diese **Teilausführung** **wahrscheinlich**  
zu einem *erfolgreichen* Verkaufsvertrag?



## Workflow-Diagnose mit Graph Mining (4)

---

- Weitere interessante Fragestellungen
  - Fehlerfreie Workflow-Ausführung?
  - Workflow-Ausführung mit geringem oder normalem Ressourcenverbrauch?
  - Identifikation kritischer Aktivitäten (besonders hoher Ressourcenverbrauch).
  - Wie sehen häufige/typische Ausführungen aus?
- Verwertung der Information
  - Zur Laufzeit: Scheduling
  - Information für Management/Optimierung (BPR, CPI)
- Ansatz: Ermitteln von *frequent patterns* bei gegebenem Workflow-Schema und Prozess-Logs.



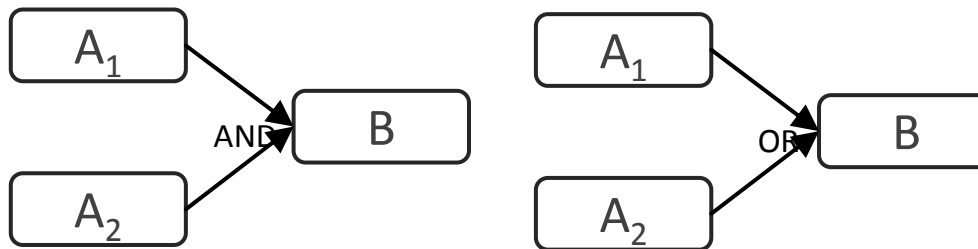
## Graph Mining

- Finden von *häufigen Subgraphen* in einer Menge von Graphen (Graph-Datenbank).
  - Häufig: Ein Graph  $G$  ist Subgraph von mindestens *minSup* Graphen der Graph-Datenbank.
- Typische Anwendungen
  - Chemie und Pharmaforschung (Moleküle)
  - Softwaretechnik (Call-Graphen)
  - Soziale Netzwerke etc.
- Auch Workflow-Schemata und -Instanzen sind Graphen!

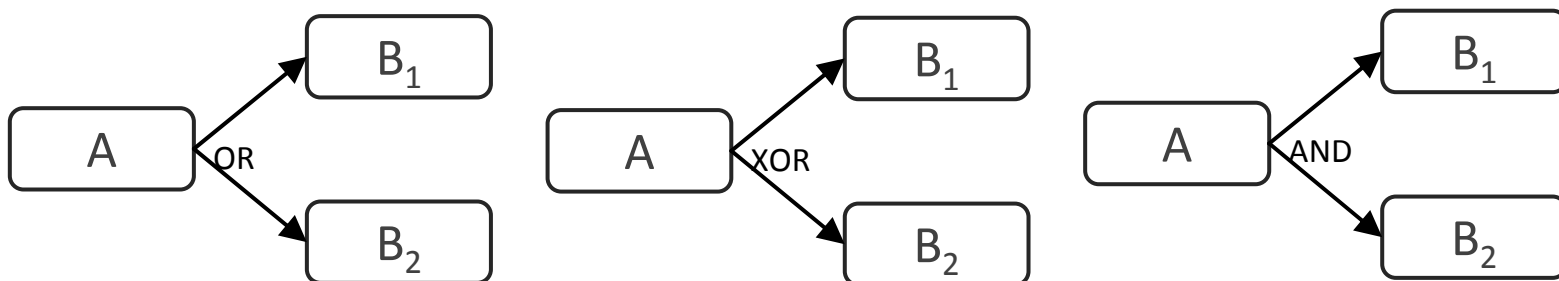


## Einschränkungen bei Workflow-Graphen

- AND-Join / OR-Join



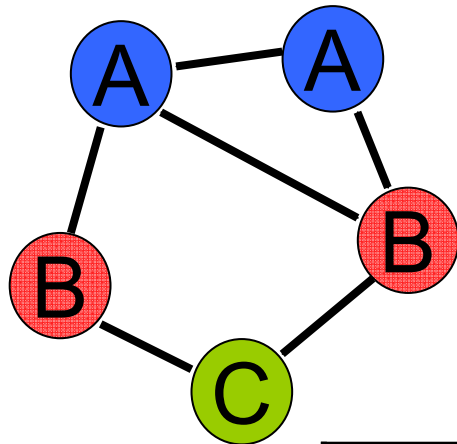
- Full Fork (OR) / Exclusive Fork (XOR) / Deterministic Fork (AND)



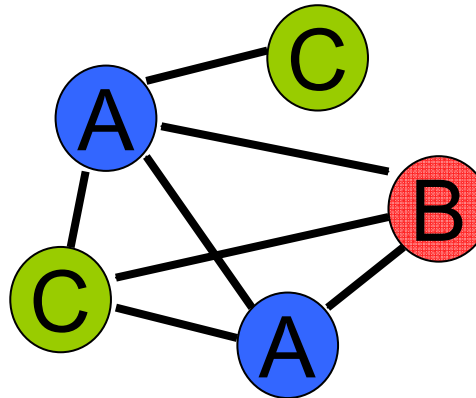


## Graph Mining - Beispiel

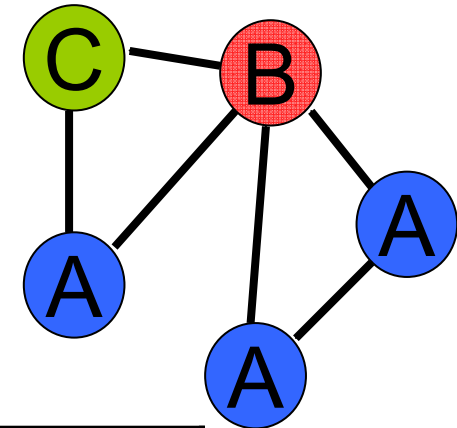
(1)



(2)



(3)



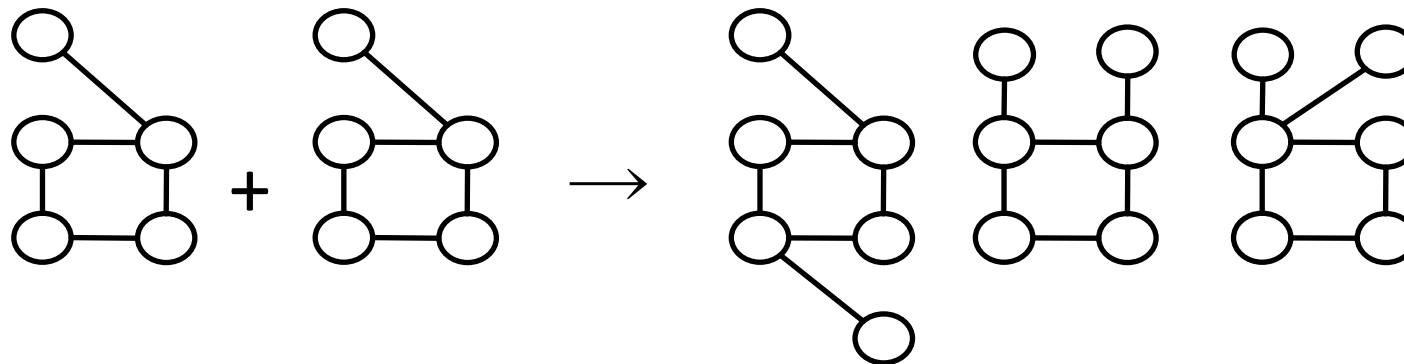
Support	1	2	3
Teilgraph			

Apriori-Prinzip: G häufig → Alle Subgraphen häufig!



# Apriori Graph Mining Algorithmus

- Iteratives Vorgehen (Breitensuche):
  1. Suche alle häufigen Kanten in der Graph-Datenbank ( $n=1$ )
  2. Generieren von Kandidaten der Größe  $n+1$  aus häufigen Subgraphen der Größe  $n$
  3. Überprüfen der Häufigkeit der Kandidaten in der Graph-Datenbank
  4. Wiederhole 2+3 solange häufige Graphen gefunden werden
- Kandidatengenerierung durch Join von  $n-1$ -Kernen:





## Nachteile von Apriori-basiertem Graph Mining

- Aufwändige Kandidatengenerierung
  - Jeder Graph der Größe  $n$  wird mit jedem verglichen.
  - Finden gemeinsamer Kerne beinhaltet Subgraphisomorphie.
  - Mergen der Kerne beinhaltet Suche nach Automorphismen.
- Aufwändige Kandidatenüberprüfung
  - Es entstehen im allgemeinen sehr viele Kandidaten.
  - Jede Überprüfung beinhaltet Subgraphisomorphie.
- Subgraphisomorphie ist NP-vollständig.
- **Kein Skalieren für große Workflow-Netze.**
  - Es existieren andere Graph Mining Algorithmen, die besser skalieren (PatternGrowth-Ansatz). Nicht Gegenstand heute.



## Algorithmus *w-find*

- Graph Mining-Algorithmus *w-find* nach GRECO
  - Finden von *frequent patterns* bzw. häufigen Teilprozessen
  - Apriori-ähnlich
  - Effizienzsteigerung durch Nutzung der Workflow-spezifischen Einschränkungen
    - Nicht sämtliche häufige Aktivitäten Grundlage der Kandidatengenerierung, sondern häufige *weak patterns*, welche die Regeln des Workflow-Schemas erfüllen.

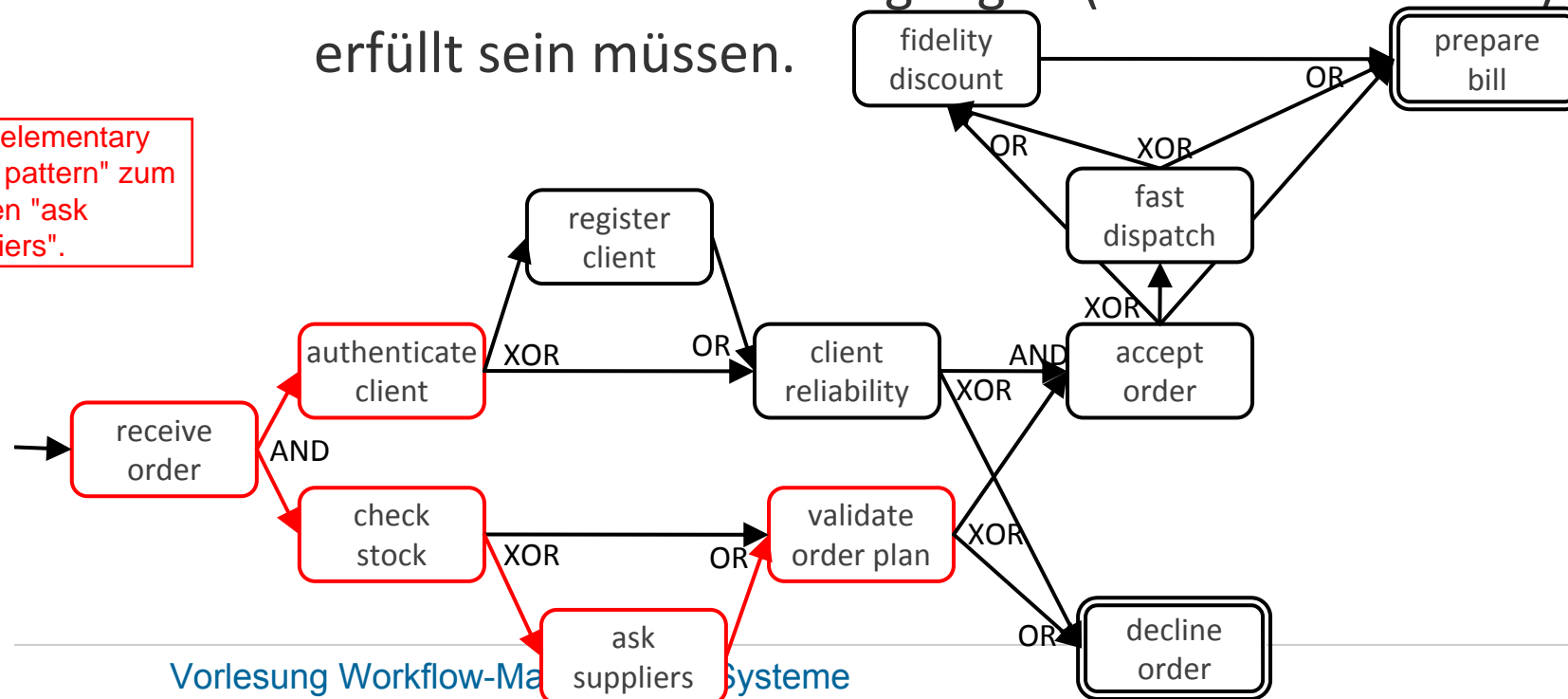


## Funktionsweise *w-find*

### 1. Finden von *elementary weak patterns*:

Ausgehend von einzelnen Aktivitäten Erweiterung auf ein zusammenhängendes *weak pattern*, in dem AND-Joins und AND-Verzweigungen (deterministic fork) erfüllt sein müssen.

Rot: "elementary weak pattern" zum Knoten "ask suppliers".

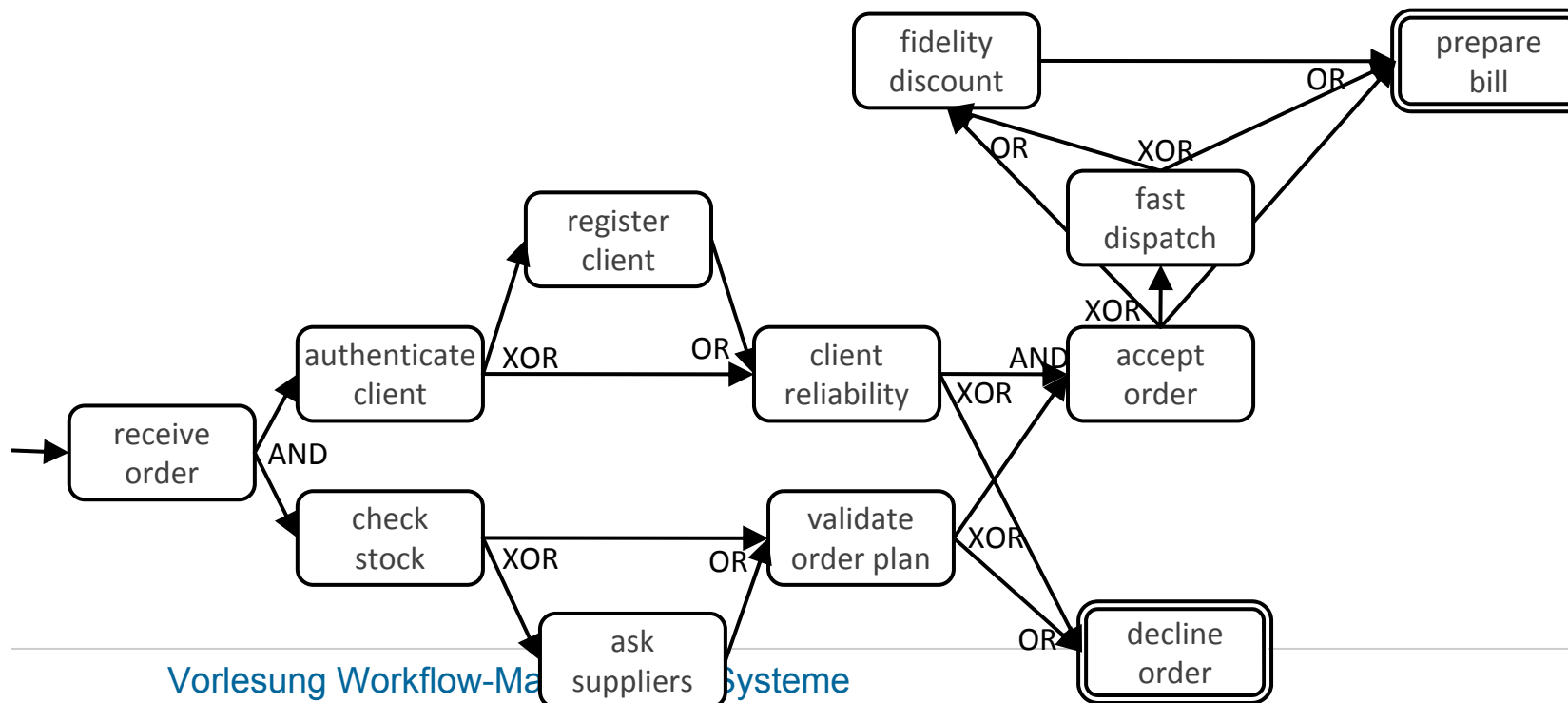




## Funktionsweise *w-find*

### 2. Identifikation der *frequent weak patterns*:

Berücksichtige nur weak patterns mit einer bestimmten Häufigkeit (*minSup*).

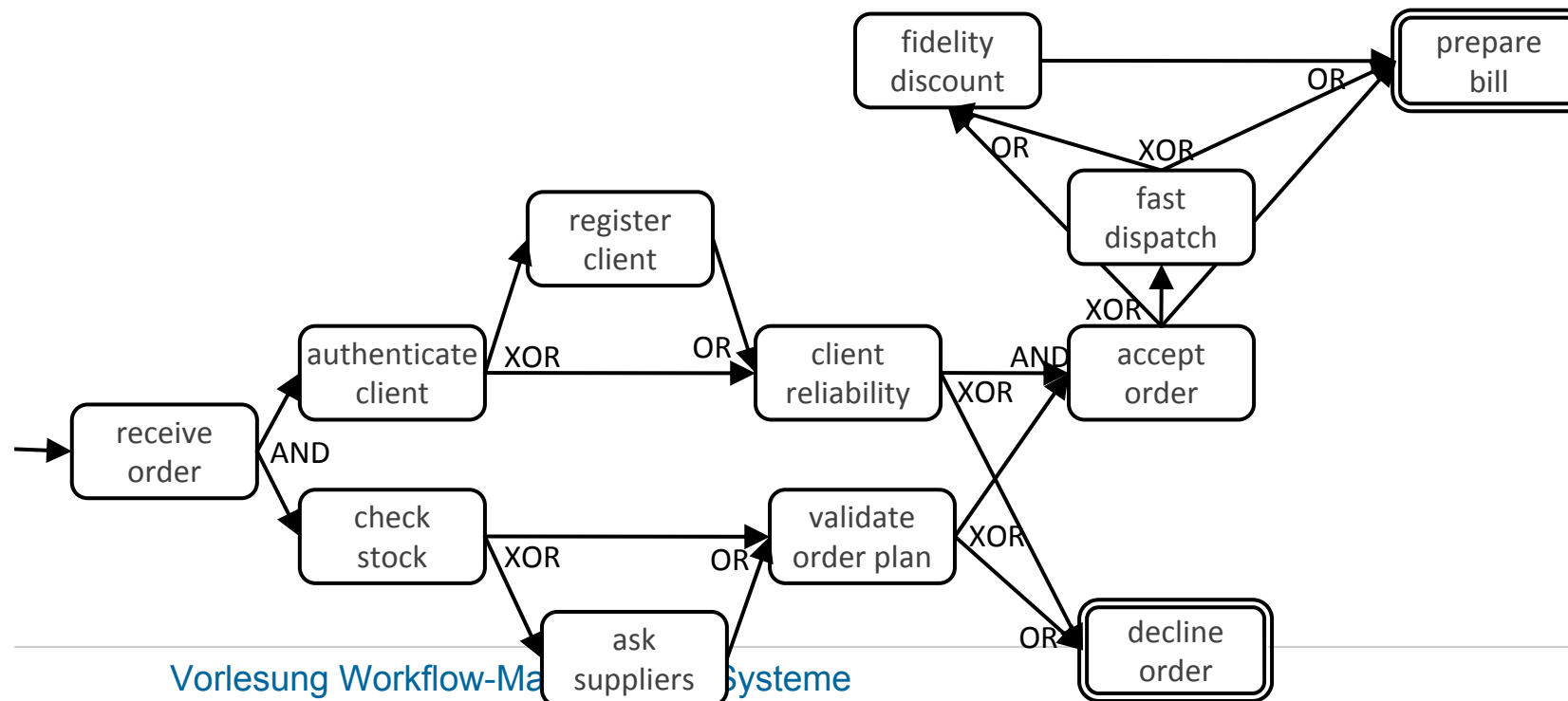




## Funktionsweise *w-find*

### 3. Erweitern der Menge der *frequent patterns* durch

- Hinzufügen von Kanten aus dem Workflow-Schema.
- Verschmelzung oder Verbindung zweier weak patterns.





## Quellen

- **W.M.P. van der Aalst and A.J.M.M. Weijters: Process Mining; M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede (Eds.), Process-Aware Information Systems: Bridging People and Software through Process Technology, pages 235-255. Wiley & Sons, 2005; <http://tabu.tm.tue.nl/wiki/publications/aalst2005e>**
- *A.K.A. de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters: Workflow Mining: Current Status and Future Directions; R. Meersman et al. (Eds.): CoopIS/DOA/ODBASE 2003, LNCS 2888, pages 389-406, 2003; <http://tabu.tm.tue.nl/wiki/publications/medeiros2003>*
- *W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster: Workflow Mining: Discovering Process Models from Event Logs; QUT Technical report, FIT-TR-2003-03, Queensland University of Technology, Brisbane, 2003; <http://tabu.tm.tue.nl/wiki/publications/aalst2003>*
- *Wil van der Aalst: Process Mining: Discovering processes from event logs; Talk at the CTIT BPM day, University of Twente, April 29th, 2005*
- *Alexander Fritz: Graph Mining für Workflows; Seminar Workflowmodellierung, Workflow-Mining und Identitätsmanagement, Institut für Programmstrukturen und Datenorganisation (IPD), Universität Karlsruhe (TH), Februar 2008; <http://dbis.ipd.uka.de/956.php>*
- *Greco, G.; Guzzo, A.; Manco, G.; Pontieri, L. & Saccà, D.: Mining Constrained Graphs: The Case of Workflow Systems; Proceedings of the Workshop on Inductive Databases and Constraint Based Mining, 2004, 155-171*