

Studienarbeit
**Modellierung des Workflows der Task
Force Erdbeben des GFZ mit
Petri netzen**

Konstanze Swist
15. Oktober 2008
swist@informatik.hu-berlin.de

Betreuer: Dirk Fahland

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die Studienarbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen angefertigt habe.

Berlin, 15. Oktober 2008

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Aufbau der Arbeit	2
2	Grundlagen	4
2.1	Workflows und Workflowmanagement	4
2.2	Prozessmodell	4
2.2.1	Syntaxelemente	5
2.2.2	Erläuterung des Prozessmodells	9
2.3	Petrinetze	10
3	Abgleich der Modelle	11
3.1	Änderungen	11
3.2	Sonderfälle	12
4	Übersetzung in Petrinetze	13
4.1	Aufgabe	13
4.2	PNML und Yasper	14
4.3	Petrinetzbausteine	14
4.3.1	Besondere Notationselemente in Yasper	14
4.3.2	Petrinetzmuster für die Workflowdarstellung	15
4.3.3	Beispiel für die Zusammensetzung der Bausteine	25
5	Schluss	28
5.1	Diskussion	28
5.2	Literaturvergleich	28
5.3	Zusammenfassung	29
A	Anhang	30
A.1	Änderungen am Prozessmodell	30
A.2	Prozessmodell der Task Force	33
A.2.1	Gesamtsicht und Hauptprozesse	33
A.2.2	Subprozesse	36
A.3	Zusammensetzung der Petrinetzbausteine	47

1 Einleitung

1.1 Motivation

Die Task Force Erdbeben am Geoforschungszentrum Potsdam (GFZ) ist eine Forschungseinrichtung, die die geophysikalischen Vorgänge vor, während und nach Erdbeben untersucht. Dies geschieht sowohl durch Austausch von Informationen mit anderen Forschungseinrichtungen, als auch durch Einsätze vor Ort. Weiterhin werden durch Erdbeben entstandene Schäden analysiert und ihre Ursachen erforscht, um nach Möglichkeit zukünftig solche Schäden zu vermeiden.

Die Forschung in diesem Bereich dient vor allem dazu, die Möglichkeiten der Erdbebenvorhersage zu verbessern. (vgl. [1])

Die Arbeit der Task Force erfordert eine große Menge an Daten und Informationen. Diese werden unter den Mitarbeitern der Task Force bisher ausschließlich durch Gespräche und Papierdokumente ausgetauscht. Ein Workflowmanagement-System kann hier Zeit und Ressourcen einsparen, indem es u.a. den elektronischen Nachrichtenaustausch unterstützt, sowie benötigte Informationen und IT-Werkzeuge automatisch zur Verfügung stellt.

Im Rahmen dieser Studienarbeit nehmen wir an, wir wollen die Arbeit der Task Force mit einem Workflowmanagement-System unterstützen. Für die Planung solcher Unterstützung sind genaue Kenntnisse der Arbeitsabläufe und Datenflüsse notwendig. Bislang sind jedoch die Abläufe in der Task Force nicht schriftlich festgehalten worden. Im Hinblick auf die Verwendung rechnergestützter Prozesse ist es sinnvoll, nicht einfach einen Text zu schreiben, sondern ein Modell der Vorgänge zu erstellen. Ein solches Modell macht es möglich, verschiedene Abstraktionsstufen einzubauen, Unstimmigkeiten und Widersprüche zu lokalisieren, ist aber trotzdem noch leicht zu lesen.

Zu Beginn dieser Studienarbeit existierte bereits ein erstes Modell, welches den Soll-Zustand darstellte, also den standardmässigen Ablauf der Arbeitsschritte. Dieses Modell wurde anhand von Informationen erstellt, die in Gesprächen mit den Mitarbeitern der Task Force gesammelt wurden (siehe [2]). Im Folgenden nennen wir dieses Modell *Prozessmodell*.

Das Prozessmodell ist als Diagramm in Microsoft Visio notiert und besteht im Prinzip derzeit nur aus konkreter Syntax in der (selbsterklärende) Arbeitsschritte über Pfeile und andere graphische Notationselemente in eine bestimmte Ordnung gebracht werden. Obwohl diese Notation ein gewisses intuitives Verständnis für die modellierten Abläufe nahelegt, bestehen noch Freiheitsgrade in der Interpretation. Das Modell ist damit automatisierten Analysemethoden z.B. zur Fehlerfreiheit oder Kosten- und Lastenanalyse nicht zugänglich.

Problematisch am Inhalt des Prozessmodells gestaltet sich weiterhin die Tatsache, dass einerseits die Abläufe aus verschiedensten Gründen variieren und andererseits die tatsächliche Durchführung oft vom gewünschten Ablauf abweichen kann. Um eine weitgehend vollständige und korrekte Darstellung der Arbeitsabläufe zu erhalten, müssen wir also zunächst herausfinden, wann die Arbeitsschritte vom normalen Ablauf abweichen und welche Gründe diese Abweichungen haben.

Zu diesem Zweck wurde ein zweites Modell erstellt, welches die Abläufe so darstellt, wie

sie bei der Einsatzvorbereitung nach dem Erdbeben am 12.09.2007 in Sumatra stattgefunden haben. Wir nennen es *Ablaufmodell*. Einige Mitarbeiter der Task Force haben dafür ihre Vorgehensweise protokolliert, um die nötigen Informationen bereitzustellen (siehe [3]). Mithilfe des Ablaufmodells ist es uns nun möglich, einen tatsächlichen Ablauf nachzuvollziehen und mit der Darstellung im Prozessmodell zu vergleichen. Das ursprüngliche Prozessmodell bietet einen Überblick über die Arbeitsschritte, wobei viele Einzelprozesse nur erwähnt, aber nicht näher beschrieben werden. Das später entstandene Ablaufmodell ist wesentlich detailgenauer als das ursprüngliche Prozessmodell und weicht außerdem besonders in der Reihenfolge der einzelnen Arbeitsschritte vom Prozessmodell ab, da viele Vorgänge davon abhängen, wann Informationen von anderen Einrichtungen zur Verfügung stehen oder wann ein Mitarbeiter abkömmlich ist, um bestimmte Aufgaben zu übernehmen (siehe [2]).

Die erste Teilaufgabe dieser Studienarbeit besteht darin, das Prozessmodell insofern zu ändern, dass es konform zum Ablaufmodell wird, eventuelle Fehler und Unstimmigkeiten zu lokalisieren und diese zu beseitigen. Nach Abschluss dieser Teilaufgabe soll das Prozessmodell so beschaffen sein, dass die im Ablaufmodell beschriebenen Arbeitsschritte im Prozessmodell nachvollziehbar sind.

Mit dem überarbeiteten Prozessmodell lassen sich die Arbeitsabläufe nachvollziehen und analysieren. Dafür ist jedoch die Kenntnis der Prozessmodellsyntax und der gemeinten, jedoch bisher nicht spezifizierten Semantik notwendig. Wollen wir ein Workflowmanagement-System einrichten, müssen wir die Bedeutung der Syntaxelemente des Prozessmodells festlegen. Eine formale Semantik für die Prozessmodellsyntax bietet außerdem die Möglichkeit, die Abläufe verschiedenen automatisierten Analysen zu unterziehen und so möglicherweise noch vor der Einrichtung die Arbeit der Task Force zu optimieren.

Im Hauptteil der Studienarbeit wollen wir daher das angepasste Prozessmodell in Petrinetze übersetzen. Petrinetze werden oft für die formale Modellierung von Arbeitsabläufen verwendet. Daher gibt es bereits viele Werkzeuge, die die Analyse, Validierung und Verifizierung von Petrinetzen unterstützen. Die Definition der Prozessmodellsyntax zielte bereits auf eine spätere Übersetzung in Petrinetze ab.

Wir überführen die einzelnen syntaktischen Ausdrucksmittel des Prozessmodells in Petrinetzbausteine, die sich dann zu einem kompletten Petrinetz zusammenfügen lassen. Das so entstandene Petrinetz soll bzgl. des gemeinten Verhaltens äquivalent zum Prozessmodell sein.

1.2 Aufbau der Arbeit

Die vorliegende Studienarbeit ist in fünf Kapitel aufgeteilt, von denen dieses das erste ist. Im folgenden Kapitel (Grundlagen) werden wir zunächst die Begriffe Workflow und Workflowmanagement behandeln, danach gehen wir auf das Prozessmodell ein, indem wir die Syntaxelemente und das Prozessmodell selbst erklären. Der dritte Teil des Grundlagen-Kapitels beschäftigt sich mit Petrinetzen.

Im dritten Kapitel (Abgleich der Modelle, ab S. 11) erläutern wir die Änderungen an der ersten Version des Prozessmodells, sowie Sonderfälle, die wir bisher nicht ins Modell aufgenommen haben. Eine tabellarische Übersicht über die Änderungen befindet sich au-

ßerdem im Anhang (S. 30).

Im Anschluss (Kapitel 4) gehen wir auf den zweiten Teil der Aufgabe ein, dokumentieren die Auswahl eines Petrinetzwerkzeuges und legen die entstandenen Petrinetzbausteine, sowie ein Beispiel für ihre Zusammensetzung dar.

Der Schluss (Kapitel 5) enthält die Diskussion der Ergebnisse, eine Auseinandersetzung mit anderen Arbeiten zu ähnlichen Problemen und eine kurze Zusammenfassung dieser Studienarbeit.

2 Grundlagen

In diesem Kapitel erläutern wir die grundlegenden Begriffe, die für das Verständnis dieser Studienarbeit nötig sind. Zunächst geht es dabei um *Workflows* und *Workflowmanagement* (Kap. 2.1). Es folgt das *Prozessmodell* (Kap. 2.2) mit Erklärung der Syntax und Erläuterung des vorliegenden Modells. Abschließend werden *Petrinetze* (Kap. 2.3) behandelt.

2.1 Workflows und Workflowmanagement

Um die Arbeit der Task Force zu optimieren, sollen ihre Arbeitsabläufe zunächst modelliert werden. Später ließe sich mithilfe dieses Modells ein Workflowmanagement-System einrichten, welches die Abarbeitung der modellierten Arbeitsschritte unterstützt.

Definition 2.1.1 *Ein Workflow ist die vollständige oder teilweise Automatisierung eines Arbeitsablaufes, in welchem Dokumente, Informationen oder Aufgaben unter Berücksichtigung einer Menge von Prozedurregeln von einem Teilnehmer zum nächsten zur Bearbeitung weitergegeben werden.*

In Prozessmodellen werden die erwähnten Prozedurregeln grafisch dargestellt, d.h. es werden bspw. die Reihenfolge der Arbeitsschritte, Zuständigkeiten, verarbeitete Daten und erzeugte Ergebnisse in einer graphischen Notation festgelegt.

Definition 2.1.2 *Ein Workflowmanagement-System definiert, erzeugt und steuert die Ausführung von Workflows mithilfe von Software, die auf ein oder mehreren Workflow-Maschinen läuft und die in der Lage ist, die Prozessdefinitionen zu interpretieren, mit den Teilnehmern zu interagieren und, wenn nötig, IT-Werkzeuge und Anwendungen aufzurufen.*

(beide Definitionen übersetzt aus [4])

Ein Workflowmanagement-System ist in der Lage, den Prozessfortschritt zu überwachen, indem es z.B. eingehende Nachrichten und Eingaben prüft. Aufgrund dieses Wissens und anhand des Prozessmodells kann es bspw. benötigte Software starten, Mitarbeiter benachrichtigen oder Dokumente und Informationen automatisch weiterleiten.

2.2 Prozessmodell

Mit dem Begriff *Prozessmodell* bezeichnen wir in dieser Studienarbeit dasjenige Modell, welches die allgemeinen Arbeitsabläufe der Task Force modelliert. (Im Vergleich dazu nennen wir das Modell des Ablaufes nach dem Erdbeben in Sumatra *Ablaufmodell*.) In diesem Kapitel erläutern wir zunächst die einzelnen Syntaxelemente des Prozessmodells und ihre gemeinte intuitive Interpretation. Im Anschluss gehen wir näher auf das eigentliche Modell ein.

Für das Prozessmodell wurde bewusst keine existierende Workflowbeschreibungssprache (wie z.B. BPEL) verwendet, sondern eine eigene Syntax entwickelt. Diese Syntax hat den Vorteil, dass sie auch für die Mitarbeiter der Task Force verständlich ist.

2.2.1 Syntaxelemente

Das Prozessmodell verwendet eine vereinfachte Flussdiagramm Notation (siehe Abb. 2), deren Elemente im Folgenden beschrieben werden. Einen Überblick über die einzelnen Ausdrucksmittel gibt das Klassendiagramm in Abbildung 1.

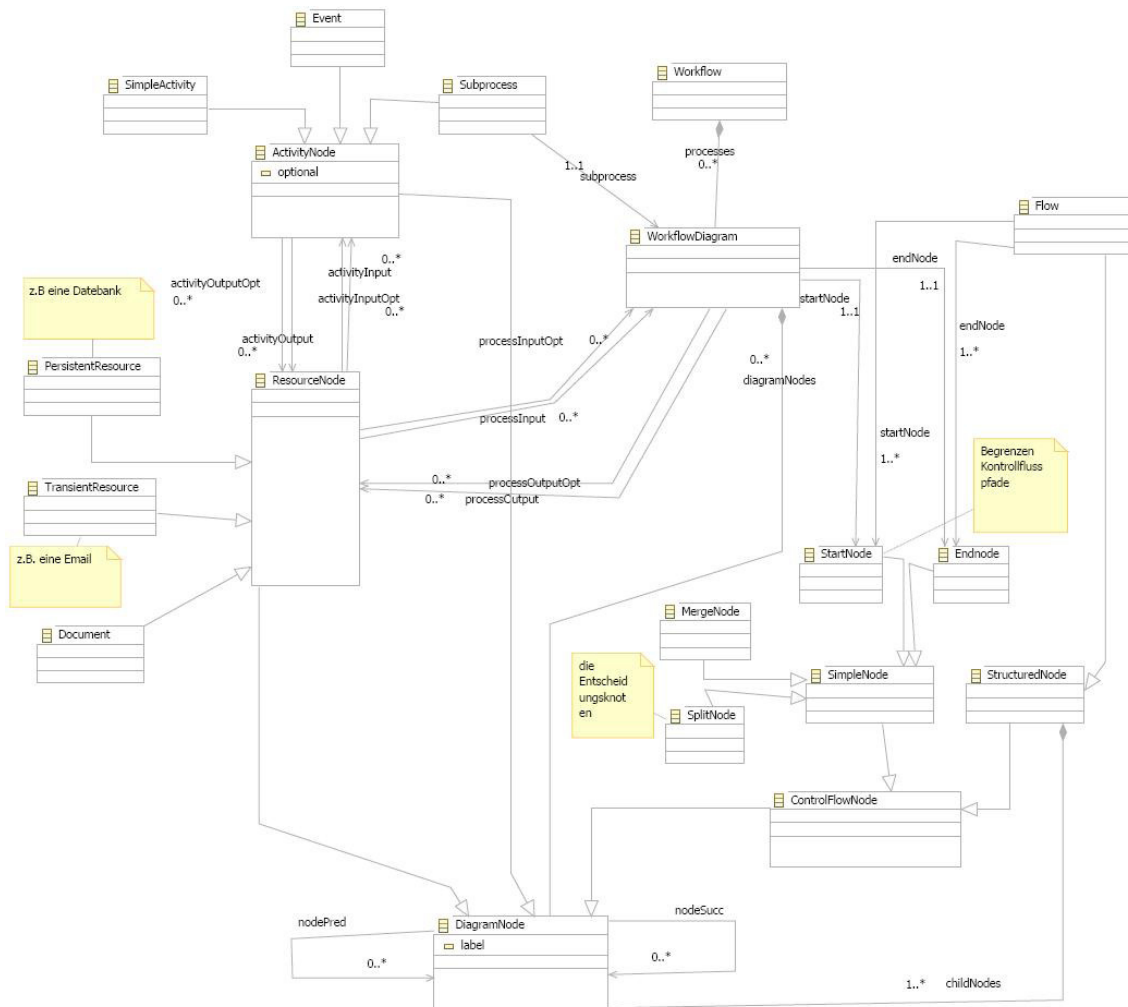


Abbildung 1: Klassendiagramm der Ausdrucksmittel im Prozessmodell

Eine Modellinstanz mit dieser Syntax kann als Graph betrachtet werden. Die Knoten sind dann Aktivitäten, Entscheidungen und Ereignisse, die Kanten Kontroll- und Datenflüsse.

Ein Knoten *schaltet*, indem der entsprechende Arbeitsschritt ausgeführt wird. Ein Knoten *k* kann dann schalten, wenn er *aktiviert* ist. Dies ist im Allgemeinen dann der Fall, wenn alle Knoten, die Kanten zu *k* besitzen, geschaltet haben (d.h. wenn alle notwendigen Eingangsdaten vorhanden und alle zuvor auszuführenden Arbeitsschritte erledigt sind).

Aktivitäten (ActivityNode, siehe Klassendiagramm in Abb. 1)

Aktivitäten sind die eigentlichen Arbeitsschritte, die an bestimmten Stellen des Prozesses ausgeführt werden. Sie können notwendige und optionale Ein- und Ausgangsdatenflüsse besitzen, sind immer Teil eines Kontrollflusses und werden als blaue Rechtecke dargestellt. Ihre Umrandung bestimmt die Art der Aktivität.

Kontrollflüsse

Kontrollflüsse bilden die Verbindungen zwischen allen Syntaxelementen, die keine Daten verkörpern. Sie definieren die Reihenfolge der Arbeitsschritte und werden als blaue Pfeile dargestellt. Jeder Kontrollfluss hat einen Anfangs- und einen Endknoten (StartNode bzw. EndNode im Klassendiagramm in Abb. 1, beides durch flache, blaue Ellipsen dargestellt), die den Kontrollflusspfad begrenzen.

Mehrere Kontrollflusskanten an einem Knoten bezeichnen nebenläufige Strukturen. Hat also ein Knoten mehrere ausgehende Kontrollflusskanten, so werden, nachdem der Knoten geschaltet hat, alle diese Kanten gleichzeitig bzw. in einer nicht bestimmten Reihenfolge verfolgt. Besitzt ein Knoten mehrere eingehende Kontrollflusskanten, so müssen alle dementsprechend vorhergehenden Knoten geschaltet haben, damit der aktuelle Knoten schalten kann.

optionale Kontrollflüsse

Optionale Kontrollflüsse können, müssen aber nicht abgearbeitet werden. Ihre Anfangs- und Endknoten haben gestrichelte Umrandungen.

einfache Aktivitäten (SimpleActivity im Klassendiagramm, Abb. 1)

Einfache Aktivitäten sind die Darstellung von atomaren Arbeitsschritten im Prozessmodell. Sie werden mit einfacher Umrandung dargestellt (siehe Abb. 2 oben).

komplexe Aktivitäten (Subprocess im Klassendiagramm, Abb. 1)

Eine komplexe Aktivität ist der Aufruf eines Subprozesses, der an anderer Stelle im Prozessmodell definiert wird.

Die Darstellung komplexer Aktivitäten unterscheidet sich von der der einfachen Aktivitäten durch zwei zusätzliche senkrechte Linien am linken und rechten Rand des Rechtecks.

Wird eine komplexe Aktivität aktiviert, so ist gleichzeitig der Kontrollfluss ihres Subprozesses aktiviert. Die komplexe Aktivität hat geschaltet, wenn der Subprozess abgearbeitet wurde.

optionale Ausführung

Optionale Aktivitäten sind Arbeitsschritte, deren Ausführung nicht zwingend erforderlich ist. Sie werden mit gestricheltem Rand dargestellt. Wird eine optionale Aktivität nicht ausgeführt, so läuft der Kontrollfluss weiter, als wäre die Aktivität nicht vorhanden.

nebenläufige Ausführung (Flow-Aktivität im Klassendiagramm, Abb. 1)

Werden mehrere Aktivitäten gleichzeitig ausgeführt, so werden diese im Prozessmodell durch einen am oberen und unteren Rand begrenzten Kasten (Flow) zusammengefasst. Eine Flow-Aktivität verhält sich wie eine komplexe Aktivität: haben alle vorhergehenden

Knoten geschaltet, so werden alle im Flow enthaltenen Kontrollflüsse aktiviert. Der Flow hat geschaltet, wenn alle in ihm enthaltenen Kontrollflüsse abgearbeitet wurden.

Auswahl / Entscheidung (SplitNode im Klassendiagramm, Abb. 1)

Die Auswahl spaltet den Kontrollfluss in zwei oder mehrere mögliche Alternativen auf, von denen genau ein Zweig wählbar ist. Sie wird als blaue Raute dargestellt und mit einem Text versehen, der die Entscheidung informal beschreibt.

Das Schalten der Auswahl verfolgt genau einen (nicht-deterministisch) gewählten Zweig und aktiviert den dementsprechend nachfolgenden Knoten.

Zusammenführung (MergeNode im Klassendiagramm, Abb. 1)

Hier werden die verschiedenen Zweige nach einer Entscheidung wieder zu einem Kontrollflussspfad zusammengefasst. Auch die Zusammenführung wird als blaue Raute, allerdings im Allgemeinen ohne Text dargestellt.

Die Zusammenführung ist aktiviert, sobald einer der eingehenden Kontrollflüsse abgearbeitet ist.

Behandlung externer Ereignisse (Event im Klassendiagramm, Abb. 1)

Externe Ereignisse werden im Prozessmodell durch blaue Sechsecke dargestellt. Im vorliegenden Fall sind dies Anfragen nach Berichten oder Karten, die bspw. von der Presse an die Task Force gerichtet werden. Da externe Ereignisse außerhalb des modellierten Workflows erzeugt werden, haben diese Elemente keine eingehenden Kontrollflusskanten und bilden somit ebenfalls den Anfang eines Kontrollflusspfades.

Datenobjekte (ResourceNode im Klassendiagramm, Abb. 1)

Im vorliegenden Modell gibt es drei verschiedene Arten von Daten:

- *Persistente Daten* (PersistentResource im Klassendiagramm) werden mit abgerundeten Seiten dargestellt. Bei einem Zugriff werden Daten gelesen oder aktualisiert.
- *Transiente Daten* (TransientResource im Klassendiagramm) werden als Rhomboide dargestellt. Ein Zugriff kann Daten produzieren oder konsumieren.
- *Dokumente*, dargestellt als Rechtecke, deren unteren Rand eine Schlangenlinie begrenzt, können sowohl persistente als auch transiente Daten verkörpern. Die Ausprägung der Datenspeicherung ist fallabhängig und im Modell nicht ausgedrückt.

Alle Darstellungen von Daten haben einen orangefarbenen Hintergrund.

Datenflüsse

Pfeile von oder zu Datenobjekten kennzeichnen Datenflusskanten (orange gefärbt). Hier werden Daten erzeugt, benötigt oder weitergeleitet. Die Pfeilrichtung bestimmt, ob Ein- oder Ausgabedaten gemeint sind. An der Position, an der die Kante das Prozesselement trifft, läßt sich ablesen, ob die Daten obligatorisch oder optional sind. Obligatorische Daten werden mit der oberen (Eingabedaten) oder unteren (Ausgabedaten) Kante des Elementes verbunden, optionale Datenflüsse treffen eine Seite des Elementes.

Workflow der Taskforce „Erdbeben“
 Dirk Fahland, Timo Mika Gläßer, Falko Theisseimann, Heiko Woith

Legende

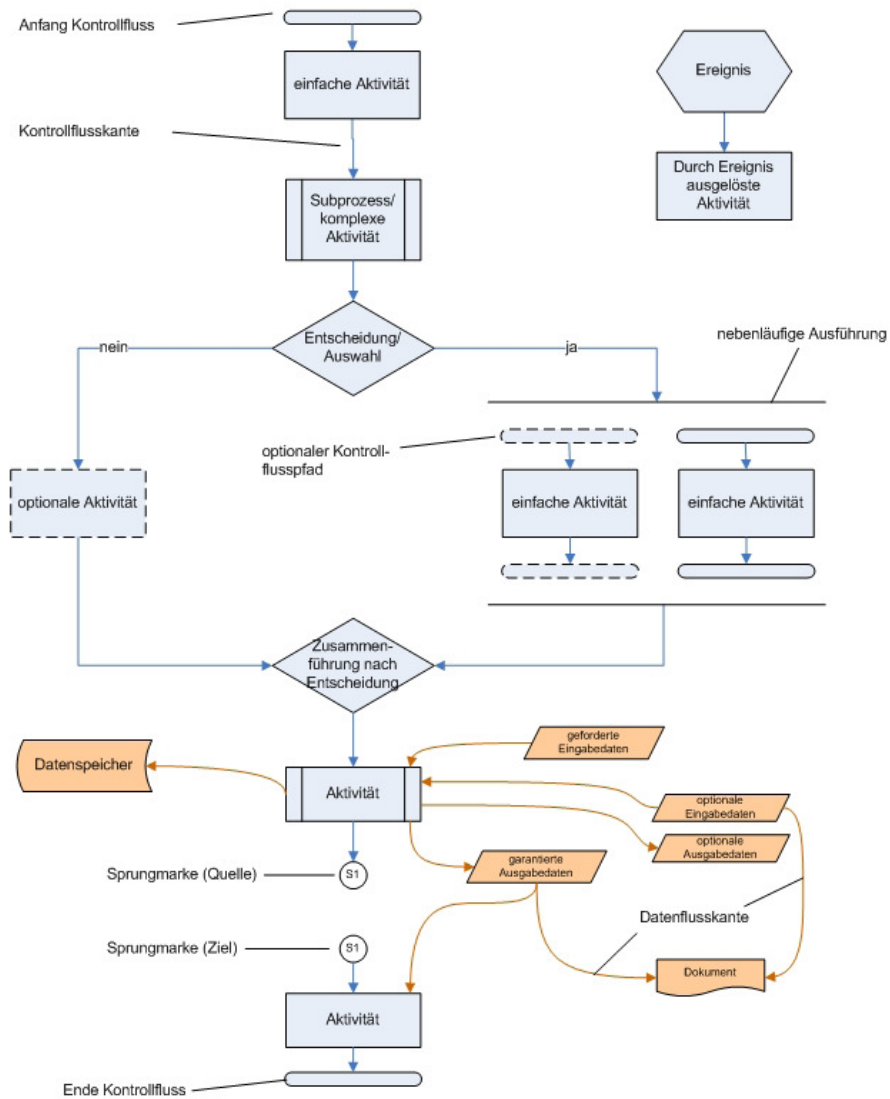


Abbildung 2: Legende der Prozessmodellsyntax (Quelle: [5])

2.2.2 Erläuterung des Prozessmodells

Sehen wir uns im Folgenden das Prozessmodell nach den durchgeführten Änderungen an:

Die graphische Darstellung des Prozessmodells (siehe Anhang A.2) wurde mit Microsoft Visio (<http://office.microsoft.com/de-de/visio/FX100487861031.aspx>) erstellt und ist in sog. *Zeichenblätter* unterteilt, wobei jedes Zeichenblatt einen Subprozess darstellt.

Das Prozessmodell besteht im Wesentlichen aus drei Detaillierungsstufen. Es gibt ein Zeichenblatt, auf dem der gesamte Ablauf in der Übersicht abgebildet ist. Nennen wir dieses Zeichenblatt *Gesamtsicht*. In der Gesamtsicht sind die *Hauptprozesse* dargestellt, welche wiederum in *Teilprozessen* verfeinert werden.

Alle Prozessdarstellungen enthalten eine (z.T. implizite) Zeitachse, nach der die zeitliche Ordnung der dargestellten Aktivitäten ersichtlich ist.

1. *Gesamtsicht* (siehe Anhang A.2.1, S. 33): Die Gesamtsicht stellt alle im Zusammenhang mit einem Erdbeben ablaufenden Prozesse im Überblick dar. Prozessbeginn ist das Stattfinden bzw. die Meldung eines Erdbebens. In der Folge werden verschiedene Hauptprozesse in Gang gesetzt: der Prozess „Informationen sammeln“, gefolgt von der „wissenschaftlichen Entscheidung“ sowie der „Auswertung“ der gesammelten Daten. Weiterhin können Anfragen nach Berichten oder Karten jederzeit eingehen. Die Auswertung von älteren Beben kann (muss aber nicht) neben dem neuen Einsatz weitergeführt werden. Auf die „wissenschaftliche Entscheidung“ folgen „Einsatzvorbereitung“, „politische Entscheidung“ und der „Einsatz“ selbst.
2. *Hauptprozesse* (siehe Anhang A.2.1, ab S. 34): Als Hauptprozesse bezeichnen wir die Prozesse „Auswertung“, „Informationen sammeln“, „wissenschaftliche Entscheidung“ und „Einsatzvorbereitung“. Der „Einsatz“ selbst wird im gegenwärtigen Modell nicht näher modelliert, da er bisher nur unzureichend beschrieben werden konnte. Einige Hauptprozesse enthalten wiederum komplexe Aktivitäten. Die von ihnen aufgerufenen Subprozesse bezeichnen wir als Teilprozesse.
3. *Teilprozesse* (siehe Anhang A.2.2, ab S. 36): Die Mehrzahl der Teilprozesse findet sich im Hauptprozess „Informationen sammeln“. Außerdem werden die Aktivitäten „Logistik“ und „Kontakte“ aus dem Hauptprozess „Einsatzvorbereitung“ und das Erstellen von Karten präzisiert.

Die vorliegende Darstellung beschreibt eine Menge von möglichen Abläufen des Workflows, erhebt jedoch keinen Anspruch auf Vollständigkeit. Es sind bereits einige Szenarien bekannt, welche bisher noch nicht in das Prozessmodell aufgenommen wurden. Auf diese Sonderfälle werden wir später noch eingehen (Kapitel 3.2).

Beispielhaft erklären wir nun den Hauptprozesses „wissenschaftliche Entscheidung“ (siehe Anhang A.2.1, S. 35): Hier wird durch die wissenschaftlichen Mitarbeiter der Task Force entschieden, ob Interesse an einem Einsatz vor Ort besteht.

Wenn ein Erdbeben auftritt, werden zunächst diverse Informationen gesammelt. Diese Informationen sind die Grundlage der wissenschaftlichen Entscheidung und werden daher als Vorbedingung im Zeichenblatt aufgeführt.

Die Entscheidung hängt in erster Linie von einer Menge von K.O.-Kriterien ab. Trifft eines oder mehrere zu (bspw. wenn das Beben unter Wasser auftrat), so ist ein Einsatz am Ort des Geschehens von vornherein ausgeschlossen und der Kontrollfluss endet bereits an dieser Stelle. Falls keine K.O.-Kriterien zutreffen, wird zunächst aus den gesammelten Bebeninformationen ein Bericht erstellt. Dieser bildet die Grundlage einer Diskussionsrunde, in welcher jeder Mitarbeiter der Task Force entscheidet, ob er Interesse hat, an einem Einsatz vor Ort teilzunehmen oder nicht. Das Ergebnis der Diskussion ist die Entscheidung für oder gegen den Einsatz vor Ort.

2.3 Petrinetze

Für die formale Modellierung von Workflows werden häufig Petrinetze verwendet, da sie eine eindeutige Semantik besitzen und somit die Verifikation, Validierung und Analyse der Workflows ermöglichen.

Definition 2.3.1 Ein *Inhibitor-Petrinetz* ist ein 4-Tupel $N = (P, T, F, I)$ mit

- einer nichtleeren Menge P von Plätzen;
- einer nichtleeren Menge T von Transitionen mit $P \cap T = \emptyset$;
- einer Flussrelation $F \subseteq ((P \times T) \cup (T \times P))$ (die Menge der Bögen) und
- einer Inhibitorrelation $I \subseteq (P \times T) \cap F$ (die Menge der Inhibitorbögen).

In der grafischen Repräsentation werden Plätze durch Kreise, Transitionen durch Quadrate und die Flussrelation durch Pfeile symbolisiert. Inhibitorbögen werden als Linien mit einem ausgefüllten Kreis an ihrem Ende dargestellt. Ein Inhibitorbogen verläuft immer von einem Platz zu einer Transition.

(vgl. [6])

Eine *Markierung* beschreibt einen Zustand des Petrinetzes. Sie weist jedem Platz eine nicht-negative Anzahl von *Marken* zu. Entsprechend der Flussrelation bezeichnet man alle Plätze mit ausgehenden Bögen zu einer Transition t als *Vorplätze von t* und alle Plätze mit eingehenden Bögen von t aus als *Nachplätze von t* . Eine Transition t ist *aktiviert*, wenn alle Vorplätze von t markiert sind. Eine aktivierte Transition kann *schalten*. Schaltet eine Transition, so wird aus jedem ihrer Vorplätze je eine Marke entfernt und in jedem ihrer Nachplätze eine Marke hinzugefügt. Verläuft ein Inhibitorbogen von einem Platz p zu einer Transition t , so verhindert eine Marke auf p die Aktivierung von t .

Auf die Vorteile der Modellierung mit Petrinetzen wird in Kapitel 4.1 eingegangen.

3 Abgleich der Modelle

Das Prozessmodell enthält noch einige Unklarheiten und Fehler. Um diese lokalisieren und beheben zu können, wurden im Rahmen des tatsächlichen Einsatzes der Task Force nach dem Erdbeben 2007 in Sumatra die Arbeitsschritte protokolliert und im *Ablaufmodell* festgehalten. Dieses bildet nun die Grundlage für die Überarbeitung des Prozessmodells (siehe Kapitel 1.1). Als Schwierigkeit hierbei erweisen sich vor allem die unterschiedlichen Detaillierungsgrade der Modelle. Für das Ablaufmodell haben Mitarbeiter der Task Force jeden ihrer Arbeitsschritte protokolliert. Dies führte zu einer sehr hohen Detailtiefe. Im Prozessmodell dagegen, das auf Grundlage von Gesprächen mit Mitarbeitern der Task Force entstand, sind allgemeine Abläufe festgehalten worden, die viel weniger Details enthalten (vgl. [2] und [3]). Beim Angleichen des Prozessmodells müssen ein sinnvoller Detaillierungsgrad festgelegt und zu tiefe Details weggelassen werden, sodass das veränderte Modell alle im Ablaufmodell festgehaltenen Vorgänge darstellt, aber nicht zu unübersichtlich wird.

Im Ergebnis der Überarbeitung soll der modellierte Ablauf im Prozessmodell nachvollziehbar sein.

In den folgenden Abschnitten erläutern wir zunächst die Änderungen am Prozessmodell. Im Anschluss gehen wir auf einige Sonderfälle ein, die bisher nicht berücksichtigt sind.

3.1 Änderungen

Um Unstimmigkeiten zunächst zu lokalisieren, haben wir versucht, die Arbeitsschritte aus dem Ablaufmodell im Prozessmodell nachzuvollziehen. In Zusammenarbeit mit der Task Force haben wir anschließend die festgestellten Unstimmigkeiten klassifiziert und die nötigen Änderungen besprochen. In diesem Gespräch haben wir außerdem noch einige grundsätzliche Fehler im Prozessmodell festgestellt und dafür ebenfalls entsprechende Lösungen besprochen. Im Wesentlichen traten drei Sorten von Unstimmigkeiten im Prozessmodell auf:

1. *Inkonsistenzen*: Arbeitsschritte, die im Prozessmodell anders modelliert sind als im Ablaufmodell,
2. *Unvollständigkeiten*: Arbeitsschritte, die im Prozessmodell fehlen und
3. *fehlerhafte Reihenfolgen*.

Im Anschluss führten wir die in der Anlage (A.1) beschriebenen Änderungen durch.

Ein Beispiel:

Im ursprünglichen Prozessmodell war die wissenschaftliche Entscheidung über einen Vor-Ort-Einsatz zwingend das Ergebnis einer Diskussion. Im Fall des Sumatra-Bebens fand aber keine solche Diskussion statt. Das Gespräch mit einem Mitarbeiter der Task Force ergab, dass hier ein Vor-Ort-Einsatz von vornherein ausgeschlossen war, weil das Beben im Wasser stattfand und für einen solchen Fall keine Messgeräte vorhanden sind. Es gibt mehrere solche Kriterien, anhand derer noch vor einer eventuellen Diskussion geprüft

wird, ob ein Vor-Ort-Einsatz überhaupt sinnvoll ist. Das Modell wurde daraufhin dahingehend geändert, dass nun vor der wissenschaftlichen Diskussion eine Prüfung auf diese K.O.-Kriterien stattfindet, die gegebenenfalls die Diskussion und den Vor-Ort-Einsatz verhindert.

Das im Ergebnis der im Anhang A.1 aufgeführten Änderungen entstandene Prozessmodell findet sich im Anhang A.2. Im nächsten Abschnitt beschreiben wir einige Fälle, die bei der Prüfung aufgefallen sind, jedoch in dieser Version des Prozessmodells keine Berücksichtigung gefunden haben.

3.2 Sonderfälle

Während der Anpassung des Modells traten einige Sonderfälle in den Abläufen zutage, die wir in der vorliegenden Version nicht modelliert haben. Diese Fälle treten sehr selten ein, sind aber so schwierig zu modellieren, dass ihre Berücksichtigung das Modell unleserlich machen würde. Wir wollen sie jedoch im Folgenden für zukünftige Versionen festhalten.

1. Die kausale Abhängigkeit der Aktivität „Bebenparameter, Bruchfläche“ von „Epizentren div. Agencies“ im Hauptprozess „Informationen sammeln“ kann in Einzelfällen aufgelöst werden und es genügt, nur eine der Aktivitäten auszuführen. Sowohl die Bebenparameter als auch die Bruchfläche lassen sich ohne Wissen über die Epizentren feststellen, da alle dafür benötigten Daten aus dem Internet extrahiert werden. In aller Regel werden jedoch beide Aktivitäten ausgeführt. Daher wurde dieser Sonderfall nicht ins Modell aufgenommen.
2. Während der Entscheidungsfindung kann es ebenfalls zu einem Sonderfall kommen. Normalerweise bricht ein „nein“ zum Vor-Ort-Einsatz bei der Prüfung auf K.O.-Kriterien oder der wissenschaftlichen Diskussion den Entscheidungsprozess ab. In diesen Fällen wäre dann eine politische Entscheidungsfindung überflüssig, da ein Vor-Ort-Einsatz auf keinen Fall stattfindet. Bei bestimmten Beben ist jedoch aus verschiedenen und oft nicht vorhersagbaren Gründen das Medieninteresse so groß, dass auch wenn kein Interesse seitens der Wissenschaftler besteht, eine politische Entscheidung *für* einen Vor-Ort-Einsatz getroffen wird und der Einsatz damit entgegen der wissenschaftlichen Entscheidung statt findet. Die Modellierung dieses Falles ist mit den gegebenen Ausdrucksmitteln zwar prinzipiell möglich, würde aber die Logik des Einsatzes und der Entscheidungsfindung erheblich komplizierter gestalten. Daher findet sich dieser Fall bisher ebenfalls nicht im Modell wieder.
3. Zukünftig sollen bei besonders interessanten Beben auch ohne Anfrage Presseberichte erstellt werden. Dies wird zur Zeit noch nicht so gehandhabt, daher wurde dieser Vorgang bisher nicht modelliert, soll aber ins Modell aufgenommen werden, wenn die Praxis dahingehend geändert wird.

Das aktuelle Modell, welches die Grundlage für die Übersetzung in Petrinetz Semantik bildete, findet sich im Anhang (A.2).

4 Übersetzung in Petrinetze

Das Prozessmodell stellt nun die Abläufe so detailliert dar, wie es mit einer informalen Syntax möglich ist. Um die Semantik formal eindeutig abzubilden reicht das Prozessmodell jedoch nicht aus. Hierfür fiel die Wahl auf Petrinetze, die oftmals für die formale Modellierung von Workflows eingesetzt werden. Die Entwicklung der Prozessmodell-syntax zielte bereits auf eine Übersetzung in Petrinetze ab.

In diesem Kapitel beschäftigen wir uns mit der Übertragung des Prozessmodells in Petrinetze. Im ersten Abschnitt erläutern wir zunächst noch einmal die Aufgabe, vor der wir nun stehen. Anschließend betrachten wir die Petrinetzbeschreibungssprache PNML und das Petrinetztool Jasper. Schließlich wenden wir uns den entstandenen Petrinetzbausteinen zu.

4.1 Aufgabe

Das Prozessmodell ist jetzt zum Ablaufmodell des Sumatra-Einsatzes konsistent. Das Problem dieses Modells ist jedoch, dass es zwar auch für Informatikfremde relativ leicht verständlich ist, aber keine formal eindeutige Semantik besitzt. Daher überführen wir das Modell in Petrinetze. Petrinetze bieten den Vorteil, eine formal eindeutige Semantik zu besitzen und sich durch Werkzeuge analysieren, verifizieren und validieren zu lassen. Petrinetztools wie Jasper ([7]) bieten die Möglichkeit, die modellierten Abläufe zu simulieren und so Fehler wie z.B. unerreichbare Zustände zu identifizieren. Weiterhin existieren Tools zur Strukturanalyse, z.B. INA ¹, und Modelchecker wie z.B. LoLA ². Eine gute Übersicht über existierende Petrinetztools bietet Petri Nets World ³.

Die Vorteile von Petrinetzen bilden jedoch gleichzeitig auch eine Schwierigkeit bei der Übersetzung: Die Ausdrucksmittel des Prozessmodells befinden sich auf einem hohen Abstraktionsgrad und fassen komplexe Vorgänge in einzelnen Ausdrucksmitteln zusammen. Wir müssen uns in den Petrinetzen für eine Interpretation der Ausdrucksmittel der Prozessmodellsyntax entscheiden und damit die Semantik festlegen. Diese Interpretation ist also nur eine von mehreren Möglichkeiten und als Vorschlag zu betrachten.

Weiterhin ist das Prozessmodell eventuell noch nicht absolut korrekt. Es ist möglich, dass das Prozessmodell zukünftig noch geändert wird. Wir müssen bei der Übersetzung darauf achten, dass sich Änderungen im Prozessmodell möglichst leicht in das Petrinetz übertragen lassen. Würden wir das Prozessmodell in einem Stück übertragen, so würde jede Änderung im Prozessmodell eine manuelle Änderung und Prüfung des gesamten Petrinetzes erfordern, was die Gefahr von neu eingebauten Inkonsistenzen erhöhen würde.

Aus diesem Grund modellieren wir die einzelnen Syntaxelemente des Prozessmodells als *Petrinetzbausteine* (kurz: *Bausteine*), die sich später automatisiert zu einem Gesamtnetz zusammensetzen lassen. Die Herausforderung dabei ist, die abstrakte und umfangreiche Syntax des Prozessmodells in die einfache Petrinetzsyntax zu übertragen und die Schnittstellen der einzelnen Bausteine so zu modellieren, dass die Bausteine sich zu einem Petrinetz komponieren lassen.

¹<http://www2.informatik.hu-berlin.de/starke/ina.html>

²<http://www2.informatik.hu-berlin.de/kschmidt/lola.html>

³<http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html>

4.2 PNML und Jasper

Um die Zusammensetzung der Bausteine zu einem kompletten Petrinetz später automatisieren zu können, wollen wir die Netzbausteine in einer maschinenlesbaren Beschreibungssprache festhalten. Eine solche Sprache ist die *Petri Net Markup Language (PNML)*.

PNML ist eine auf XML basierende Beschreibungssprache für Petrinetze. Ein PNML-Dokument setzt sich im Wesentlichen aus einem oder mehreren Netzen zusammen. Hinzu kommen eventuell einige toolspezifische Informationen, die jedoch besonders gekennzeichnet und somit leicht von verschiedenen Werkzeugen ignorierbar sind. Ein jedes Netz enthält Plätze, Transitionen und Bögen, sowie Informationen über ihre graphische Darstellung (vgl. [8] und [9]).

Ein Vorteil einer XML-basierten Beschreibungssprache liegt in der weiten Verbreitung von XML: es existieren bereits viele Werkzeuge zum Parsen von XML-konformen Dokumenten. Dadurch wird es leichter, ein Werkzeug zu entwickeln, welches die in dieser Studienarbeit entstandenen Petrinetzbausteine zu einem Gesamtnetz zusammensetzt.

Ein weiterer Vorteil von PNML ist, dass es sich hierbei um ein Austauschformat handelt. Ein mit PNML beschriebenes Petrinetz kann leicht auch von anderen Werkzeugen als dem, mit dem es erstellt wurde, gelesen und bearbeitet werden.

Aufgrund dieser Vorteile fiel unsere Entscheidung auf PNML als Beschreibungssprache.

Obwohl wegen der allgemeinen Struktur von XML-konformen Dokumenten prinzipiell möglich, ist es nicht ganz trivial, ein PNML-Dokument komplett von Hand zu erstellen. Es existiert eine Vielzahl von grafischen Werkzeugen zum Erstellen und Bearbeiten von PNML beschriebenen Petrinetzen. Die Wahl fiel hier auf *Jasper* (Homepage: [7]), welches bereits am Lehrstuhl eingesetzt wird und den Ansprüchen dieser Aufgabe genügt. Jasper bietet die Möglichkeit, Petrinetze grafisch (per drag-and-drop) zusammenzusetzen und sie automatisch oder manuell zu testen. Es ist einfach in der Handhabung und bietet zudem einige Notationselemente, die die Modellierung vereinfachen und die Darstellung der entstehenden Petrinetze übersichtlicher machen (siehe Abschnitt 4.3.1).

4.3 Petrinetzbausteine

In diesem Abschnitt beschäftigen wir uns mit den Petrinetzbausteinen, die das Ergebnis dieser Studienarbeit sind. Für ihre Erzeugung haben wir besondere Notationselemente verwendet, die Jasper zur Verfügung stellt. Diese erläutern wir im ersten Abschnitt. Anschließend wenden wir uns den einzelnen Bausteinen zu.

4.3.1 Besondere Notationselemente in Jasper

Wie oben bereits erwähnt, bietet Jasper einige Notationselemente, die oft verwendete Petrinetzmuster zusammenfassen. In dieser Studienarbeit wurden insbesondere *XOR-Transitionen* verwendet. Dieses Jasper-Element vereinfacht die Darstellung von Entscheidungen in Petrinetzen, bei denen genau einer der Zweige nach der Entscheidung aktiviert wird. Die Übersetzung einer XOR-Transition in einfache Petrinetz Syntax findet sich in Abb. 3. Wenn eine XOR-Transition mehrere eingehende oder ausgehende Bögen besitzt,

wird immer genau einer davon gewählt. Das bedeutet, dass sie sowohl für Entscheidungen verwendet werden kann, als auch für die Zusammenführung nach einer Entscheidung. Eine XOR-Transition kann jedoch nicht mehrere Bögen gleichzeitig bedienen.

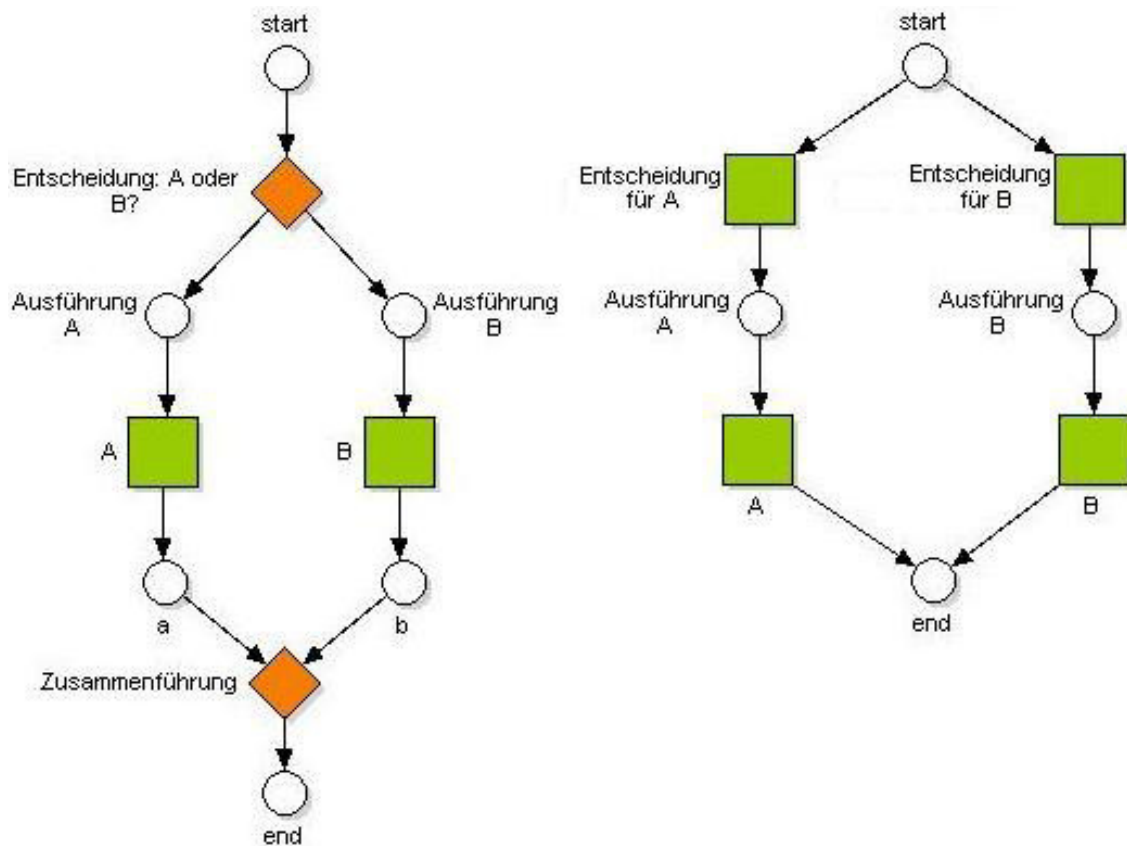


Abbildung 3: Übersetzung des XOR-Elementes (links) in ein einfaches Petrinetz (rechts)

Weiterhin gibt es in Jasper die Möglichkeit, Subnetze auf eigenen Seiten darzustellen, was die Übersichtlichkeit bei großen Netzen erhöht. Dafür wird im übergeordneten Petrinetz eine Transition als *subnet-Transition* definiert. Alle Vor- und Nachplätze dieser Transition werden als Referenzknoten in das Subnetz kopiert (siehe Abb. 4), welches wiederum beliebig komplex sein kann. Der Verwendung der subnet-Transition entspricht das Einsetzen des zugehörigen Subnetzes an ihrer Stelle.

4.3.2 Petrinetzmuster für die Workflowdarstellung

Die Übersetzung der einzelnen syntaktischen Elemente des Prozessmodells ergibt eine Menge von Bausteinen, aus denen sich sämtliche Teile des Modells zusammensetzen lassen. Jeder Petrinetzbaustein ist das Ergebnis der Übersetzung eines Musters der Prozessmodellsyntax. Die Tabelle in Abbildung 5 verdeutlicht, wie die einzelnen Bestandteile des Prozessmodells in Petrinetze überführt werden. Im Folgenden sehen wir das

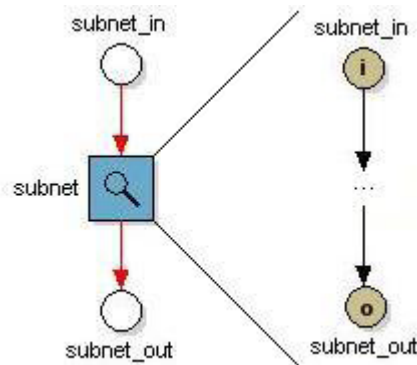


Abbildung 4: Subnetzdarstellung in Jasper

Übersetzungsprinzip und wie die Komposition der Bausteine funktioniert. Danach stellen wir die einzelnen Bausteine vor. Wir beschränken uns dabei auf die Petrinetzbausteine für Aktivitäten (einfache und komplexe), Subprozesse, nebenläufige Prozesse, Entscheidungen, Kontrollflüsse und Datenflüsse für transiente Daten, für die wir annehmen, dass sie konsumiert werden.

Prozessmodellmuster	→	Petrinetzübersetzung
Knoten	→	Baustein
Datenflusskante	→	Bogen
verbinden der Knoten per Kontrollfluss	→	verschmelzen der Schnittstellen
Gesamtprozess	⇒	Petrinetz

Abbildung 5: Zusammensetzung der übersetzten Elemente

4.3.2.1 Übersetzungsprinzip und Komposition der Bausteine

Jeder Petrinetzbaustein beginnt mit einem Startplatz (..._start) und endet mit einem Schlussplatz (..._end). Zwei Bausteine lassen sich folglich zusammensetzen, indem der Schlussplatz des einen mit dem Startplatz des nächsten Bausteins zu einem Platz verschmolzen wird.

Es gibt jedoch einige Ausnahmen: So werden die Bausteine für Entscheidung und Flow mehrere Schlussplätze und der Baustein für die Zusammenführung mehrere Startplätze haben. Weiterhin haben wir bei der Definition der Prozessmodellsyntax nicht ausgeschlossen, dass Knoten mehrere ein- und ausgehende Kanten haben können. Wir müssen also eine Möglichkeit finden, mehrere ein- und ausgehende Kontrollflusskanten zu modellieren. Bei Auswahl, Flow und Zusammenführung ist dies offensichtlich Teil der Bausteine. Für die Überführung von Knoten mit mehreren Anschlüssen benötigen wir jedoch Interfacebausteine (siehe Abb. 6), die den Start- bzw. den Schlussplatz des Bausteins aufspalten, sodass mehrere Anschlüsse entstehen.

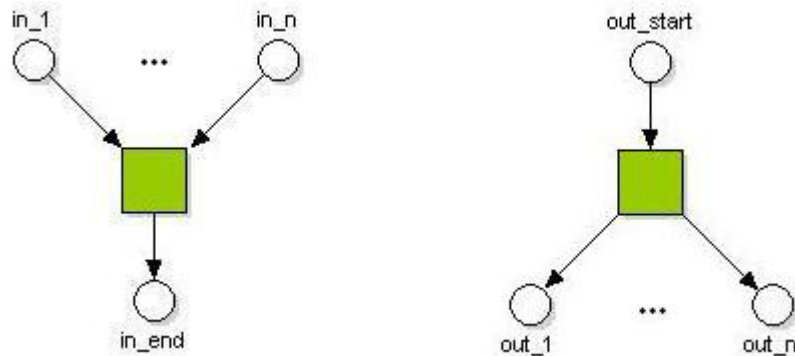


Abbildung 6: Interfaces für mehrere eingehende bzw. ausgehende Kontrollflusskanten

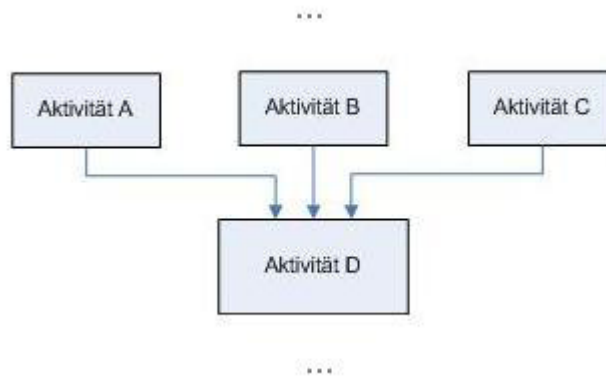


Abbildung 7: Beispiel für eine Aktivität mit mehreren eingehenden Kanten in Prozessmodellsyntax

Hat im Prozessmodell also eine Aktivität mehrere eingehende Kontrollflusskanten, so verbinden wir den entsprechenden Petrinetzbaustein mit dem Interface für mehrere Eingangskanten (in Abbildung 6 links), indem wir den Schlussplatz („in_end“) des Interfaces mit dem Startplatz des Bausteins zu einem Platz verschmelzen. Analog dazu verfahren wir mit Prozessmodellelementen, die mehrere ausgehende Kontrollflusskanten besitzen, indem wir das zweite Interface verwenden.

Ein so entstandenes *Petrinetzfragment* (kurz: *Fragment*) lässt sich nun mit mehreren anderen Bausteinen (oder Fragmenten) komponieren, indem wir je einen Startplatz des entstandenen Fragmentes mit dem Schlussplatz des anzuschließenden Bausteines verschmelzen.

Ein Beispiel: In Abb. 7 sehen wir die einfache Aktivität D mit drei eingehenden Kontrollflusskanten von den Aktivitäten A, B und C. Um diesen Prozessmodellausschnitt zu übersetzen, benötigen wir also vier Bausteine für einfache Aktivitäten und das Interface für drei eingehende Kanten. Wir verschmelzen die Schlussplätze der Aktivitätsbausteine A, B und C mit je einem Startplatz des Interfaces, sowie den Schlussplatz des Interfaces

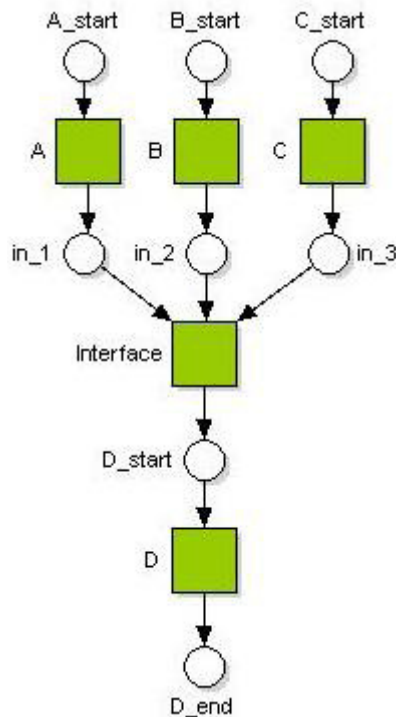


Abbildung 8: Beispiel für die Verwendung des Interfaces für mehrere eingehende Kanten (Übersetzung von Abb. 7)

mit dem Startplatz des Aktivitätsbausteins D und erhalten das Petrinetz in Abb. 8. Nachdem wir die Komposition der Bausteine betrachtet haben, wenden wir uns nun den einzelnen Petrinetzbausteinen zu.

4.3.2.2 Bausteine

Aktivitäten

Grundsätzlich existieren im Prozessmodell zwei Arten von Aktivitäten: einfache und komplexe. In Petrinetzen sind einfache Aktivitäten mit Transitionen gleichzusetzen. Komplexe Aktivitäten werden als subnet-Transitionen dargestellt (siehe Abb. 9). Die Übersetzung des durch eine komplexe Aktivität aufgerufenen Subprozesses in ein Petrinetz ist das zugehörige Subnetz zur subnet-Transition. Transitionen, die Aktivitäten verkörpern, nennen wir *Aktivitätstransitionen* (kurz: *A-Transitionen*).

Kanten

In den Petrinetzen gibt es keinen sichtbaren Unterschied zwischen Kontroll- und Datenflusskanten. Beide werden mit einfachen Bögen dargestellt.

Kontrollflüsse

Die Anfangs- und Endknoten des gesamten Prozesses werden als Plätze im Petrinetz dargestellt. Die Anfangs- und Endknoten von Subprozessen entsprechen also denjenigen Re-

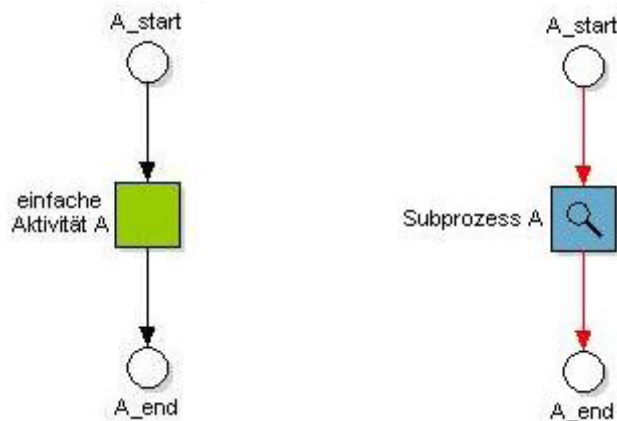


Abbildung 9: einfache und komplexe Aktivitäten

ferenzknoten im Subnetz, die sich nicht auf Datenflüsse beziehen.

Datenflüsse

Die Positionen, an denen Daten benötigt werden, modellieren wir ebenfalls mithilfe von Plätzen. Damit ergibt sich, dass Datenflusskanten durch Bögen repräsentiert werden, die Datenplätze mit Transitionen verbinden.

Bei Subprozessen entstehen durch die Modellierung mittels Subnetz-Transitionen im zugehörigen Subnetz Referenzplätze, die Kontroll- und Datenfluss weiterreichen.

Parametrisierung von Aktivitäten

Da die Darstellungen von einfachen und komplexen Aktivitäten sich sehr ähnlich sind, werden wir der Übersicht halber im Folgenden auf die Trennung zwischen beiden verzichten und nur noch die Parametrisierung von einfachen Aktivitäten darstellen. Komplexe Aktivitäten verhalten sich jeweils entsprechend.

notwendige Daten

Notwendige Ein- und Ausgabedatenflüsse einer Aktivität werden als zusätzliche ein- bzw. ausgehende Bögen der entsprechenden A-Transition dargestellt (siehe Abb. 10).

optionale Daten

Optionale Datenflüsse von bzw. zu einer Aktivität führen dazu, dass die Aktivität mithilfe mehrerer A-Transitionen dargestellt werden muss. Die Ausführung soll ja davon abhängen, ob Daten vorhanden sind oder nicht. Daher muss es für jede Möglichkeit der Ausführung mit oder ohne Daten eine eigene A-Transition geben.

Bei optionalen Ausgabedaten (Abb. 11) wird eine A-Transition mit und eine ohne Plätze für ausgehenden Datenfluss erzeugt.

Für Aktivitäten mit optionalen Eingabedaten verwenden wir wieder zwei A-Transitionen: eine, die die Ausführung mit Eingabedaten und eine, die die Ausführung ohne Eingabe-

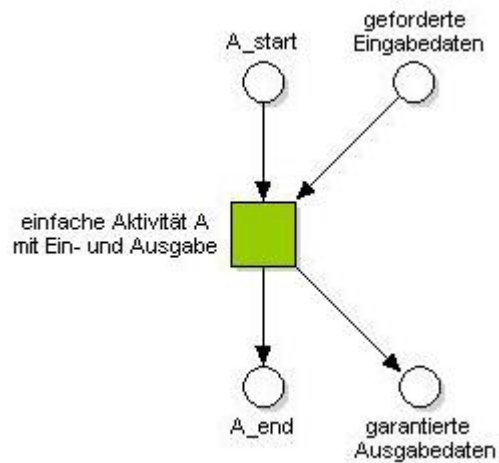


Abbildung 10: Aktivität mit zwingenden Ein- und Ausgabedaten

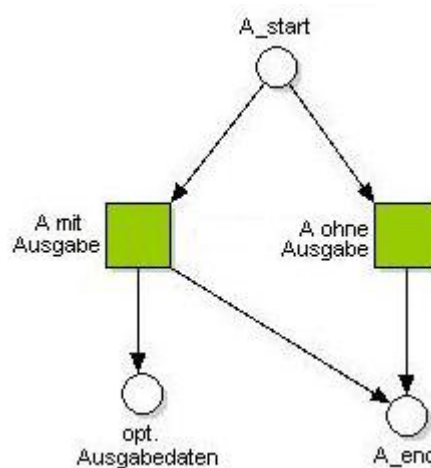


Abbildung 11: Aktivität mit optionalen Ausgabedaten

daten beschreibt. Wenn wir es dabei belassen, stoßen wir jedoch auf ein Problem: Betrachten wir Abbildung 12a, so können wir sehen, dass durch diesen Baustein noch nicht sichergestellt ist, dass Eingabedaten, wenn sie vorhanden sind, auch verwendet werden. Die A-Transition „A ohne Eingabe“ ist auch dann aktiviert, wenn der Platz „optionale Eingabedaten“ markiert ist. Dies können wir verhindern, indem wir einen Inhibitorbogen verwenden. In Abbildung 12b wird die A-Transition „A ohne Eingabe“ geblockt, wenn der Platz „optionale Eingabedaten“ markiert ist. Dieser Baustein stellt also die gewünschte Funktionalität dar.

Sind beide Arten von optionalen Daten vorhanden, muss es eine A-Transition für jede Kombination von Ein- und Ausgabedaten geben (siehe Abb. 13), wobei wieder das Vorhandensein eines Tokens im Eingabedatenplatz die A-Transitionen ohne Eingabedaten blockiert.

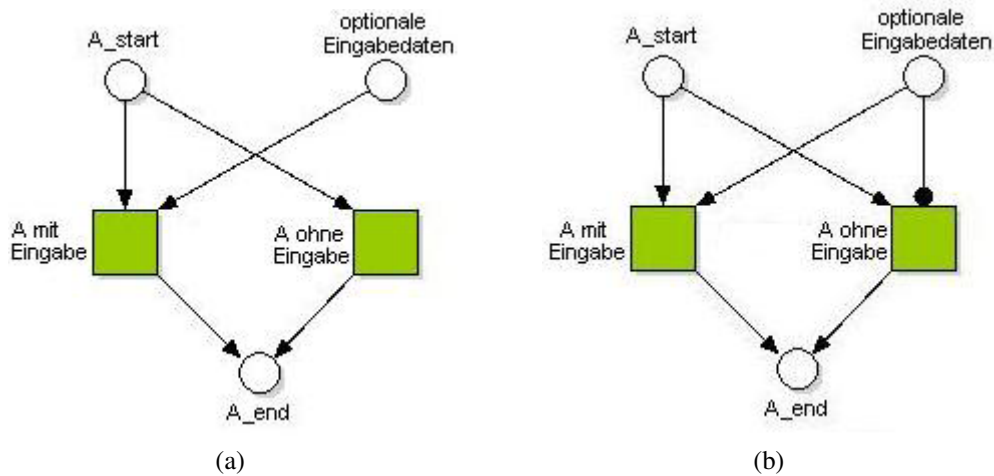


Abbildung 12: Aktivität mit optionalen Eingabedaten (a) ohne, (b) mit Inhibitorbogen

Normalerweise gibt es zwischen einer A-Transition und einem Datenplatz höchstens einen Bogen. Es kann jedoch bei komplexen Aktivitäten vorkommen, dass im zugehörigen Subnetz mehrere Knoten auf das gleiche Datum zugreifen. In einem solchen Fall würden wir gerne mehrere Kanten zwischen verschiedenen Transitionen und einem Referenzplatz einfügen und dies im übergeordneten Netz durch eine Kante zwischen subnet-Transition und Datenplatz repräsentieren. Da es in Jasper jedoch nicht möglich ist, in einem Subnetz mehrere Bögen zu einem Referenzplatz zu ziehen, müssen wir im übergeordneten Netz bereits zwischen der subnet-Transition und dem entsprechenden Datenplatz so viele Bögen einzeichnen, wie es Datenzugriffe im Subnetz gibt. Im Subnetz wird dann automatisch für jeden Zugriff eine Kopie des Referenzplatzes eingefügt.

optionale Ausführung einer Aktivität

Eine weitere Notation ist die optionale Ausführung. Hierbei wird eine Entscheidung getroffen, ob oder ob nicht eine Aktivität zur Ausführung gelangt. Daher wird dieses Notationselement mithilfe des oben erwähnten XOR-Elementes übersetzt (siehe Abb. 14).

externe Ereignisse

Die neu eingeführte Notation für Ereignisse haben wir als Baustein nicht explizit anders modelliert als eine durch den Kontrollfluss ausgelöste Aktivität. Die Besonderheit hierbei liegt in der Anbindung im Gesamtnetz. Der Startplatz einer ereignisgesteuerten Aktivität ist nicht Nachplatz einer anderen Transition, sondern wird als Schnittstellenplatz des Gesamtnetzes modelliert, der der „Umgebung“ des Netzes zugänglich ist. Diese Klasse von Petri-Netzen wurde als Open Workflow Net formalisiert, auf die wir hier nicht weiter eingehen (siehe [10]). In PNML repräsentieren wir Schnittstellenplätze über spezielle Namen: „event_input_A“ / „event_output_A“. Je nach zur Verfügung stehendem Werkzeug könnten auch die o.g. toolspezifischen Elemente von PNML genutzt werden.

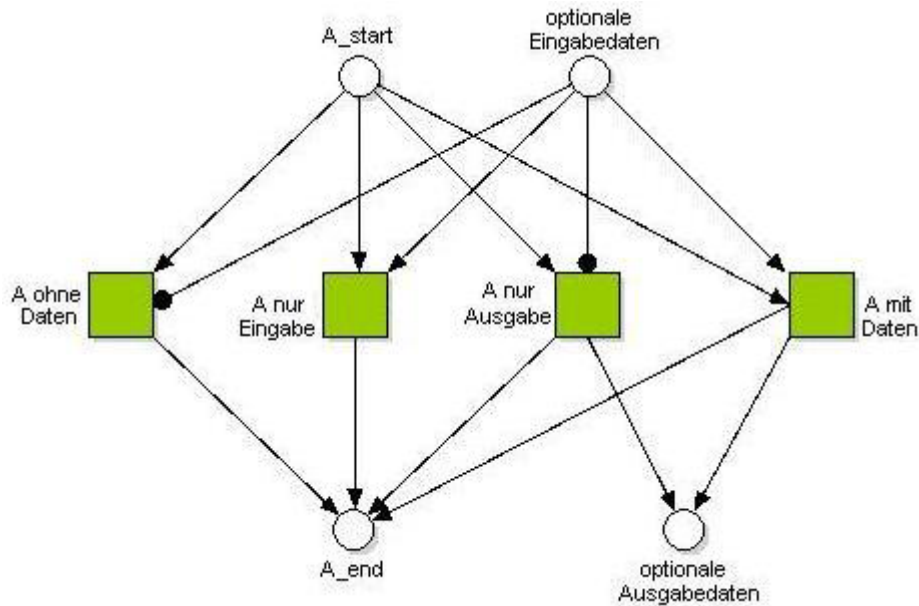


Abbildung 13: Aktivität mit optionalen Ein- und Ausgabedaten

nebenläufige Ausführung (Flow)

Diese wird durch Verzweigung an den Nachplätzen einer Transition vor der nebenläufigen Ausführung (synchroner Start) bzw. Zusammenführung der Vorplätze der nachfolgenden Transition (synchrones Ende) modelliert (siehe Abb. 15).

Entscheidungen**Auswahl (split)**

Der Kontrollfluss im Petrinetz wird hier mithilfe eines XORs geteilt.

Zusammenführung (merge)

Nach einer Entscheidung sollte der Kontrollfluss wieder vereinigt werden. Dies erreichen wir erneut durch eine XOR-Transition.

Beide Bausteine finden sich in Abb. 16.

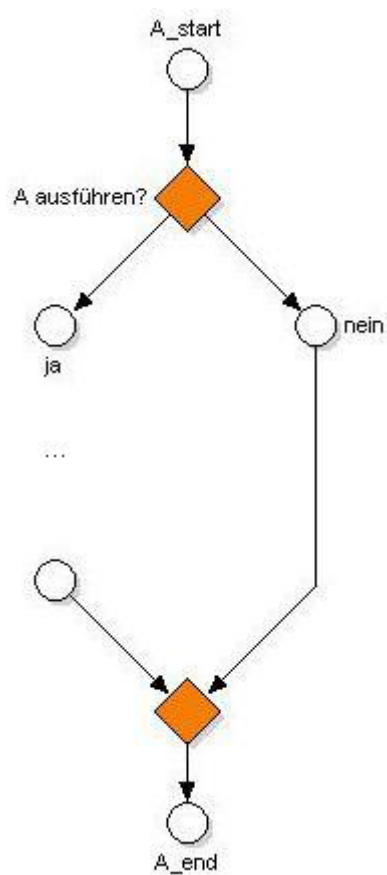


Abbildung 14: optionale Ausführung

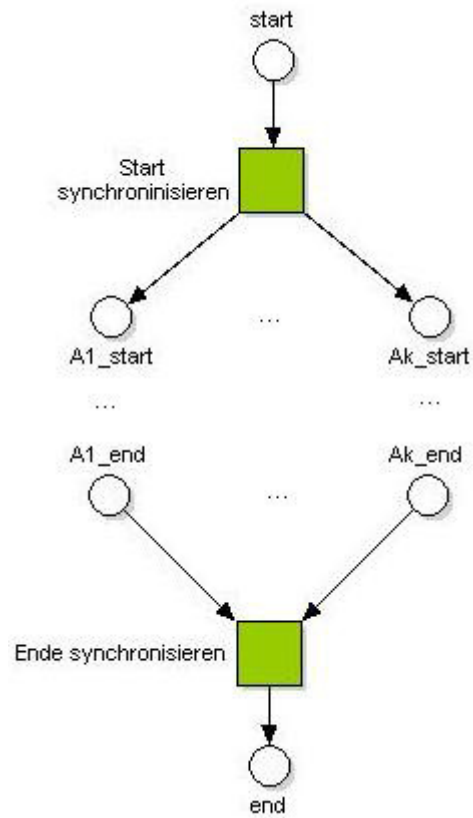


Abbildung 15: nebenläufige Ausführung mehrerer Aktivitäten

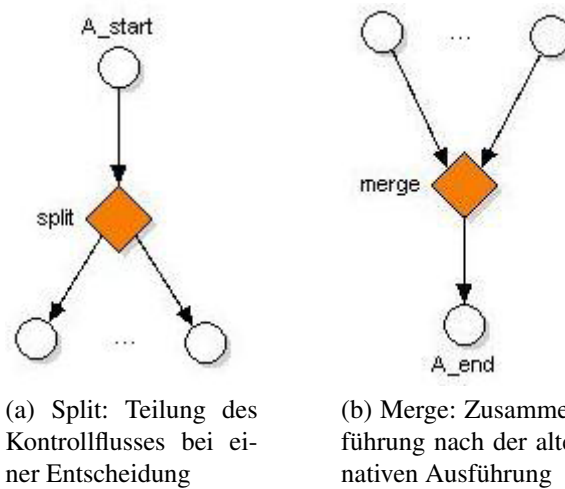


Abbildung 16: Entscheidung: (a) split, (b) merge

4.3.3 Beispiel für die Zusammensetzung der Bausteine

Mit Hilfe der oben beschriebenen Bausteine lässt sich nun das gesamte Prozessmodell in ein Petrinetz übersetzen. Um dies zu veranschaulichen, übersetzen wir hier beispielhaft die komplexe Aktivität „wissenschaftliche Entscheidung“ und ihren Subprozess (siehe Anhang A.2.1, S. 35 oben).

In der Gesamtsicht des Prozessmodells (Anhang A.2.1, S. 33) ist die „wissenschaftliche Entscheidung“ eine komplexe Aktivität mit notwendigen Ausgabedaten. Es ergibt sich also das in Abb. 17 dargestellte Petrinetzmuster. Zu beachten ist dabei der Datenfluss: Wenn wir uns den Subprozess ansehen, stellen wir fest, dass sich darin drei Aktivitäten befinden, die das Ausgabedatum „Entscheidung“ erzeugen können. Wir haben also drei Datenzugriffe im Subprozess und benötigen daher drei Bögen von der subnet-Transition zum Datenknoten. Damit entstehen im Subnetz drei Kopien des Datenreferenzknotens.

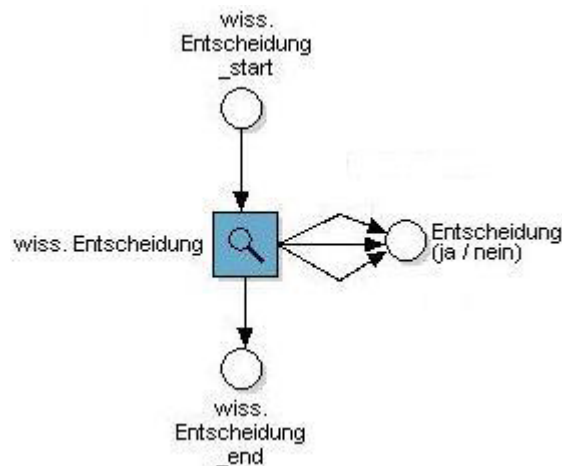


Abbildung 17: Teilprozess „wissenschaftliche Entscheidung“

Das zugehörige Subnetz entwickeln wir nun Schritt für Schritt (siehe Abb. 18): Der Kontrollflussanfang ist durch den Start-Referenzknoten der übergeordneten subnet-Transition bereits gegeben. Ihm folgt im Subprozess eine Entscheidung. Wir verwenden also zunächst den split-Baustein (Abb. 16a) und verschmelzen seinen Startplatz mit dem Start-Referenzknoten. Die Entscheidung benennen wir konform zum Prozessmodell „K.O. Kriterien?“.

Im Prozessmodell führt der „ja“-Zweig der Entscheidung (also zutreffende K.O.-Kriterien) zu einer einfachen Aktivität „Einsatz wegen K.O. Kriterien ablehnen“ mit garantiertem Ausgabedatum „Entscheidung = nein“. Wir verwenden also den Baustein für eine einfache Aktivität mit garantiertem Ausgabedatum, benennen die A-Transition entsprechend und verbinden seinen Startplatz mit dem Schlussplatz des split-Bausteins im „ja“-Zweig. Den Datenzweig des Bausteins verbinden wir mit einer der Kopien des Datenreferenzknotens.

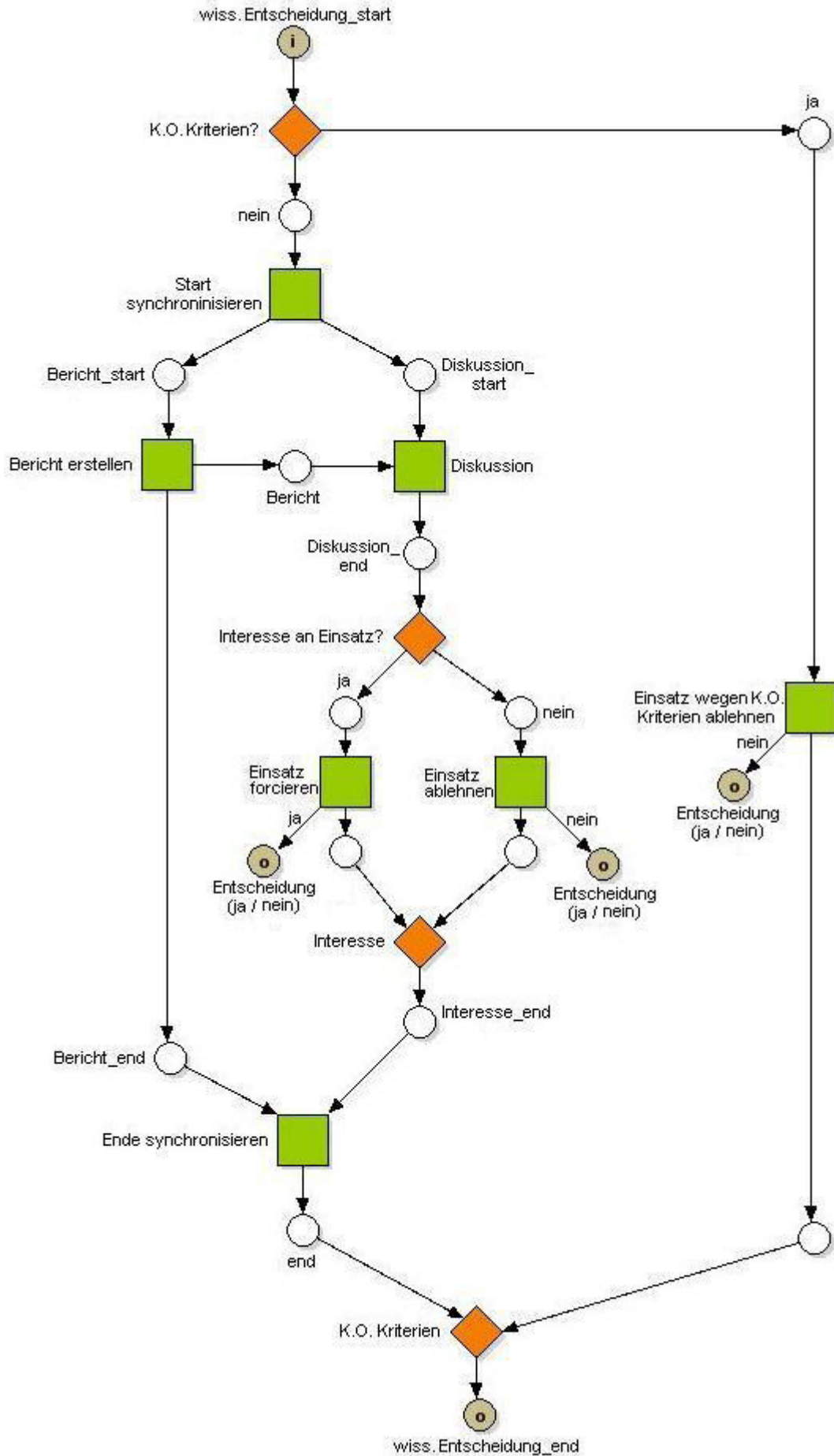
Vor dem Ende des Kontrollflusses im Prozessmodell werden die zu Beginn mit Hilfe der Auswahl getrennten Kontrollflüsse wieder vereinigt. Dies repräsentieren wir mit

dem merge-Baustein, dessen Schlussplatz wir mit dem End-Referenzknoten verschmelzen. Einen der Startknoten des merge-Bausteins verschmelzen wir mit dem Endknoten an der A-Transition „Einsatz wegen K.O. Kriterien ablehnen“.

In dieser Weise verfahren wir, um den gesamten Subprozess „wissenschaftliche Entscheidung“ in ein Petrinetz zu übersetzen. Das Ergebnis dieser Übersetzung findet sich im Anhang A.3 auf S. 47.

Betrachten wir den Subprozess „wissenschaftliche Entscheidung“ (siehe Prozessmodell im Anhang A.2.2, S. 35 oben) und das entstandene Petrinetz, können wir leicht feststellen, dass das Petrinetz den Prozess bezüglich der in Abschnitt 2 informal beschriebenen Semantik formal korrekt widerspiegelt.

A.3 Zusammensetzung der Petrinetzbausteine



Literatur

- [1] Task Force Erdbeben. Homepage der Task Force Erdbeben am GFZ Potsdam. http://www.gfz-potsdam.de/pb2/pb21/Task_Force/task1d.html, Januar 2008 (letzte Sichtung).
- [2] Dirk Fahland. diverse Gesprächsprotokolle. zu erfragen bei Herrn Fahland, 2007.
- [3] Heiko Woith. Protokolle des Sumatra-Bebens vom 12.9.2007, Versionen 1-3. zu erfragen bei Herrn Fahland, September 2007.
- [4] Rob Allen. Workflow: An Introduction. *Workflow Handbook*, 2001.
- [5] Dirk Fahland and Konstanze Swist. Legende der Prozessmodellsyntax. während der Anpassung des Modells geändert, März 2007.
- [6] Peter H. Starke. *Analyse von Petri-Netz-Modellen*. Teubner, 1990.
- [7] TU Eindhoven and Deloitte. Yasper - Homepage. <http://www.yasper.org>, Juni 2008 (letzte Sichtung).
- [8] Michael Weber and Ekkart Kindler. The Petri Net Markup Language. http://www2.informatik.hu-berlin.de/top/pnml/download/about/PNML_LNCS.pdf, April 2002.
- [9] Jonathan Billington and Søren Christensen et al. The Petri Net Markup Language: Concepts, Technology, and Tools. http://www2.informatik.hu-berlin.de/top/pnml/download/about/PNML_CTT.pdf, März 2003.
- [10] Wolfgang Reisig et al. Kommunizierende Workflow-Services modellieren und analysieren. *Informatik - Forschung und Entwicklung*, 20(1-2):90 – 101, Oktober 2005.
- [11] B. F. van Dongen, J. Mendling, and W. M. P. van der Aalst. Structural patterns for soundness of business process models. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 116–128, Washington, DC, USA, 2006. IEEE Computer Society.
- [12] Christian Stahl. Transformation von BPEL4WS in Petrinetze. Diplomarbeit, Humboldt-Universität zu Berlin, April 2004.
- [13] Niels Lohmann. A feature-complete Petri net semantics for WS-BPEL 2.0. In Marlon Dumas and Reiko Heckel, editors, *Web Services and Formal Methods, Forth International Workshop, WS-FM 2007 Brisbane, Australia, September 28-29, 2007, Proceedings*, volume 4937 of *Lecture Notes in Computer Science*, pages 77–91. Springer-Verlag, 2008.
- [14] Sebastian Hinz. Implementierung einer Petrinetz-Semantik für BPEL. Diplomarbeit, Humboldt-Universität zu Berlin, März 2005.

-
- [15] Chun Ouyang, Eric Verbeek, Wil M. P. van der Aalst, Stephan Breutel, Marlon Dumas, and Arthur H. M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Sci. Comput. Program.*, 67(2-3):162–198, 2007.
- [16] Harald Störrle. Semantics and Verification of Data Flow in UML 2.0 Activities, Juni 2004.
- [17] Harald Störrle. Semantics of Control-Flow in UML 2.0 Activities, März 2004.
- [18] Harald Störrle. Semantics of Exceptions in UML 2.0 Activities, April 2004.
- [19] Harald Störrle. Semantics of Structured Nodes in UML 2.0 Activities, Mai 2004.
- [20] Harald Störrle and Jan Hendrik Hausmann. Towards a Formal Semantics of UML 2.0 Activities. <http://www.pst.ifi.lmu.de/stoerrle/>, November 2004.
- [21] Tony Spiteri Staines. Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets. In *ECBS*, pages 191–200, 2008.
- [22] Wil M. P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information & Software Technology*, 41(10):639–650, 1999.
- [23] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Formal Semantics and Analysis of business process models in BPMN. *Inf. Softw. Technol.*, 50(12):1281–1294, 2008.
- [24] A.H.M. ter Hofstede N. Russell and W.M.P. van der Aalst. newYAWL: Specifying a Workflow Reference Language using Coloured Petri Nets. In K. Jensen, editor, *Proceedings of the Eighth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2007)*, volume 584, pages 107–126, Aarhus, Dänemark, Oktober 2007. University of Aarhus.