Kapitel 6

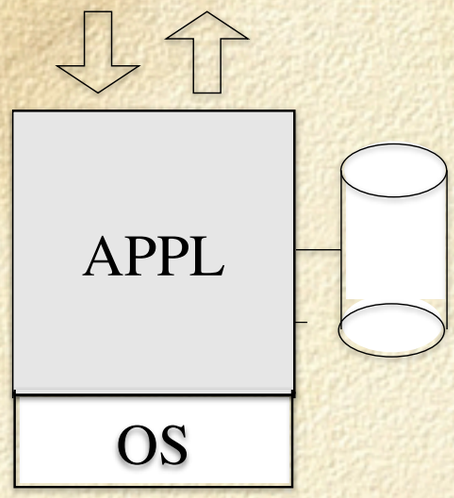
Konsistenz

6.1 Ablaufkonsistenz: Workflow

6.2 Datenkonsistenz: Serialisierbarkeit

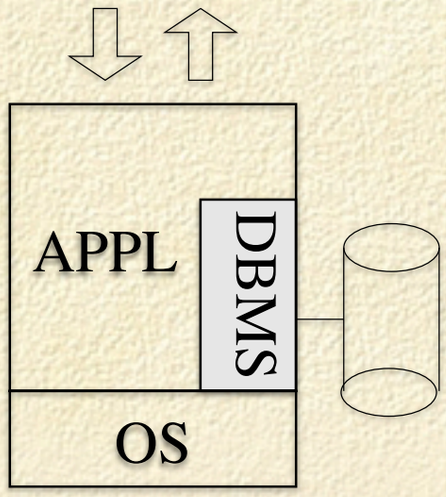
6.1 Ablaufkonsistenz: Workflow

(application systems, APPL)



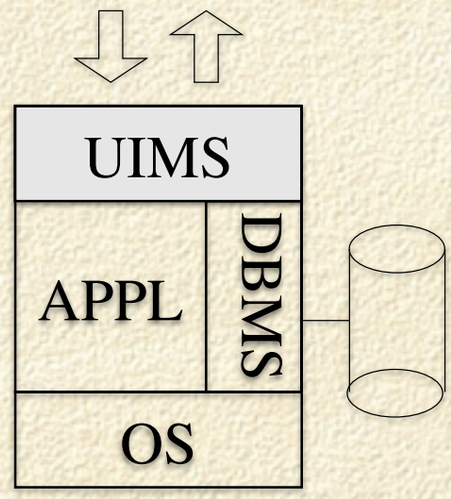
1960's

(database management systems, DBMSs)



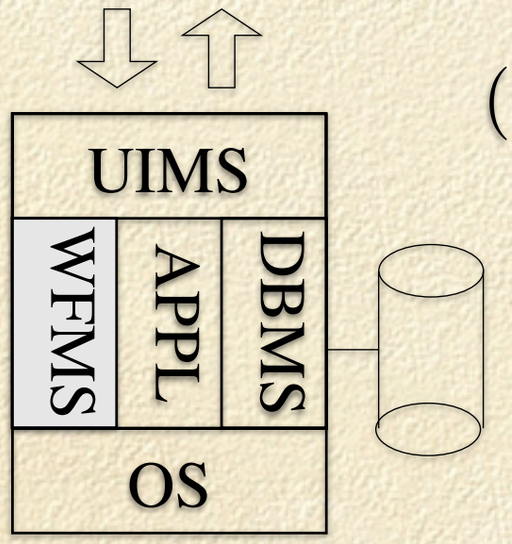
1970's

(user interface management systems, UIMSs)



1980's

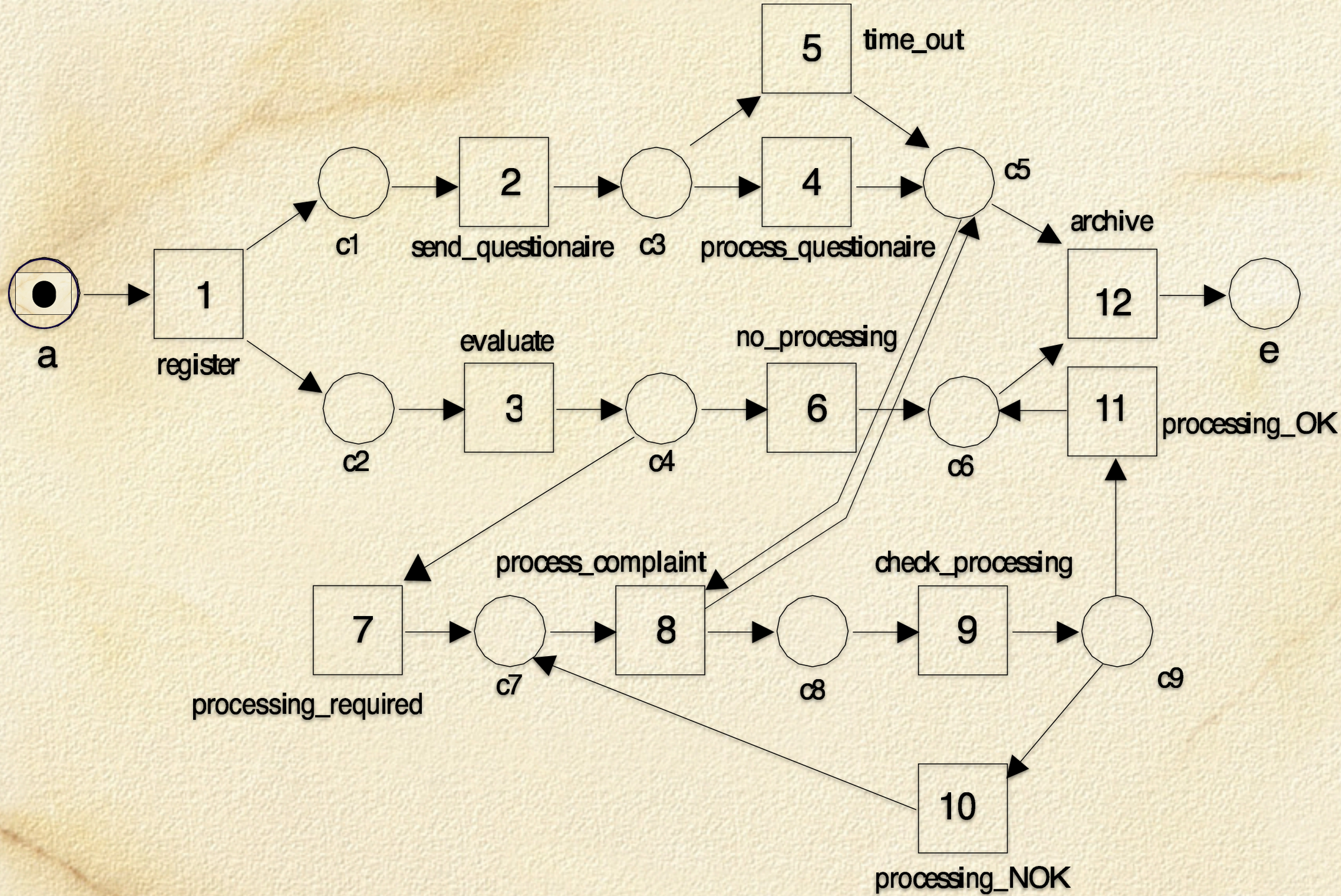
(operating system, OS)



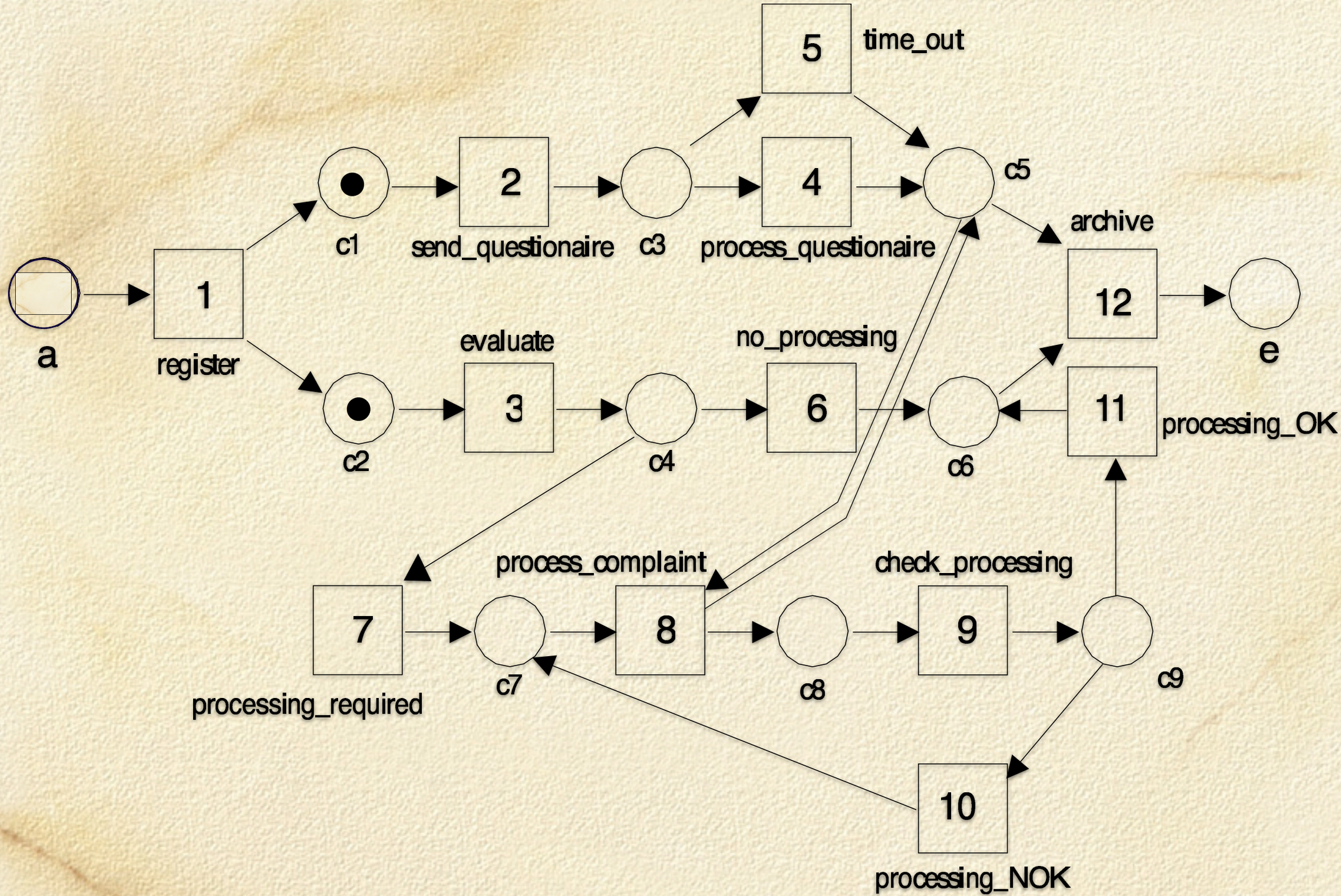
1990's

(workflow management systems, WFMSs)

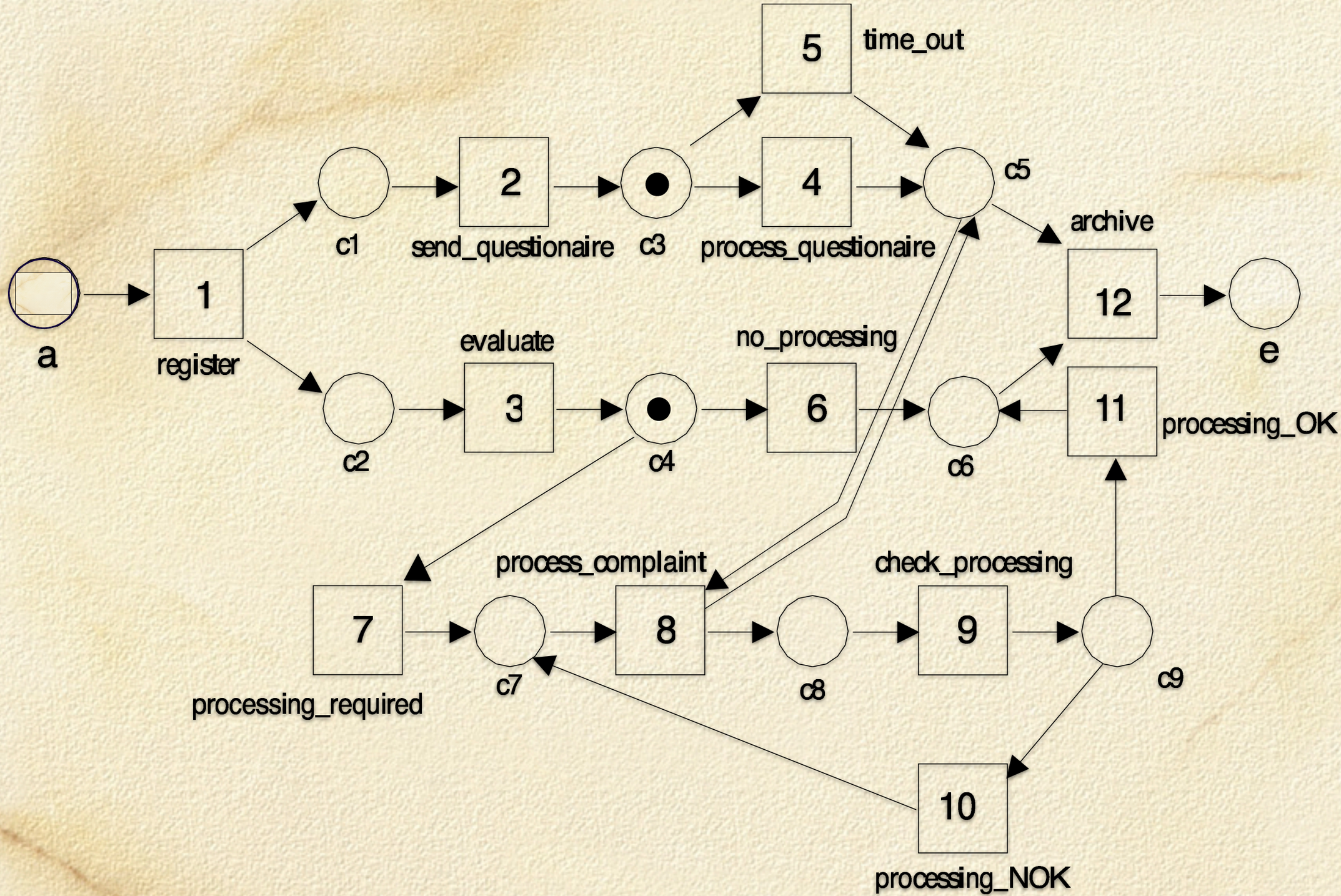
Seite 193



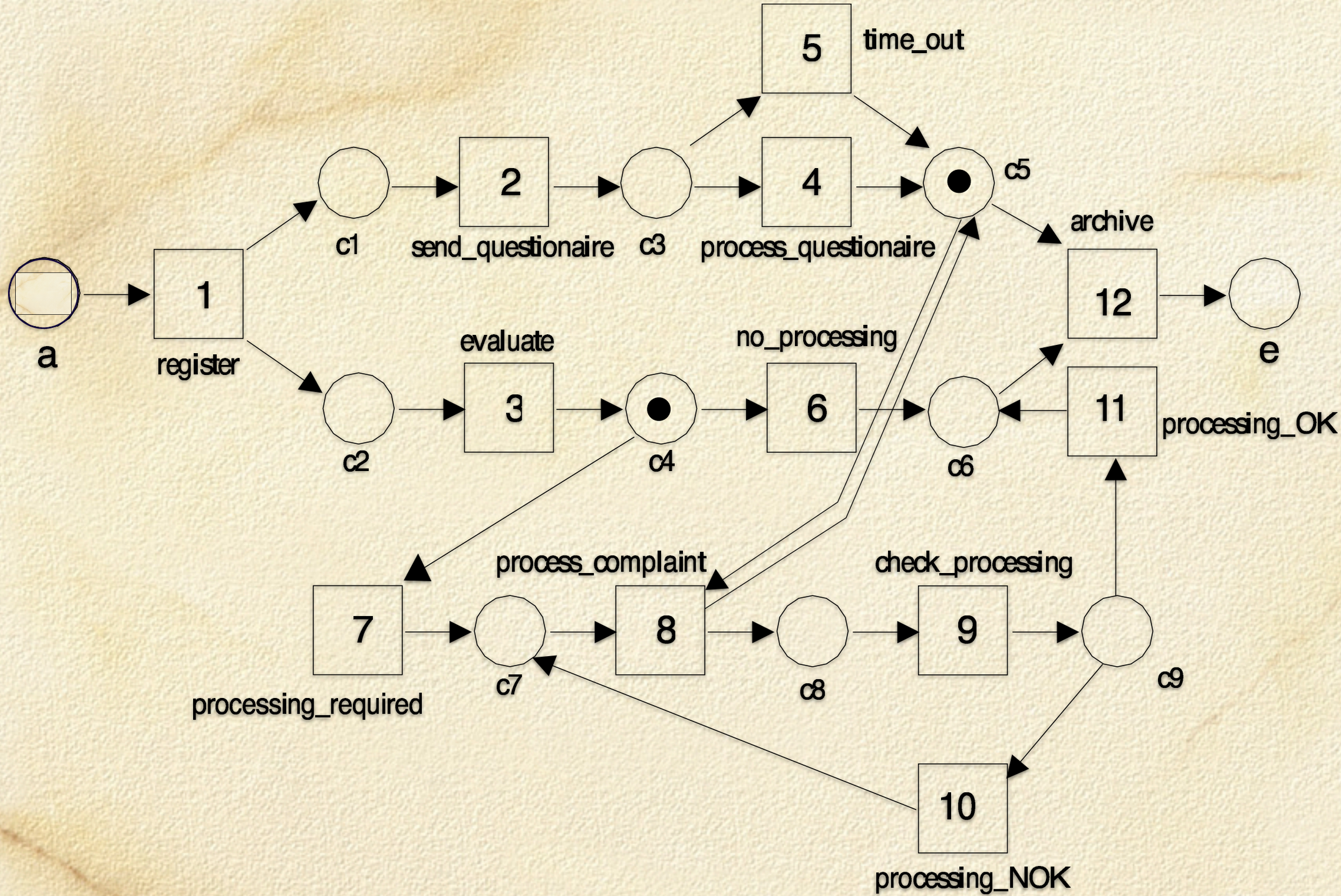
1. Aufnahme einer Beschwerde (register)



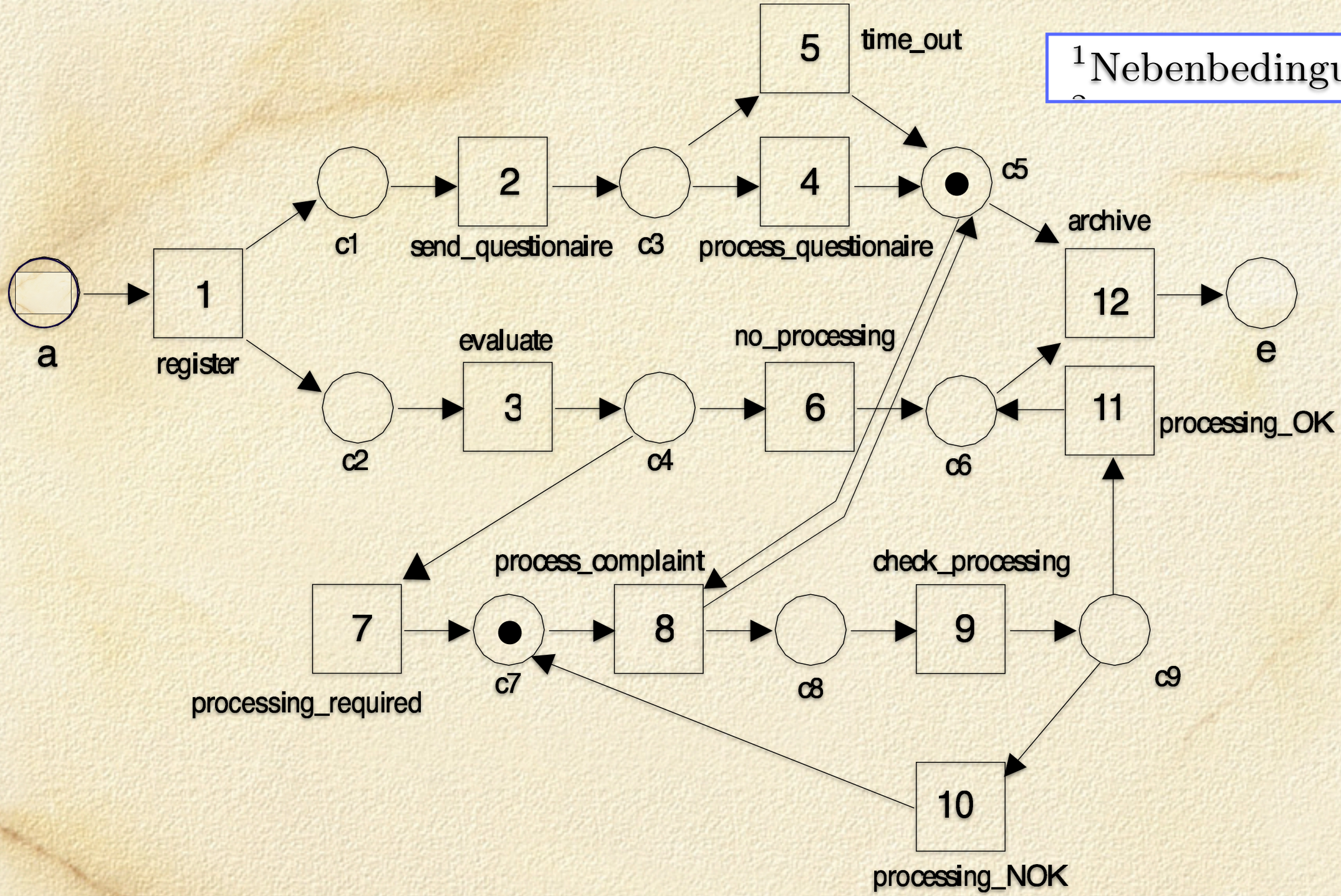
2. Fragebogen an Beschwerdeführer (send_questionnaire)
3. Bewertung (evaluate) (nebenläufig zu 2.)



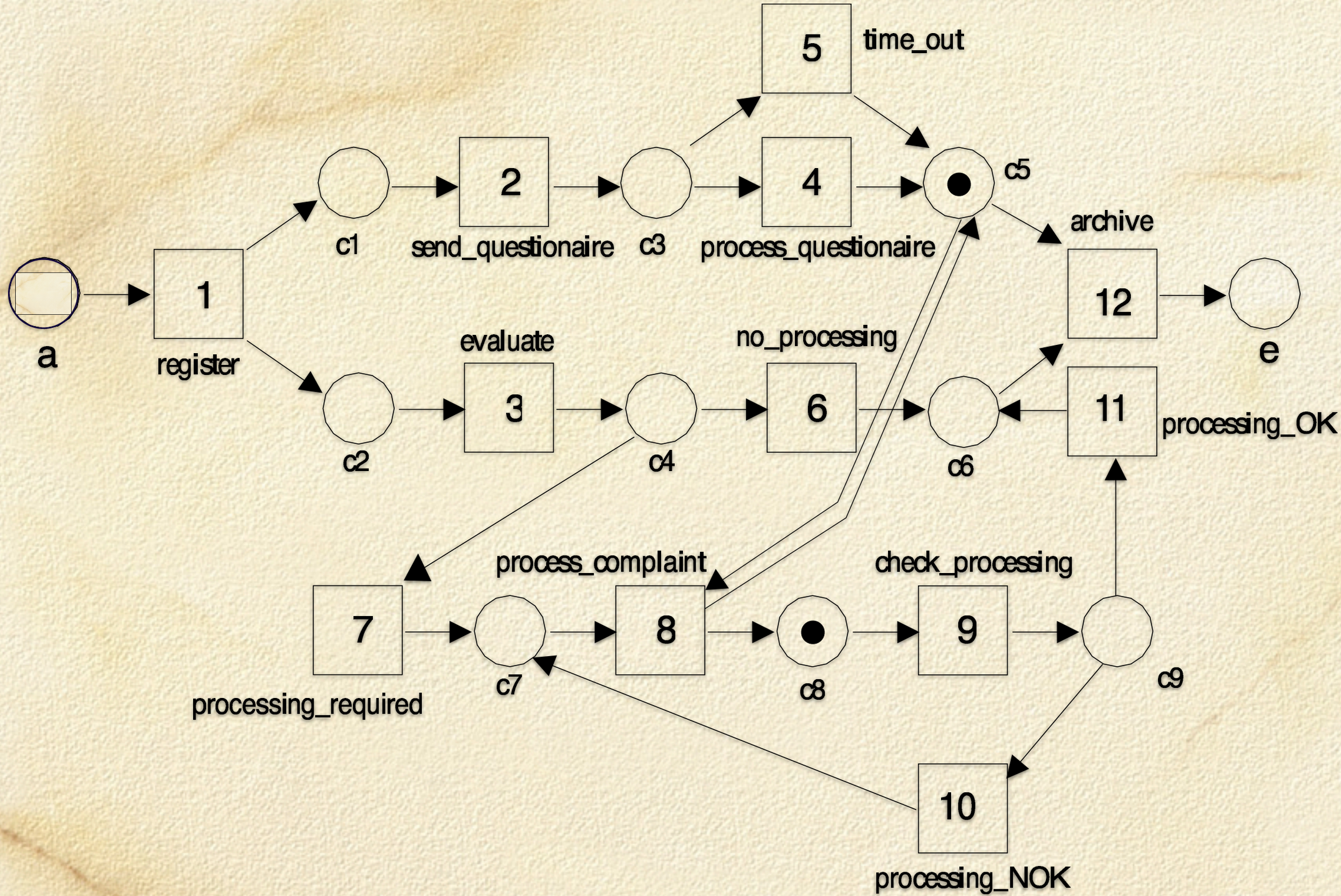
4. Fragebogenauswertung (process_questonaire), falls Rücklauf innerhalb von 2 Wochen, sonst:
5. Nichtberücksichtigung des Fragebogens (time_out).



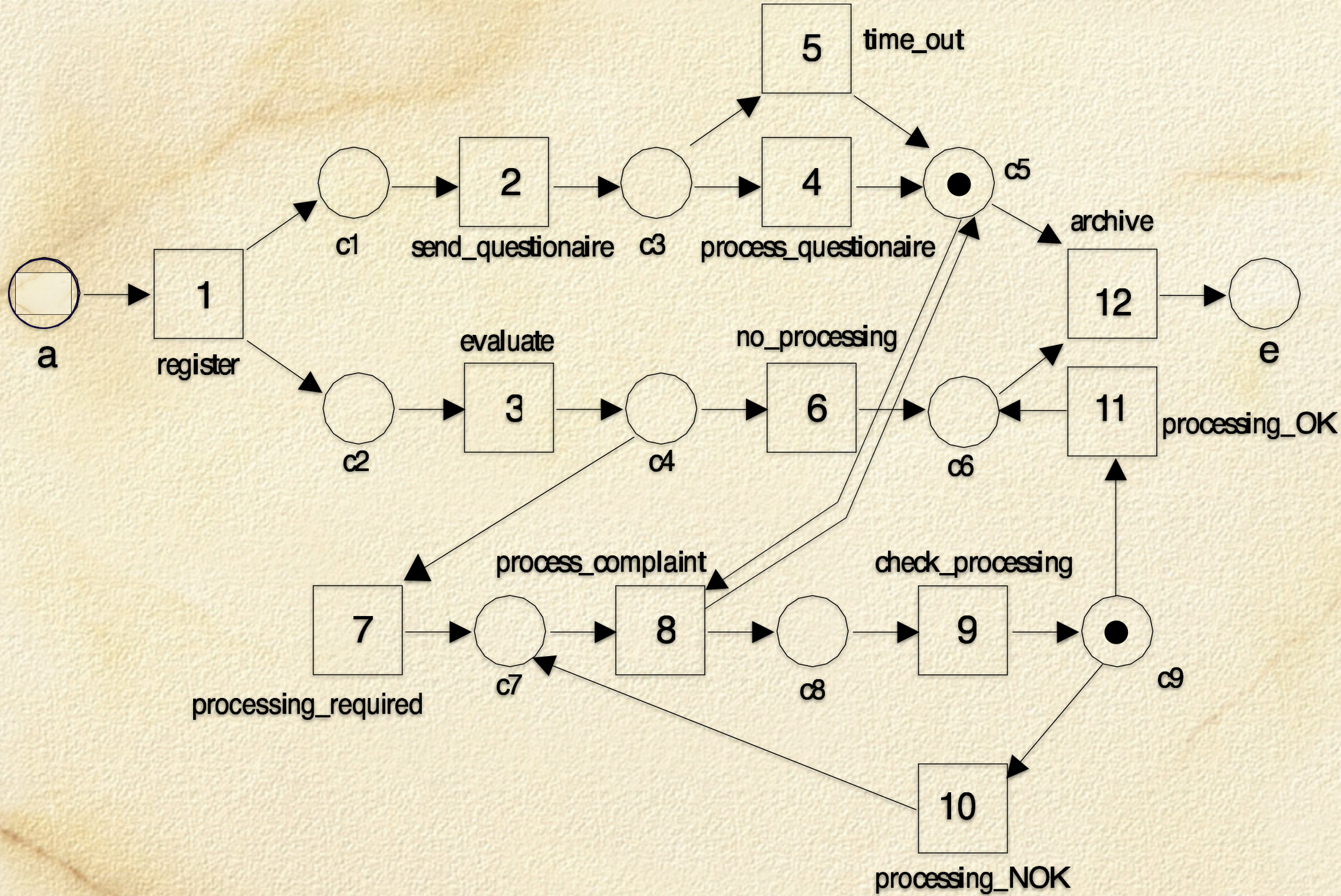
6. Je nach Ergebnis der Bewertung (3.) : Aussetzung der Bearbeitung (no_processing) oder
7. Beginn der eigentlichen Prüfung (processing_required)



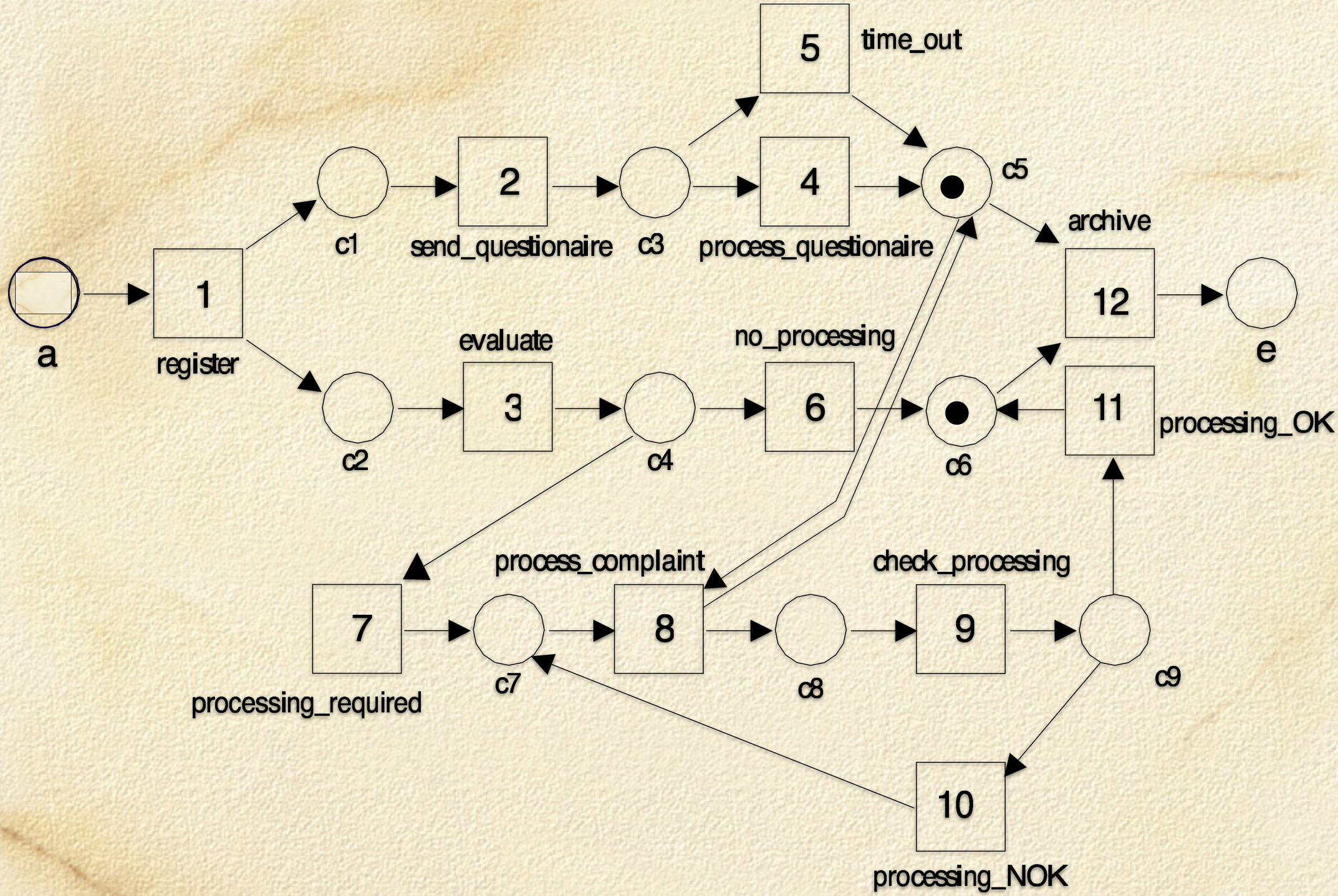
8. Bearbeitung der Beschwerde (process_complaint) unter Berücksichtigung des Fragebogens[1]



9. Bewertung der Bearbeitung (check_processing) mit dem Ergebnis



10. erneute Prüfung (processing_nok) oder
11. Abschluss (processing_ok)



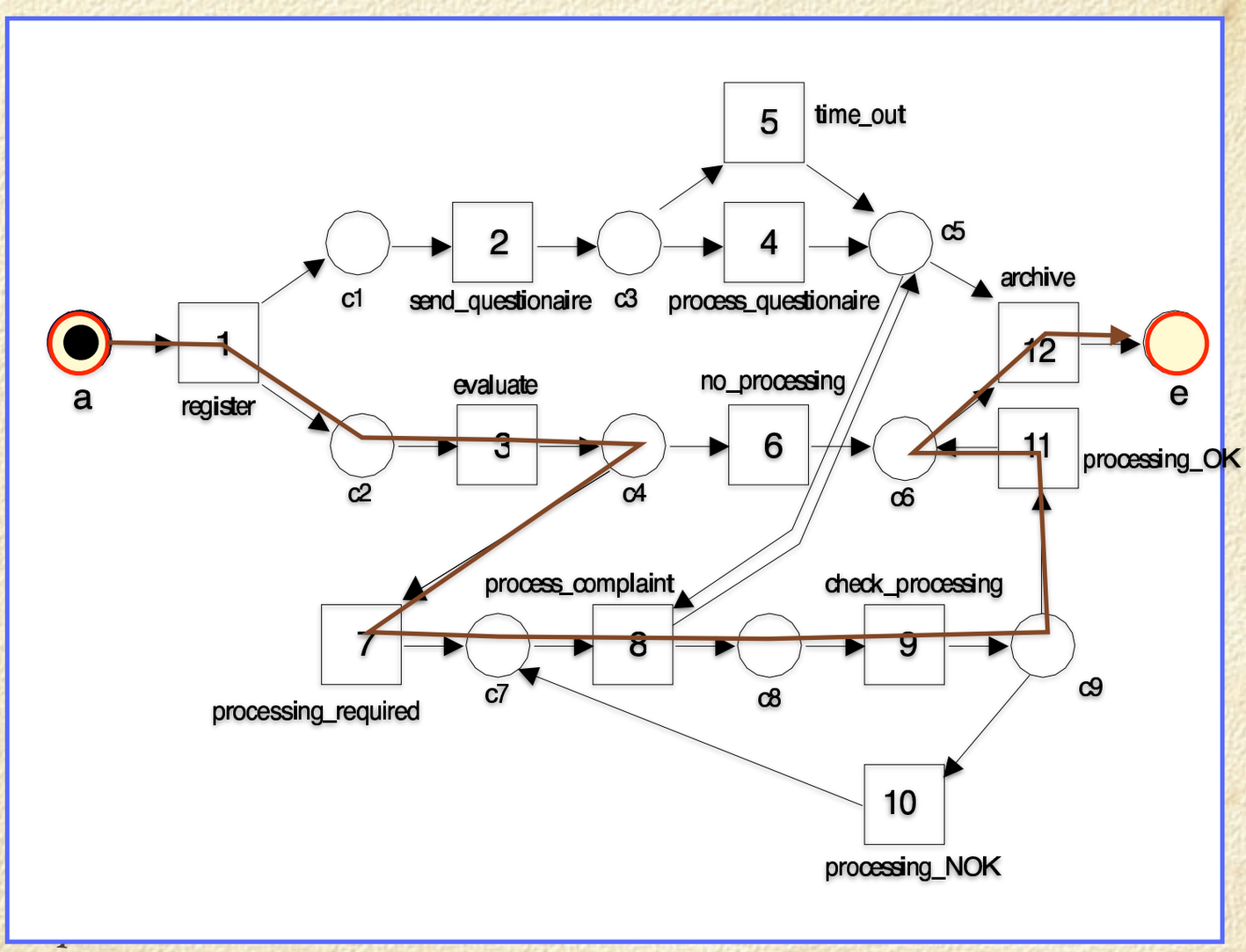
12. Ablage (archive)

Definition 6.1 *Ein P/T-Netz* $\mathcal{N} = (P, T, F, \mathbf{m}_a)^2$ *heißt* Workflow-Netz *(WF-Netz), falls*

b) alle Plätze und Transitionen auf Pfaden zwischen a und e liegen und

c) Anfangsmarkierung $\mathbf{m}_a$

Endmarkierung $\mathbf{m}_e$



Typische Strukturen („patterns[3]“)

`http://www.workflowpatterns.com/patterns`

Pattern 2 (Parallel Split)

FLASH animation of Parallel Split pattern

Description

The divergence of a branch into two or more parallel branches each of which execute concurrently.

Synonyms

AND-split, parallel routing, parallel split, fork.

Examples

After completion of the *capture enrolment* task, run the *create student profile* and *issue enrolment confirmation* tasks simultaneously.

When an *intrusion alarm* is received, trigger the *despatch patrol* task and the *inform police* task immediately.

Once the customer has paid for the goods, pack them and issue a receipt.

Motivation

The *Parallel Split* pattern allows a single thread of execution to be split into two or more branches which can execute tasks concurrently. These branches may or may not be re-synchr

Overview

Figure 2 illustrates the implementation of the *Parallel Split*. After task A has completed, two distinct threads of execution are initiated and tasks B and C can proceed concurrently.

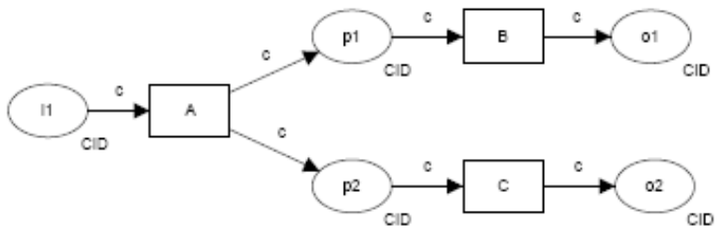


Figure 2: Parallel split pattern

Context

There are no specific context conditions for this pattern.

Implementation

The *Parallel Split* pattern is implemented by all of the offerings examined. It may be depicted either *explicitly* or *implicitly* in process models. Where it is represented explicitly, a *Split* with one incoming edge and two or more outgoing edges. Where it is represented implicitly, this can be done in one of two ways: either (1) the edge representing control branches or (2) the task after which the *Parallel Split* occurs has multiple outgoing edges which do not have any conditions associated with them or where it does these conditions alwa

Evaluation Criteria

Full support for this pattern is demonstrated by the provision of a construct (either implicit or explicit) that allows the thread of control at a given point in a process to be s

Product Evaluation

To achieve a + rating (direct support) or a +/- rating (partial support) the product should satisfy the corresponding evaluation criterion of the pattern. Otherwise a - rating

Product/Language	Version	Score	Motivation
Staffware	10	+	Supported through a step construct that has multiple outgoing arcs.
Websphere MQ	3.4	+	Supported through multiple outgoing arcs from an activity.
FLOWer	3.51	+	Nodes in static, dynamic and sequential subplans have an AND-split semantics.
COSA	5.1	+	Supported by multiple outgoing arcs from an activity. None of the arcs have tran
iPlanet	3.0	+	Supported by multiple outgoing routers from an activity
SAP Workflow	4.6c	+	Directly supported. SAP allows for structured parallel processes, using the fork c Since there has to be a one-to-one correspondence between splits and joins, som them structured.
FileNet	3.5	+	Directly supported by a step where all outgoing routes are unconditional.
BPEL	1.1	+	Supported by <flow> construct.
Websphere Integration Developer	6.0	+	Supported by the <flow> activity.
Oracle BPEL	10.1.2	+	Supported by the <flow> construct.
BPMN	1.0	+	Supported by AND-split gateway
XPDL	2.0	+	Supported by the AND-split construct
UML ADs	2.0	+	Supported by the ForkNode construct. It may also be represented implicitly by jo actions or activities.
EPC (implemented by ARIS toolset 6.2)		+	Supported by the AND-split connector.
jBPM	3.1.4	+	jBPM supports parallel split through: i) the *Fork Node* construct, which splits a tread to multiple treads running in para ii) by defining the tasks to be run in parallel within the same *Task Node*.
OpenWFE	1.7.3	+	OpenWFE implements parallel split through the construct <concurrence>
Enhydra Shark	2	+	Enhydra Shark supports parallel split through i) an *activity* node with a *transition restriction* <Split Type = "AND"> specifying ii) a *routing activity* node with a *transition restriction* <Split Type = "AND"> spe contrast to an activity, a routing activity does not implement any actions.

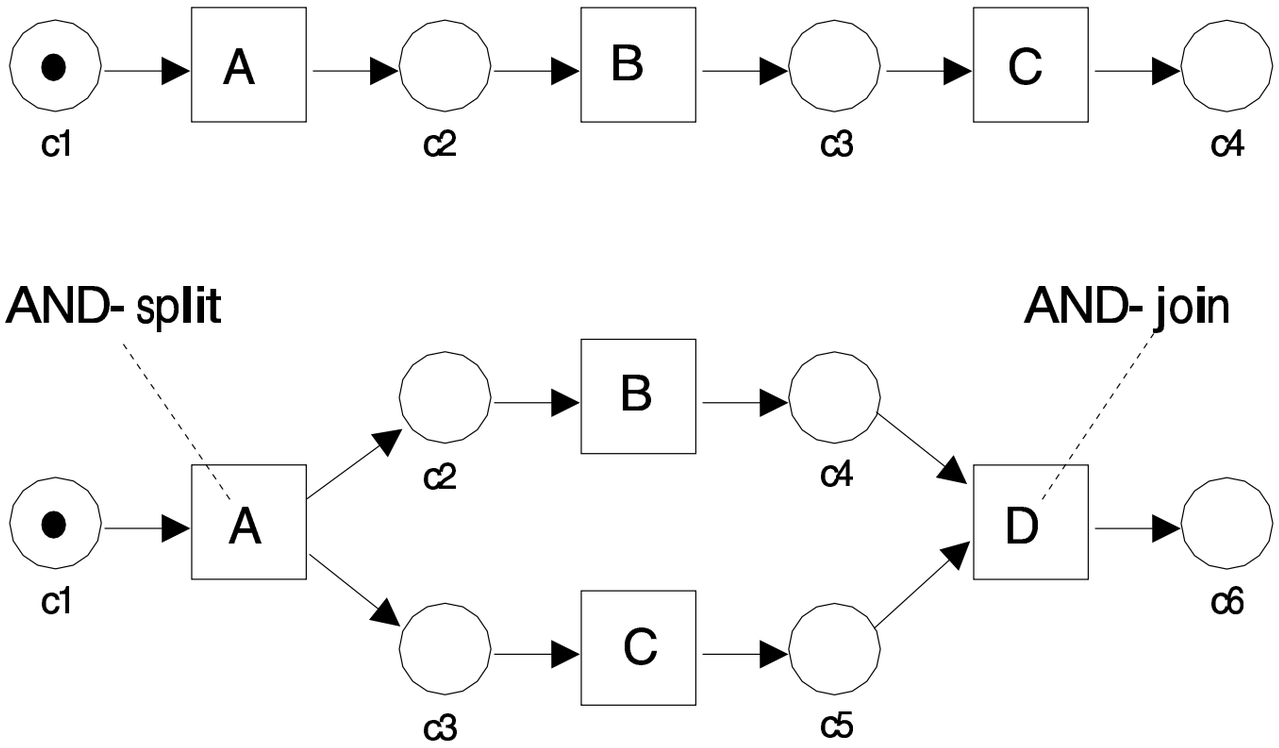


Abbildung 6.4 Sequentielle und nebenläufige Bearbeitung

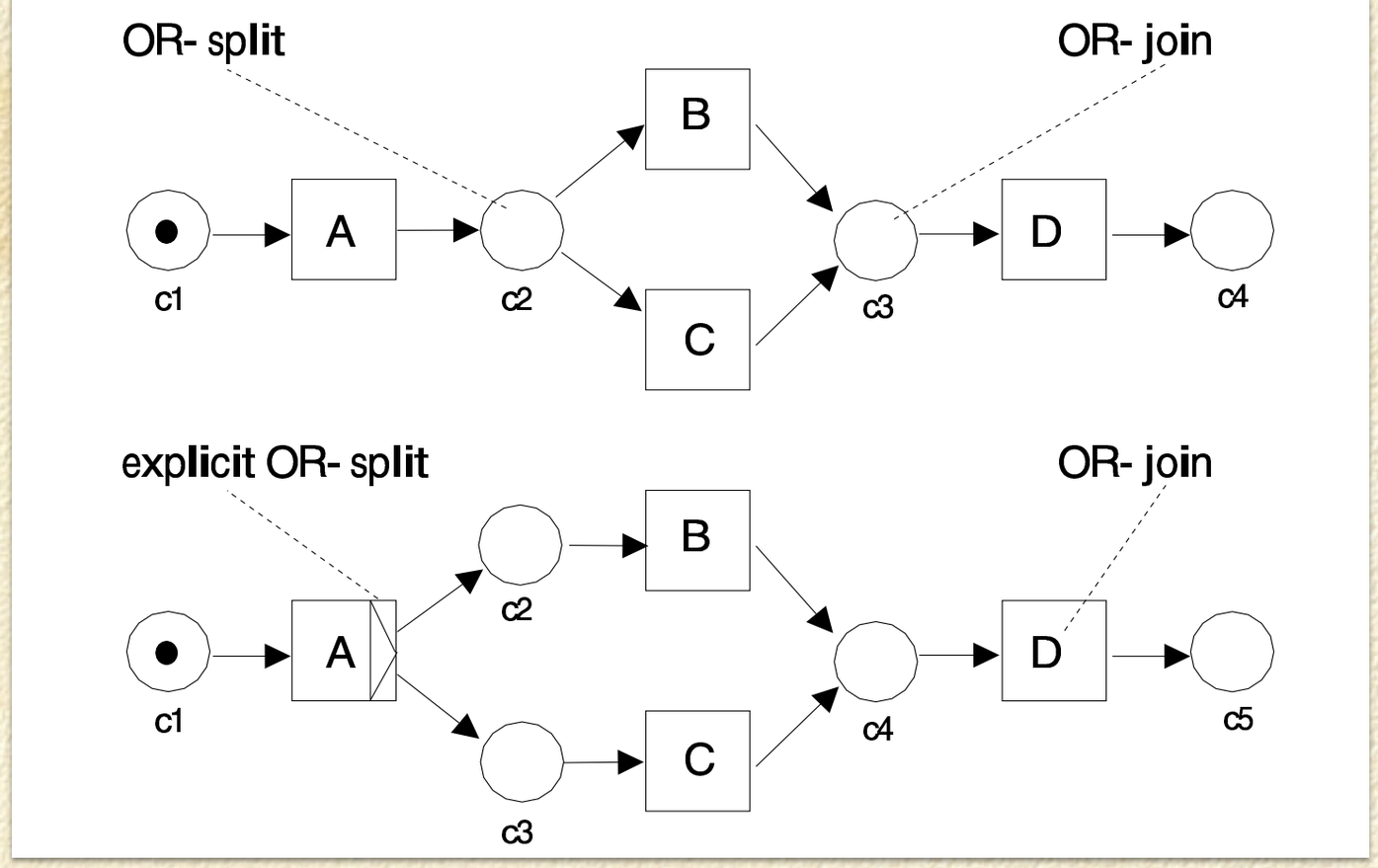


Abbildung 6.5 Alternative Bearbeitung mit und ohne expliziten Testbedingungen

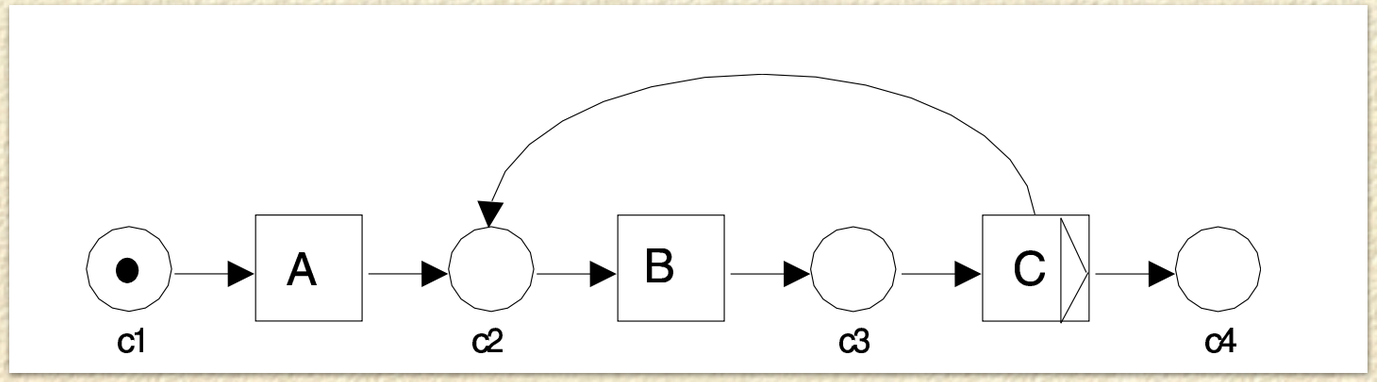


Abbildung 6.6 Iteration

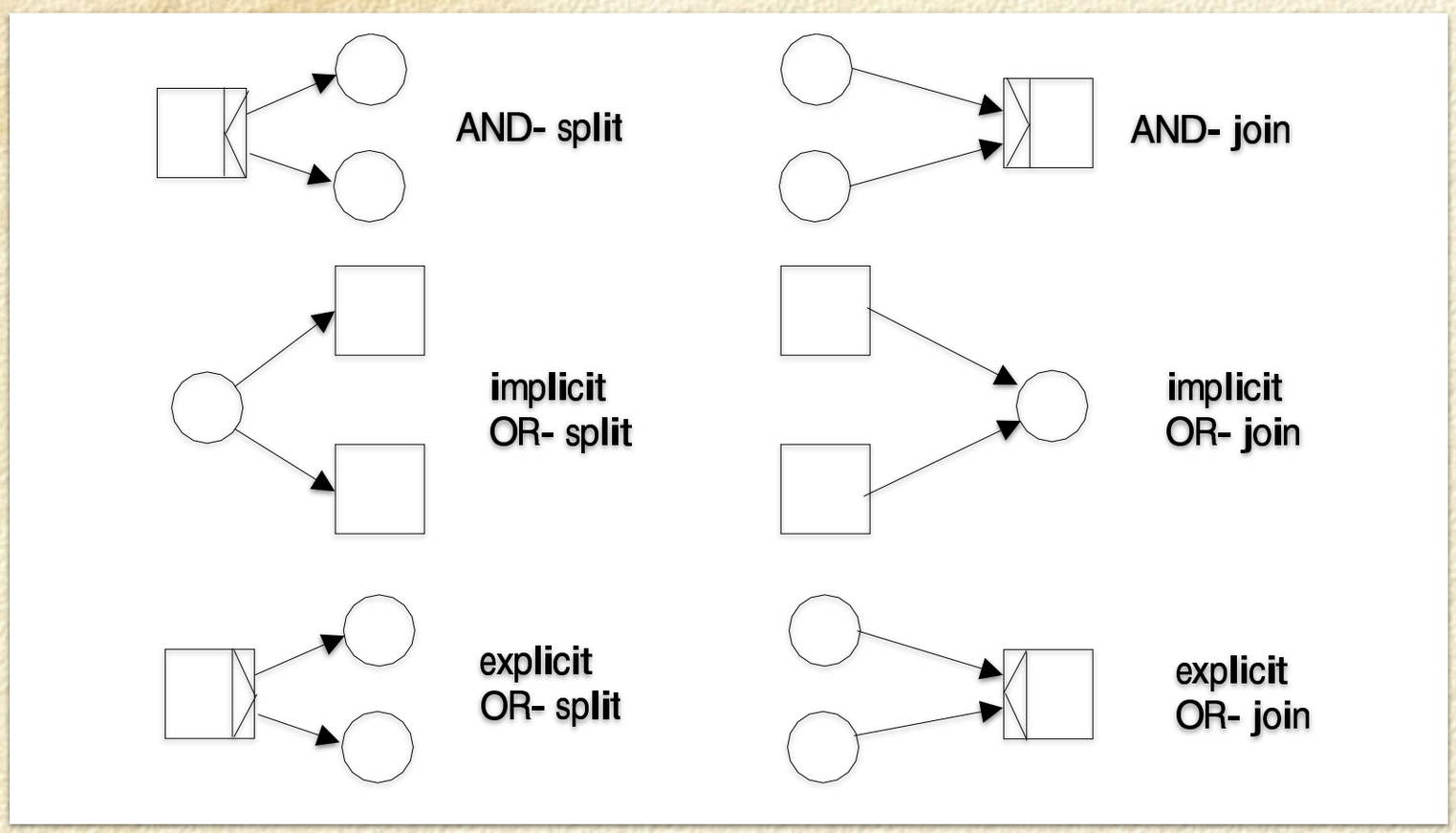


Abbildung 6.7 Graphische Symbole für Verzweigung

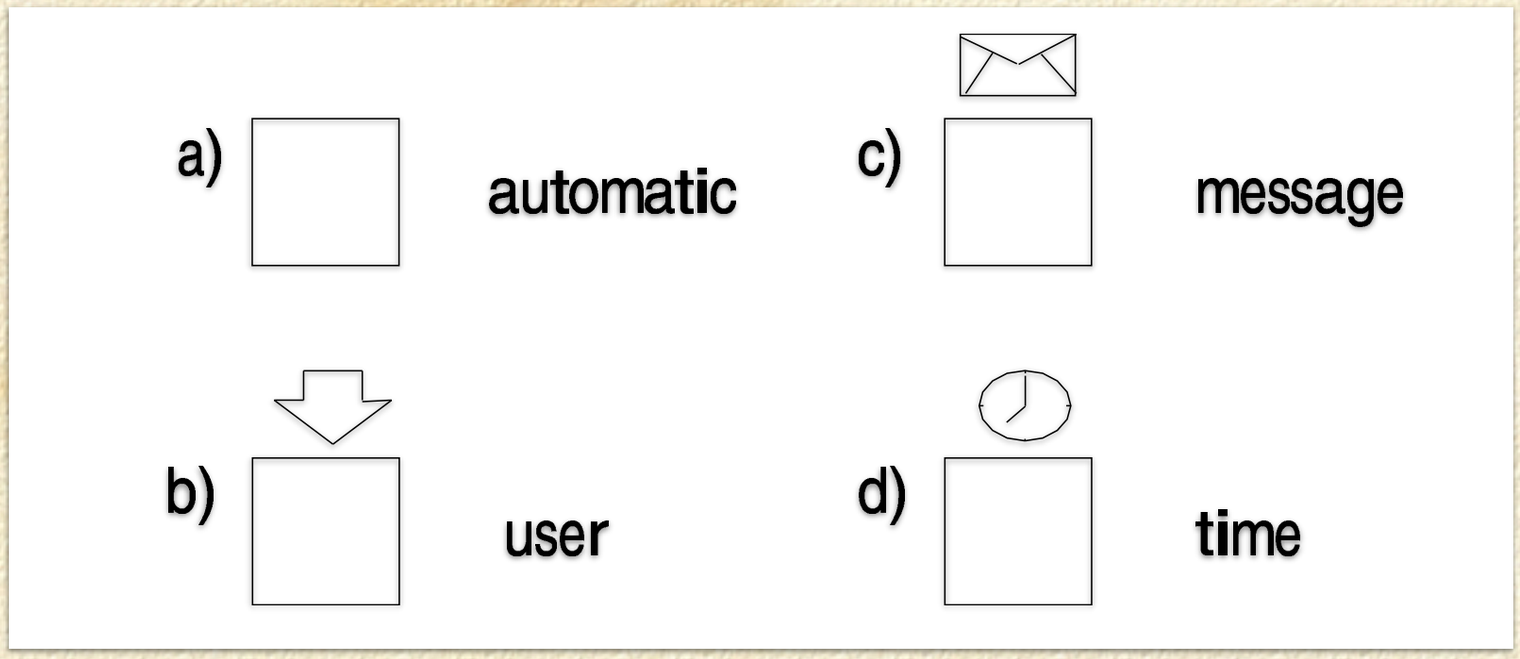


Abbildung 6.8 Trigger eines Workflowsystems

a) Automatisch (keine externe Eingabe notwendig).

b) Benutzer (user) : ein Bearbeiter oder Benutzer oder eine Funktionseinheit nimmt einen Auftrag an und bearbeitet ihn.

c) Nachricht (message) : eine Nachricht von außen wird benötigt (Brief, Anruf, e-mail, Fax).

d) Zeit (time) : es besteht eine Zeitbedingung für die Bearbeitung.

Die Erfahrung aus der Praxis zeigt, dass

- Workflow-Prozesse oft nicht richtig verstanden werden (Mehrdeutigkeit, Widersprüche, Verklemmungen),

- allein schon die Modellierung durch Petrinetze Mängel aufdeckt und

- bei fertiggestellten Petrinetz-Modellen von Workflow-Systemen Mängel durch strukturelle Untersuchungen aufgedeckt oder durch Werkzeuge (automatisch) gefunden werden.

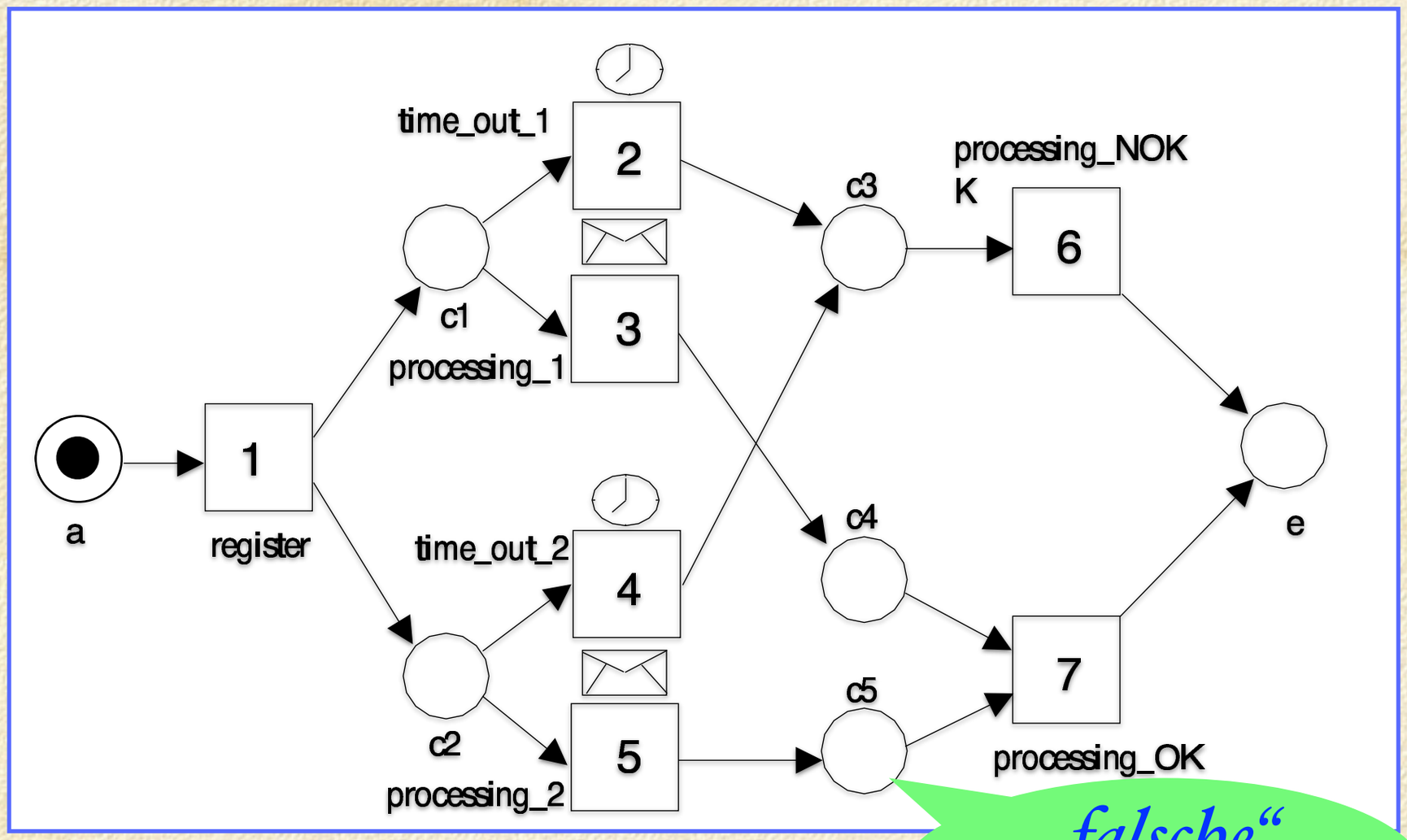


Abbildung 6.9 Problematisches Workflownetz
tung

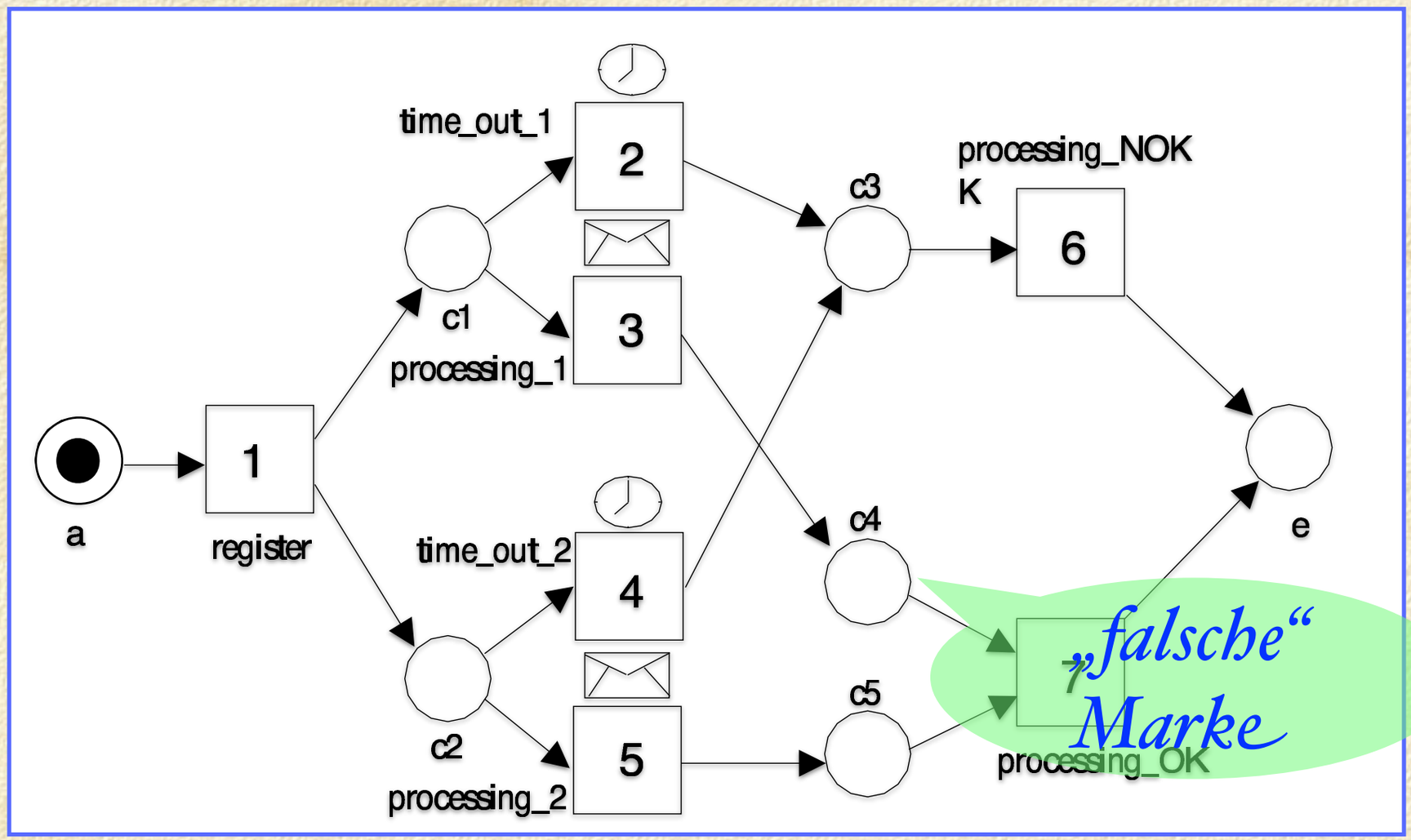


Abbildung 6.9 Problematisches Workflownetz für Beschwerdebearbeitung

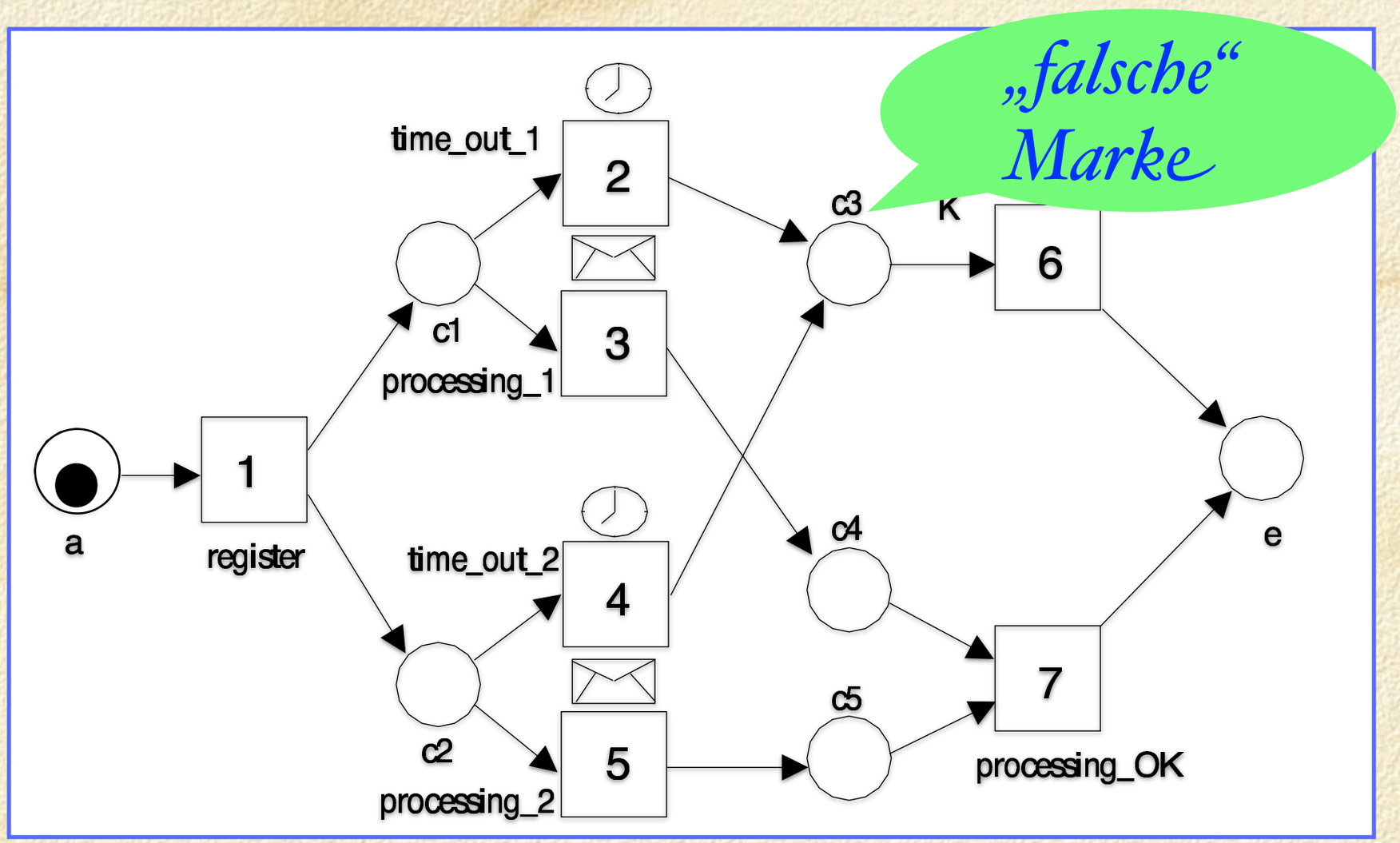


Abbildung 6.9 Problematisches Workflownetz für Beschwerdebearbeitung

Definition 6.2 *Ein WF-Netz* $\mathcal{N} = (P, T, F, \mathbf{m}_a)$ *heißt* korrekt, *falls gilt :*

a) $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N})\ \exists w \in T^* : \mathbf{m} \xrightarrow{w} \mathbf{m}_e$

b) $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m}(e) \geq 1 \Rightarrow \mathbf{m} = \mathbf{m}_e$

c) $\forall t \in T\ \exists \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m} \xrightarrow{t}$

a) Aus jeder erreichbaren Markierung ist eine ordnungsgemäße Termination möglich.

b) Genau eine Marke in dem Endplatz e ist die einzige Möglichkeit zu terminieren.

c) Jede Transition kann in einer möglichen Schaltfolge schalten, denn sonst wäre sie nutzlos.

WF-Netz $\mathcal{N}$ korrekt,

⇕

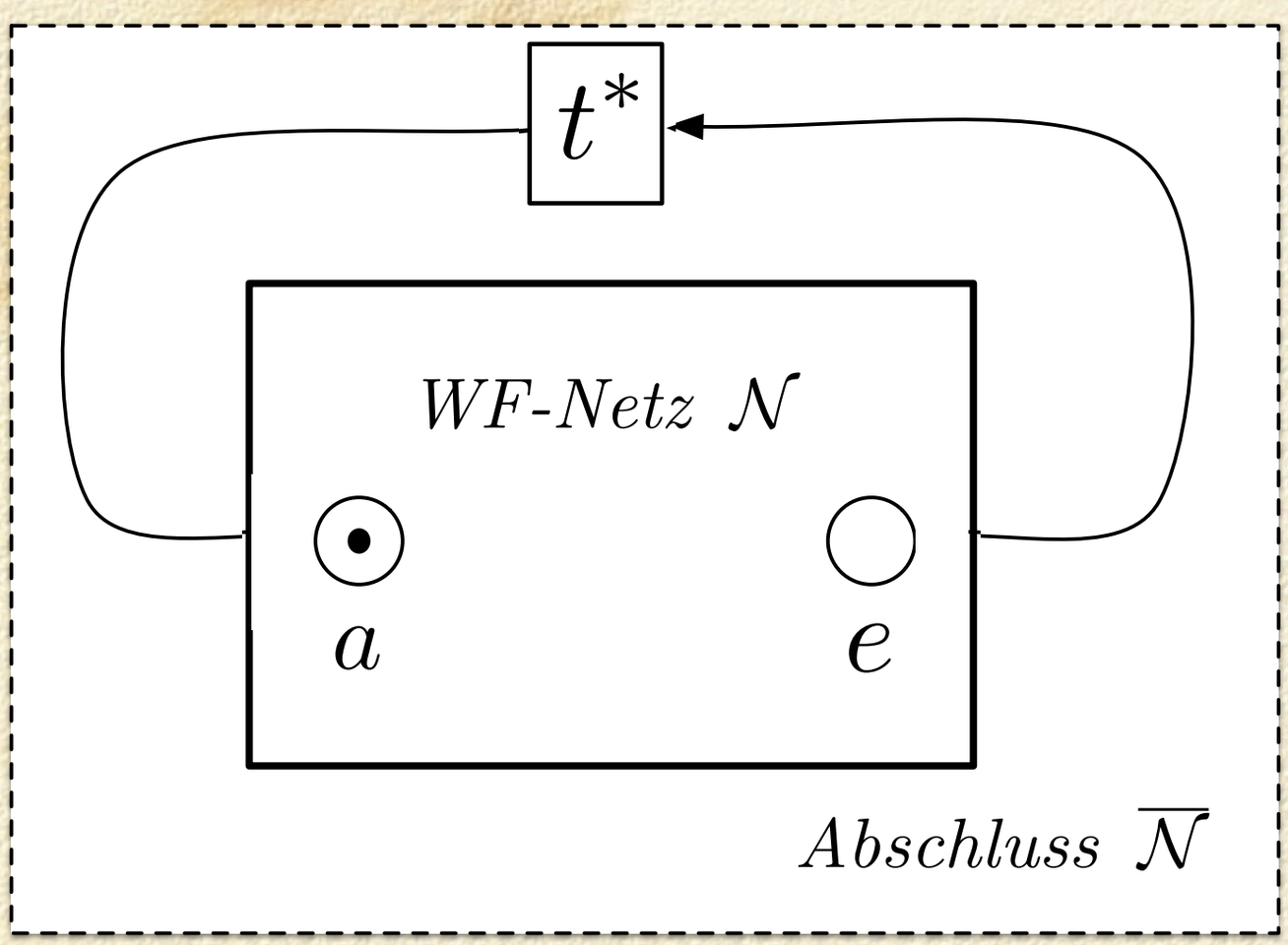
Abschluss von N

$\overline{\mathcal{N}}$ *lebendig und beschränkt*

Jede Transition behält die Eigenschaft potenziell zu schalten

Es gibt eine obere Schranke für die Markenzahl

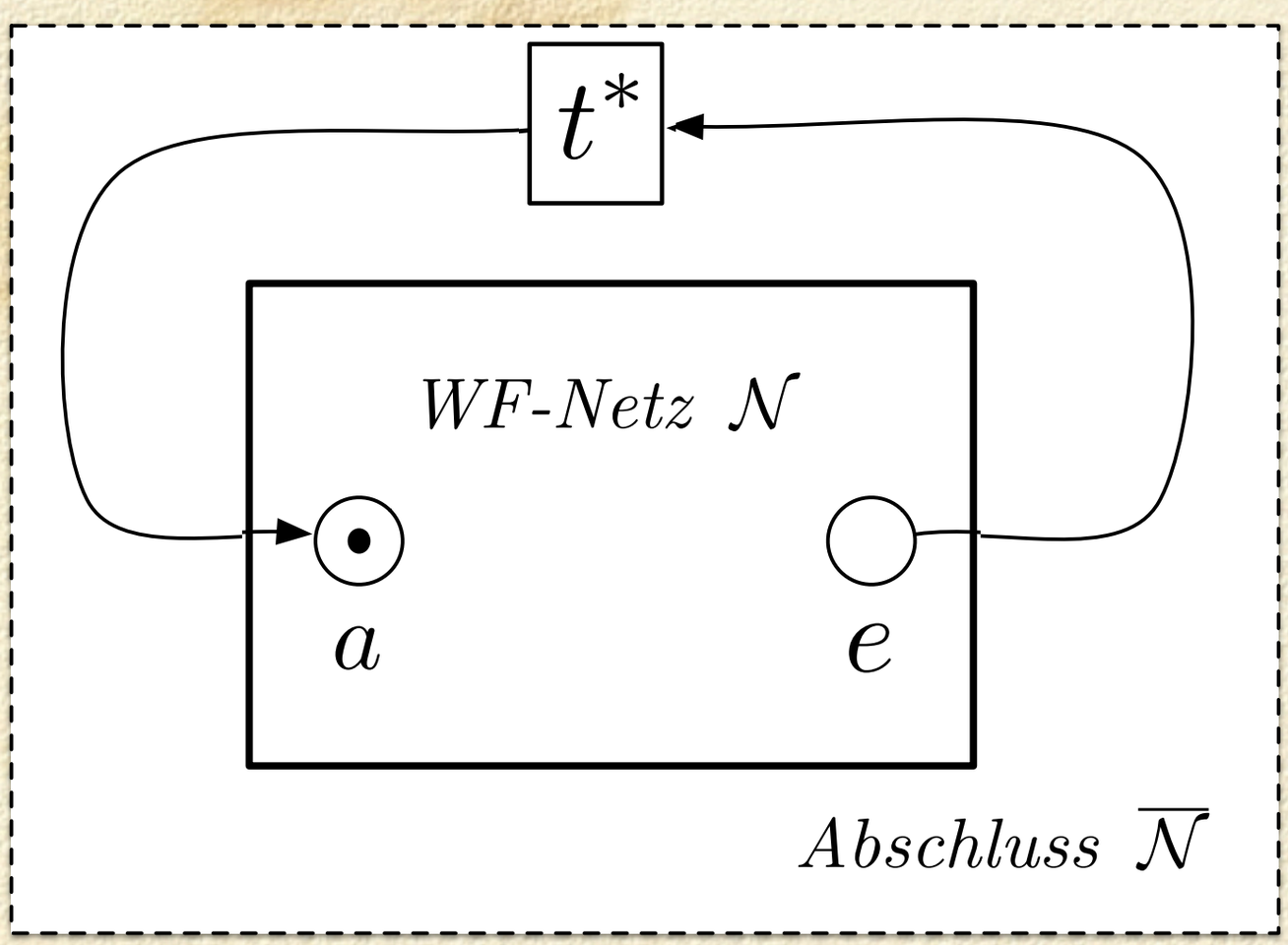
Eigenschaften, für die Algorithmen bestehen.



Definition 6.3 *Für ein WF-Netz* $\mathcal{N} = (P, T, F, \mathbf{m}_a)$ *mit Anfangsplatz* a *und Endplatz* e *heißt* $\overline{\mathcal{N}} = (P, T', F', \mathbf{m}_a)$ *der* Abschluss *von* $\mathcal{N}$*, falls gilt :*

a) $T' := T \cup \{t^*\}$ *für eine neue Transition* $t^* \notin T$

b) $F' := F \cup \{(e, t^*), (t^*, a)\}$



Satz 6.4 *Ein WF-Netz $\mathcal{N}$ ist genau dann korrekt, wenn sein Abschluss $\overline{\mathcal{N}}$ lebendig und beschränkt ist.*

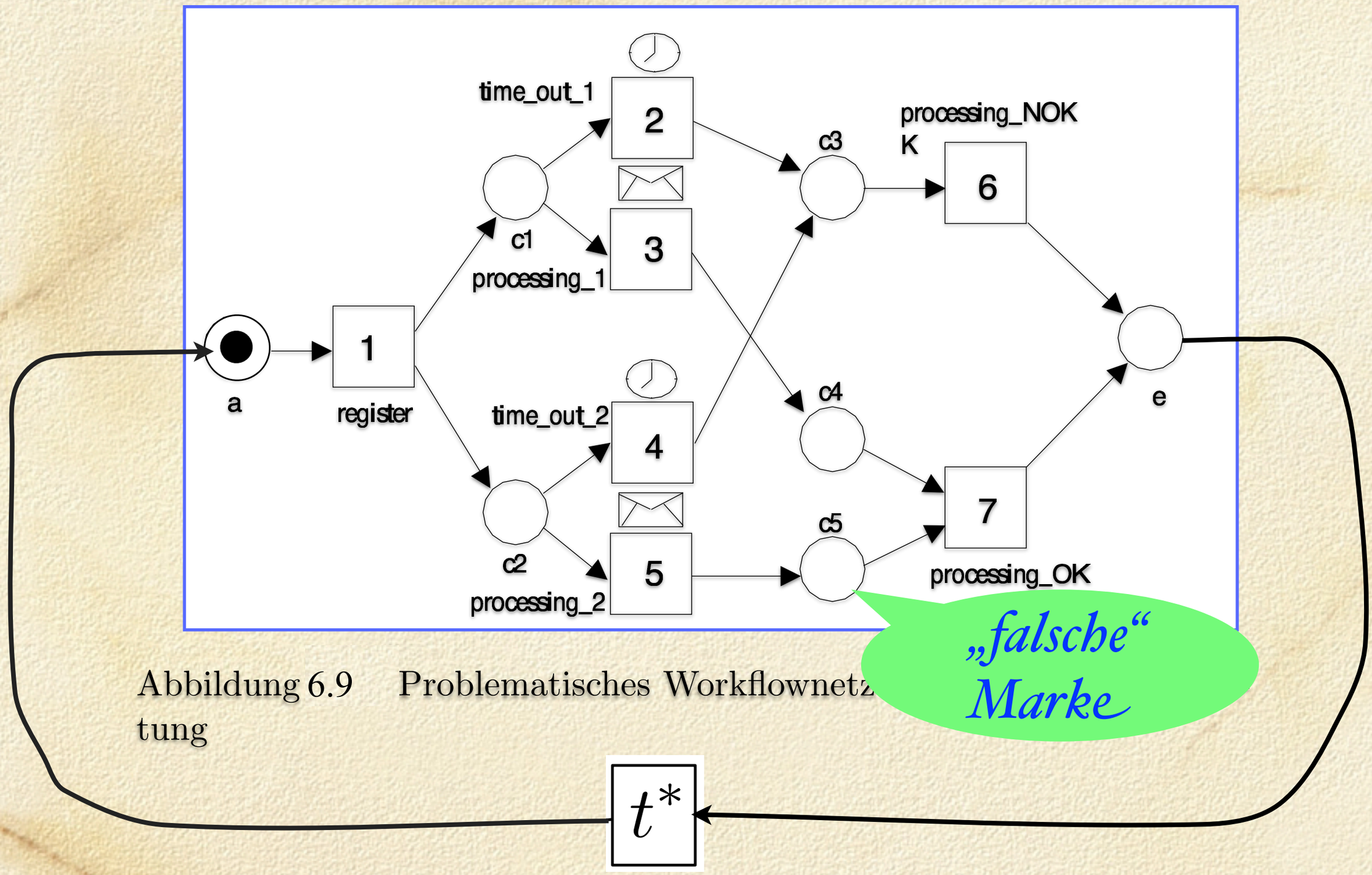


Abbildung 6.9 Problematisches Workflownetz
tung

?

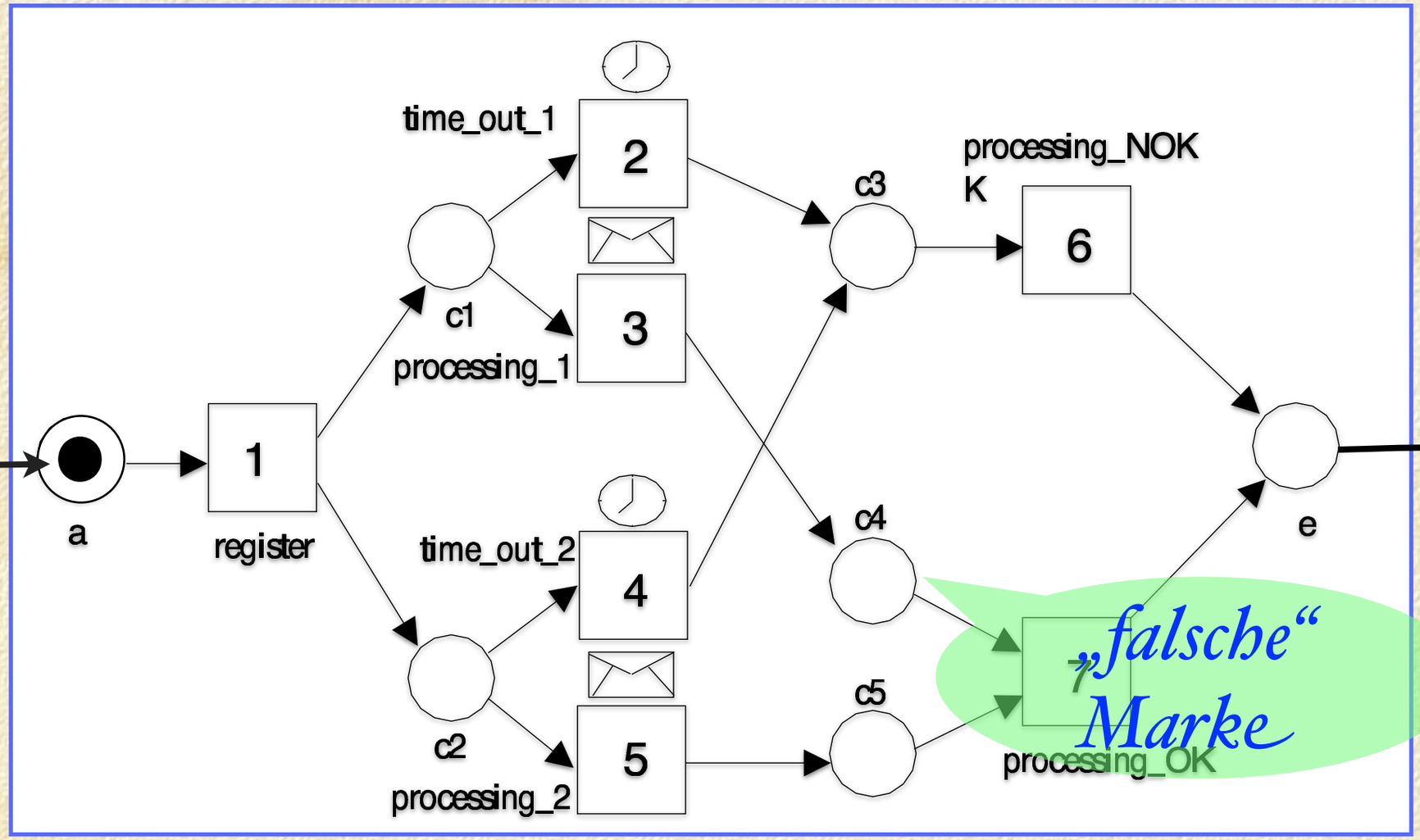
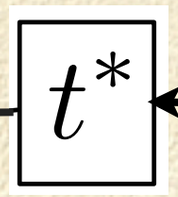


Abbildung 6.9 Problematisches Workflownetz für Beschwerdebearbeitung



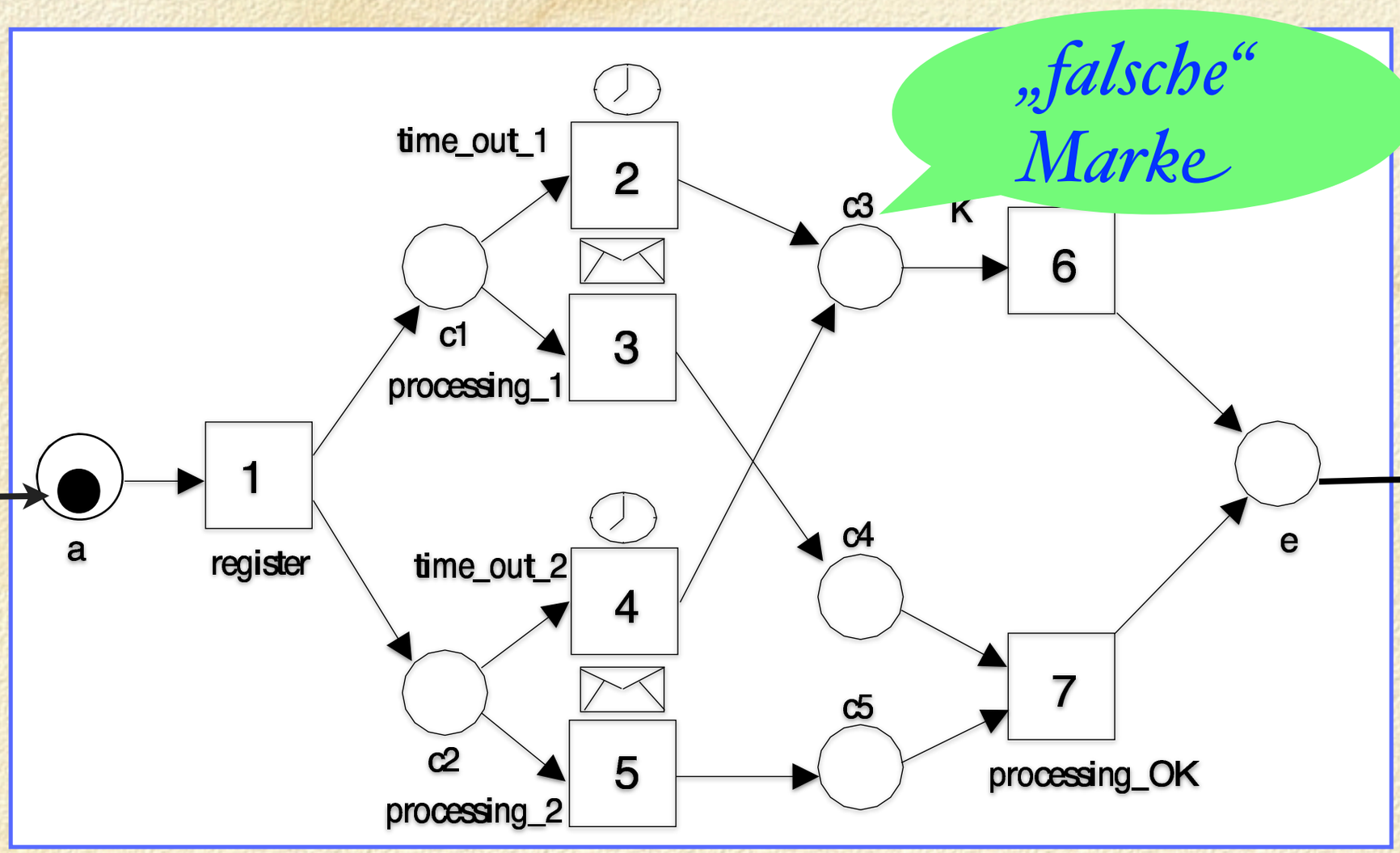
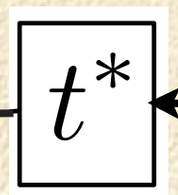


Abbildung 6.9 Problematisches Workflownetz für Beschwerdebearbeitung



?

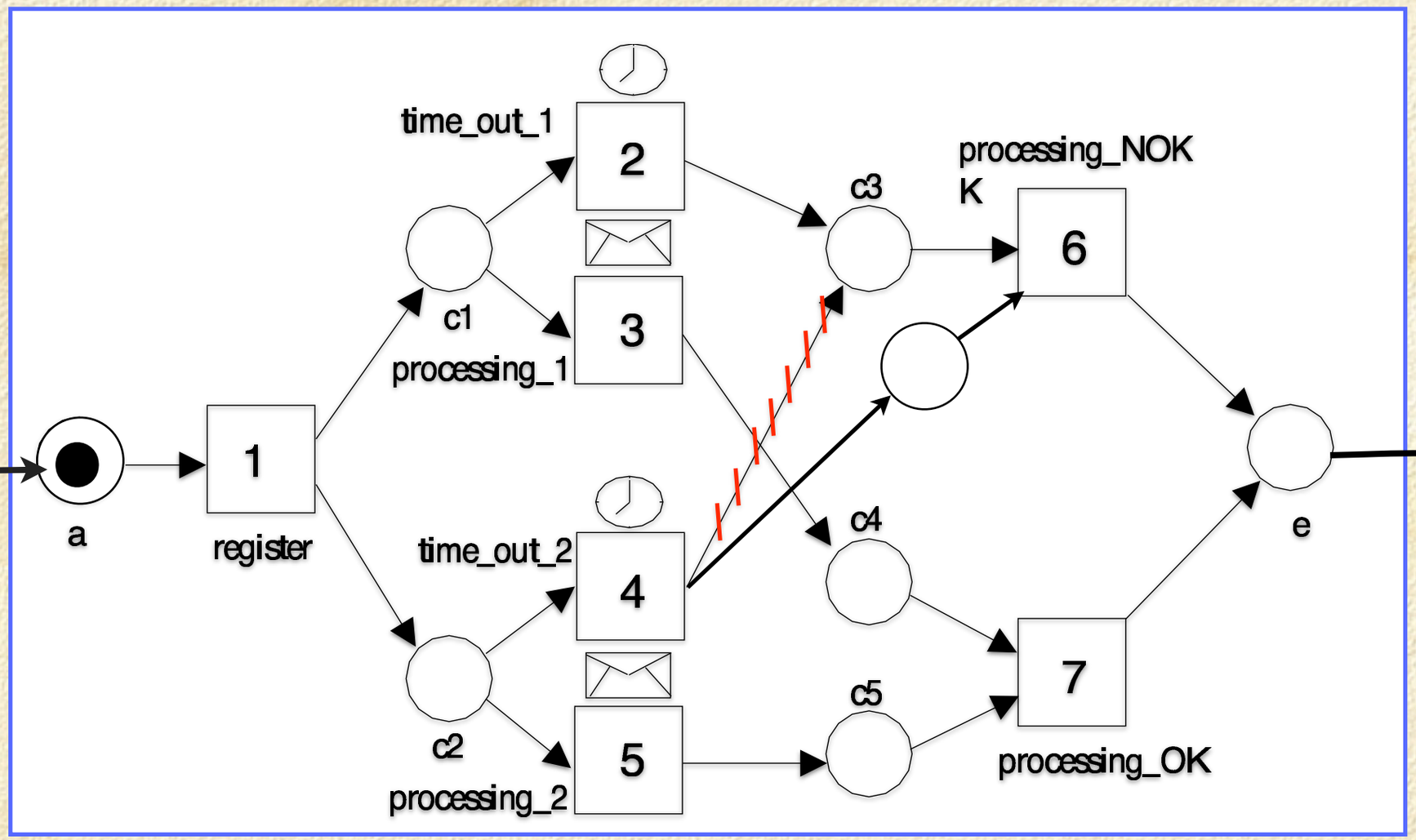
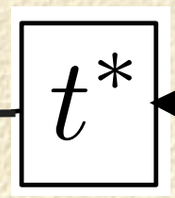


Abbildung 6.9 Problematisches Workflownetz für Beschwerdebearbeitung



http://www2.informatik.hu-berlin.de/top/lehre/petriweb/

Carl Adam Petri Memorial Symposium

Humboldt-Universität zu Berlin

Friday, February 4, 2011
Senatssaal, Unter den Linden 6, Berlin

Carl Adam Petri was one of the most influential computer scientists of our time. He was a true visionary who pioneered the scientific modeling of discrete concurrent systems. His seminal Ph.D. thesis from 1962 outlined a whole research program to set up new foundations for computer science. The thesis was genuinely revolutionary, as what is was proposing went against all the current trends in computer science. Very few scientists recognized then the brilliancy of his way of thinking. But today, "Petri Nets" stands for a very broad framework for construction and analysis of concurrent and distributed systems. Although a big number of other models have been developed in the meantime, Petri Nets remain a central model and they are often used as a yardstick for other models of concurrency.

Carl Adam Petri passed away on July 2, 2010. His loss is felt deeply by friends and colleagues around the world.

This symposium commemorates the life and work of this great scientist. The speakers will present many facets of the scientific influence of Petri, and some of them will also reflect on personal contacts/friendships with him.

Organizing Committee: Wolfgang Reisig, Grzegorz Rozenberg, Bernd Krämer, Rüdiger Valk

last modified 10-11-5

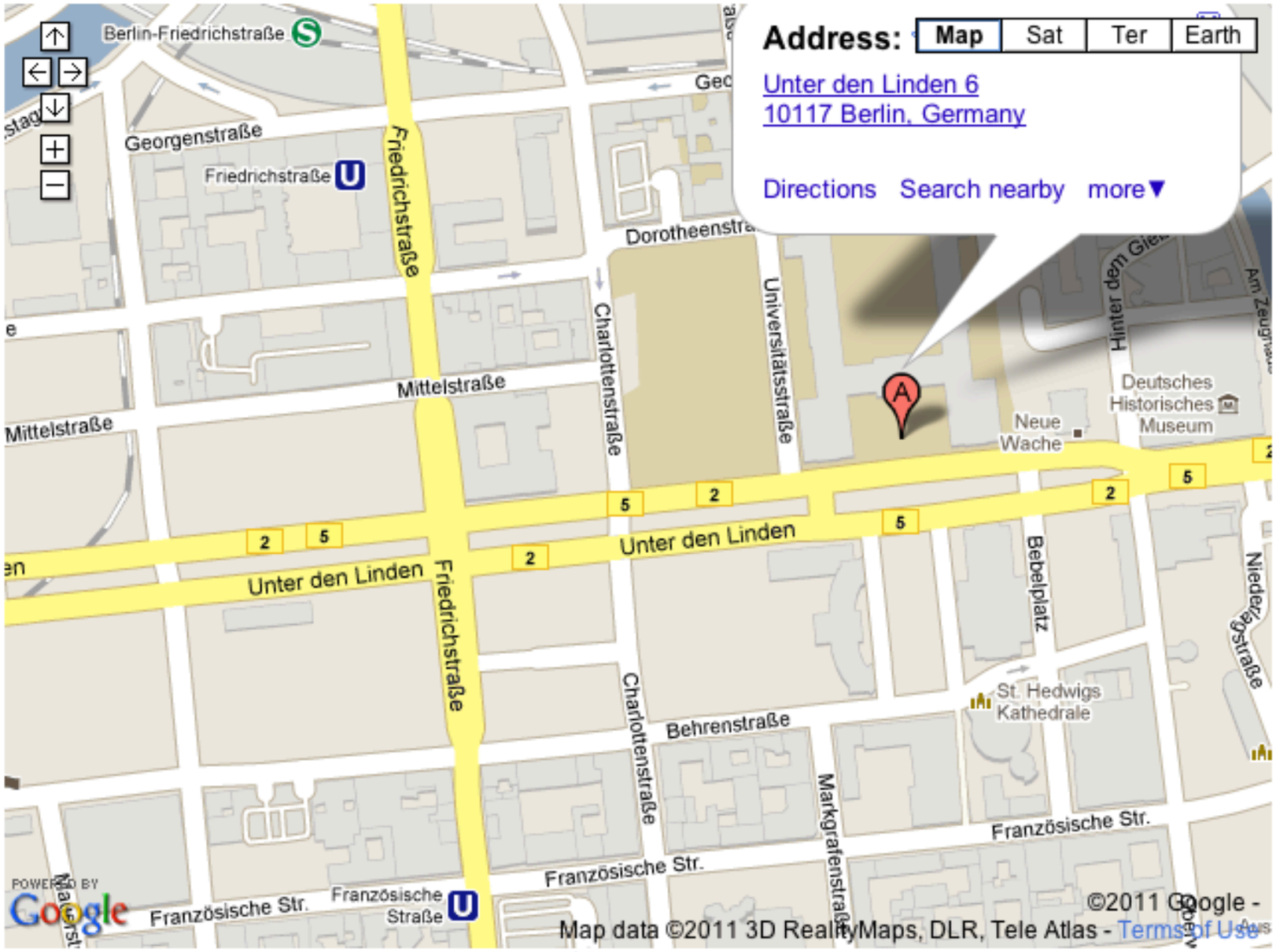
Under the auspices of:

THE ACADEMY OF EUROPE

IEEE computer society

sertech.

Carl Adam Petri Memorial Symposium - Location



bigger map

Carl Adam Petri Memorial Symposium - Scientific prog

9.15	Opening Session
9.30	Personal Reminiscences (Hartmann Genrich, Bernd Krämer, Gerard Memmi, Anastasia Pagnoni, Ulrich Trottenberg, and others)
10.20	Wilhelm Füßl, Deutsches Museum München: Collecting scientific papers. The Archives of the Deutsches Museum and Carl Adam Petri
10.30	Kurt Lautenbach, Koblenz: Belief Revision and Petri Nets
11.00	Coffee
11.30	P.S. Thiagarajan, Singapore: Probabilistic analysis of bio-pathways dynamics
12.00	Rüdiger Valk, Hamburg: Petri's nets and the physical basis of information flow
12.30	Wolfgang Reisig, Berlin: Concurrency based properties
13.00	Buffet
14.00	Kurt Jensen, Aarhus: High-level Petri Nets
14.30	Manuel Silva, Zaragoza: Nets: On synchronic relations and on fluid models
15.00	Wojciech Penczek, Warszawa: Bounded parametric model checking for Petri nets
15.30	Coffee
16.00	Jörg Desel, Hagen: Partial Orders Fit For Work
16.30	Wil van der Aalst, Eindhoven: Discovering Petri Nets: Evidence-Based Business Process Management
17.00	Karsten Wolf, Rostock: Petri Net State Spaces
17.30	End of Symposium
19.15	Dinner

Carl Adam Petri Memorial Symposium - Registration

Please register before January 20, 2011. Later registrations can not be guaranteed, and cost an extra of 20 Euro.

You may register for

- **The Symposium,** including registration, hand outs, participation in the scientific talks, the lunch buffet and the evening dinner at the Humboldt University Faculty Club Cum Laude at the price of **80 Euro**.
- **The Talks only,** including registration, hand outs, participation in the scientific talks, and the lunch buffet at the price of **30 Euro**.
- **The Dinner only,** including the evening dinner at the Humboldt University Faculty Club Cum Laude at the price of **50 Euro**.

Registration and Payment

Sondertarif für Studierende: 15 €

Carl Adam Petri Memorial Symposium - Registration

Step1:Event Selection

Event

(•) Symposium	80€
Symposium includes all (talks, lunch and dinner)	
( ) Talks only (incl, lunch but no dinner)	30€
( ) Student (Talks only, no lunch, no dinner)	15€

Extra

0 ▲▼ dinner ticket(s) (each 50€)	0€

Proceed to Step 2: Personal Information

For