Automatisierte Compliance-Prüfung von Geschäftsprozessen

Inaugural-Dissertation
zur
Erlangung der Doktorwürde
der Wirtschafts- und Verhaltenswissenschaftlichen Fakultät
der Albert-Ludwigs-Universität Freiburg im Breisgau

vorgelegt von

Lutz Lowis
aus Mönchengladbach

im Sommersemester 2011

Dekan:
Prof. Dr. Dieter K. Tscheulin

Gutachtende:
Erstgutachter Prof. Dr. Dr. h.c. Günter Müller
Zweitgutachter Prof. Dr. Dirk Neumann

Datum des Promotionsbeschlusses:
20. September 2011

Inhaltsverzeichnis

1 Von manueller zu automatisierter Compliance-Prüfung 5
1.1 Einleitung . . . 5
1.2 Wirtschaftliche Betrachtung von Compliance . . . 6
1.3 Automatisierung von Compliance . . . 8
1.3.1 Bisherige IT-Unterstützung der Prüfung . . . 8
1.3.2 Prüfungszeitpunkte und Prüfungsebenen . . . 9
1.3.3 Schwachstellen im Informationsfluss . . . 12
1.3.4 Anwendungsfall „Modell-Prüfung vor der Ausführung“ . . . 14
1.4 Problemstellung und Lösungsansatz . . . 16
1.5 Aufbau der Arbeit . . . 17

2 Regeleinhaltung in Workflows 19
2.1 Regeln . . . 19
2.1.1 Compliance-Regularien . . . 19
2.1.2 Regelsprachen zur Modellierung . . . 29
2.1.3 Regel-Elemente . . . 30
2.2 Workflows . . . 33
2.2.1 Standardmodelle für Workflows . . . 33
2.2.2 Modellierung mit Petrinetzen . . . 36
2.2.3 Workflow-Elemente . . . 38
2.3 Fallbeispiel . . . 41
2.4 Bestehende Analyseverfahren . . . 43
2.4.1 Nicht-Petrinetz-basierte Verfahren . . . 46
2.4.2 Petrinetz-basierte Verfahren . . . 48
2.5 Eignung bestehender Verfahren zur Compliance-Prüfung . . . 54
2.5.1 Zu erfassende Informationsfluss-Eigenschaften . . . 54
2.5.2 Einordnung der bisherigen Verfahren . . . 62

3 Workflow-Analyse mit Reveal 67
3.1 Ablauf . . . 67
3.2 Transformation von Regeln und Workflows in Petrinetze . . . 68
3.2.1 Regel-Transformation . . . 68
3.2.2 Workflow-Transformation . . . 71
3.3 Vorlagen zur Formalisierung von Regeln . . . 74
3.3.1 Korrektheit . . . 74

3.3.2 Modalität . . . 78
3.3.3 Orte . . . 79
3.3.4 Reihenfolge . . . 80
3.3.5 Ressourcen . . . 84
3.3.6 Verwendung . . . 84
3.3.7 Zeitlimits . . . 86
3.4 Analysealgorithmus . . . 88
3.4.1 Verknüpfung von Regel- und Workflow-Petrinetzen . . . 90
3.4.2 Suchlauf . . . 90
3.4.3 Lokalisierung von Schwachstellen . . . 92
3.5 Implementierung . . . 94
3.5.1 Regel-Transformation . . . 96
3.5.2 Workflow-Transformation . . . 98
3.5.3 Analyse . . . 101

4 Evaluation **105**
4.1 Workflow-Prüfung durch REVEAL . . . 105
4.1.1 Korrektheit . . . 108
4.1.2 Datenfluss . . . 109
4.1.3 Kontrollfluss . . . 113
4.1.4 Widerspruchsfreiheit zwischen Daten- und Kontrollfluss . . . 120
4.1.5 Einhaltung von HIPAA-Regeln . . . 121
4.1.6 Geschwindigkeit der Prüfung . . . 123
4.2 Bewertung des REVEAL-Verfahrens . . . 124

5 Zusammenfassung und Ausblick **127**

Literaturverzeichnis **131**

Abbildungsverzeichnis **145**

Tabellenverzeichnis **149**

1 Von manueller zu automatisierter Compliance-Prüfung

1.1 Einleitung

Compliance, also die Regeleinhaltung und ihr Nachweis, ist für Unternehmen gesetzlich verpflichtend. Neben dem unternehmerischen Eigeninteresse, Abläufe zu überwachen, um sie verbessern zu können, besteht demnach auch ein Zwang zur Kontrolle. Die nötigen Compliance-Prüfungen sind allerdings so aufwändig, dass in den USA Schätzungen zufolge 8% des Bruttoinlandsprodukts dafür ausgegeben werden (Bace et al., 2006).

Den Prüfkosten gegenüber steht das finanzielle Risiko der Sanktionen. Diese fallen gerade dann sehr hoch aus, wenn ein Unternehmen im Schadensfall nicht nachweisen kann, nach bestem Gewissen (Stichwort „best practices“) für die Regeleinhaltung gesorgt zu haben. Bei Compliance-Verstößen drohen neben Geldstrafen auch Haftstrafen für die Verantwortlichen wie etwa nicht nur den Geschäftsführer, sondern auch den Leiter der Informationstechnologie (Chief Information Officer, CIO) (Berghel, 2005).

Der CIO wird deswegen mit zur Verantwortung gezogen, weil in Industriestaaten der überwiegende Teil der Geschäftsprozesse nicht mehr rein manuell ausgeführt wird. In Deutschland werden bereits zwei von drei Geschäftsprozessen durch Computersysteme unterstützt, Tendenz steigend (Padovan et al., 2005). Dennoch findet die Compliance-Prüfung bisher überwiegend manuell statt (Müller und Terzidis, 2008). Während die automatisierte Ausführung von Geschäftsprozessen also bereits weit fortgeschritten ist, steht die automatisierte **Prüfung** eher noch am Anfang. Im Bereich der flexiblen Geschäftsprozesse bedeutet dies etwa, dass Prozesse zwar dank Automatisierung schnell geändert und in geänderter Form ausgeführt werden können, z.B. durch Systeme wie Aristaflow (Dadam et al., 2009), die Prüfung aber wegen des manuellen Aufwands nicht Schritt halten kann.

Das ungeprüfte Ausführen von Geschäftsprozessen kann drastische volkswirtschaftliche Schäden nach sich ziehen, wie z.B. die Enron- oder Worldcom-Skandale (siehe Abschnitt 1.2) zeigten. Ein vielversprechender Ansatz, um Geschäftsprozesse ebenso schnell zu prüfen, wie sie geändert und ausgeführt werden, ist die **Automatisierung**.

Neben der Prüfungsgeschwindigkeit spielt der Prüfungsumfang eine wichtige Rolle. Je mehr Regeln ein Prüfverfahren bereits vor der Ausführung eines Geschäftsprozesses erfassen kann, desto mehr Regelverstöße lassen sich bei der Ausführung vermeiden. Vor diesem Hintergrund stellt die vorliegende Arbeit ein zur **Entwurfszeit** einsetzbares Verfahren zur automatisierten Compliance-Prüfung von Geschäftsprozessen vor.

Anstelle des Begriffs „Geschäftsprozess“ wird im Folgenden der Begriff **Workflow** verwendet, um den Bezug zur Automatisierung zu betonen. Dabei wird auf die Definitionen von Hansen und Neumann (2005) zurückgegriffen:

- **Geschäftsprozess**: „Ein Geschäftsprozess besteht aus einer Menge miteinander verknüpfter Aktivitäten, welche in einer bestimmten Reihenfolge ausgeführt werden, um ein festgelegtes Ziel zu erreichen. Die verschiedenen Aktivitäten können sequenziell und/oder parallel gestartet und ausgeführt werden.“ (Hansen und Neumann, 2005, S. 233, Kap. 2.2.5.1)

- **Workflow**: „Ein vollständig oder teilweise automatisierter Geschäftsprozess wird als Workflow bezeichnet.“ (Hansen und Neumann, 2005, S. 234, Kap. 2.2.5.1)

Der Workflow-Begriff hebt also die informatische Sicht auf Abläufe hervor, während beim Geschäftsprozess-Begriff die wirtschaftlichen Ziele, zu denen ein Ablauf führen soll, im Vordergrund stünden. Die Frage, ob der zu untersuchende Ablauf wirtschaftlich sinnvoll ist, ist aber nicht Teil der Compliance-Betrachtungen. Es zählt nur die Frage, ob die aufgestellten Regeln eingehalten werden.

Ein Workflow besteht aus Aktivitäten, die in einer bestimmten Reihenfolge ausgeführt werden und dabei Daten austauschen. Die Ausführungsreihenfolge der Aktivitäten wird als **Kontrollfluss** bezeichnet, der bei der Ausführung entstehende Datenaustausch als **Datenfluss** (siehe Abschnitt 1.3.3). Die Teile des Workflows, in denen der Kontroll- oder Datenfluss möglicherweise einen Regelverstoß hervorrufen, werden im Folgenden als **Schwachstellen** bezeichnet (siehe Abschnitt 1.3.3). Die Compliance-Prüfung von Geschäftsprozessen wird in dieser Arbeit auf den informatischen Bereich der Regeleinhaltung in Workflows eingegrenzt und dort als Schwachstellensuche im Daten- und Kontrollfluss umgesetzt.

Für die automatisierte Prüfung von Workflows gibt es bereits Ansätze. Diese zielen aber vor allem auf die syntaktische und funktionale Korrektheit eines Workflows ab, beispielsweise auf die Abwesenheit sogenannter Deadlocks. Das sind Stellen, an denen die Ausführung in einem unendlichen Wartezustand blockiert. Compliance beinhaltet aber - über die grundsätzlich weiterhin nötige syntaktische Korrektheit hinaus - vor allem semantische und nicht-funktionale Regeln. Diese werden von bisherigen Verfahren nicht im nötigen Ausmaß berücksichtigt, wie im zweiten Kapitel gezeigt werden wird.

1.2 Wirtschaftliche Betrachtung von Compliance

Bislang werden Compliance-Prüfungen hauptsächlich manuell durchgeführt (Müller und Terzidis, 2008). Dadurch entstehen einerseits hohe Kosten für die vielen Arbeitsstunden entsprechend qualifizierter und zertifizierter Prüfer. Andererseits können nicht viele Unternehmen pro Jahr geprüft werden, weil eine einzelne Prüfung sehr lange dauert. In Deutschland beispielsweise muss ein Unternehmen etwa alle 39.000 Jahre eine Datenschutzkontrolle „befürchten“ (Xamit, 2009).

Der größte Faktor sowohl bei den Prüfkosten wie auch bei der Aufdeckungswahrscheinlichkeit ist die Zeit. Eine Annahme der vorliegenden Arbeit ist, dass sich die für eine Compliance-Prüfung benötigte Zeit drastisch reduzieren lässt, wenn ein automatisiertes Verfahren für die Prüfung eingesetzt wird. Nachfolgend wird aufgezeigt, wie hoch die Kosten für Compliance sind und welche Einsparungen möglich sind.

Laut Umfragen der Financial Executives International (FEI) erwarteten die größten US-amerikanischen Unternehmen im Schnitt 35.000 Stunden Arbeitsaufwand zur Anpassung an SOX und eine Erhöhung ihrer Audit-Ausgaben um etwa 40 Prozent (Financial Executives International (FEI), 2004). Tatsächlich stiegen im Zeitraum von 2003 bis 2006 die jährlichen Audit-Ausgaben um knapp 60 Prozent an. Gegenüber den ursprünglichen Ausgaben ohne die Compliance-Auflagen stiegen die Audit-Kosten um über 180 Prozent bei den größten, 250 Prozent bei den mittelgroßen und über 300 Prozent bei den kleinen Unternehmen. Etwa ein Viertel der befragten Unternehmen gab an, den Compliance-Auflagen ausweichen zu wollen, indem sie sich von der Börse zurückziehen oder aufkaufen lassen (Hartman, 2007).

Die jährlichen Kosten für Compliance sind seit Einführung der Regularien um ein Drittel gesunken. Einer der Hauptgründe für diesen Rückgang war die verstärkte Automatisierung (Financial Executives International (FEI), 2007). Zum Beispiel wurden in einem Fall die Kosten für die Eigenentwicklung eines Systems zur automatisierten Compliance-Überwachung auf eine Million Dollar geschätzt; die Einrichtung eines bestehenden, kommerziellen Systems kostete letztendlich eine Viertelmillion Dollar. Unabhängig davon, ob das automatisierte System selbst entwickelt oder eingekauft wurde, schätzte Forrester Consulting die anschließenden jährlichen Kosten auf etwa 100.000 Dollar (North, 2008). Dies entspricht einem Zehntel der bei manuellen Prüfungen mindestens anfallenden Kosten von jährlich einer Million Dollar (Hartman, 2007).

Compliance mit PCI-DSS, dem Datenschutzstandard der Bezahlkartenindustrie (Payment Card Industry, 2009), verursacht pro Unternehmen jeweils etwa 225.000 Dollar jährliche Kosten (Ponemon Institute, 2010) für die nötigen Audits durch sogenannte Qualified Security Assessors. Für die große Mehrheit der Unternehmen gelten verschiedene Regularien gleichzeitig, so dass beispielsweise SOX-Compliance *und* PCI-DSS-Compliance erreicht werden müssen. Im allgemeinen Fall lässt sich nicht sagen, wie die Kosten der Prüfung gegen mehrere Regularien sich von denen der Einzelprüfung unterscheiden. Logisch erscheint, dass mit steigender Unterschiedlichkeit der Regularien die Kosten immer näher an die Summe der Einzelprüfungen geraten. Selbst bei hoher Ähnlichkeit der Regularien steigen allein weger der Regelmenge die Gesamtausgaben für die Prüfung mit der Anzahl der anzuwendenen Regularien. Der größte Kostenfaktor bei den Prüfungen ist die Arbeitszeit der hochqualifizierten und spezialisierten Auditoren. Der obigen Annahme folgend, dass sich die Prüfungszeit durch Automatisierung stark reduzieren lässt, können mit einem automatisierten Verfahren in der gleichen Zeit mehr Prüfungen stattfinden. Sofern ausser der benötigten Zeit die sonstigen Umstände gleich bleiben, lassen sich durch eine Automatisierung der Prüfung die Prüfkosten senken.

1.3 Automatisierung von Compliance

Von den wirtschaftlichen Anreizen abgesehen wird die Automatisierung von Compliance auch durch Regelwerke angetrieben. Sicherheitsrahmenwerke wie der IT-Grundschutz des BSI (Bundesamt für Sicherheit in der Informationstechnik (BSI), 2008) oder die *Control OBjectives for Information and related Technology* (COBIT) (IT Governance Institute, 2007) verlangen die Wiederhol- und Nachvollziehbarkeit der Prüfung. COBIT definiert als einen von vier Bereichen *Monitor and Evaluate*, worin mit *ME2: Monitor and Evaluate Internal Control* sowie *ME3: Ensure Compliance With External Requirements* ganz konkret die Überwachung der Regeleinhaltung gefordert wird. Die Überwachung bzw. Prüfung soll möglichst automatisiert erfolgen und häufig stattfinden.

1.3.1 Bisherige IT-Unterstützung der Prüfung

Die bestehende IT-Unterstützung lässt sich in zwei Bereiche aufteilen: Software, die die Dokumentation der Prüfung unterstützt, und Software, die die eigentliche Prüfung im Sinne eines Verfahrens unterstützt.

Software zur Prüfungsdokumentation

Bei heutigen Compliance-Prüfungen kommt meist nur Standard-Bürosoftware zur Verwaltung von Checklisten und Tabellen zum Einsatz. Dementsprechend lange und kostspielig ist die Prüfung (vgl. Abschnitt 1.2). Die einzelnen, beispielsweise in Excel-Tabellen festgehaltenen Prüfungsergebnisse schnell und fehlerfrei zu aggregieren, ist eine manuell schwer zu bewältigende Aufgabe.

Unterstützende Software bietet Checklisten und Formulare, die den Prüfer durch einzelne Prüfungsphasen führen und die Ergebnisse automatisiert aggregieren. Die Beantwortung der einzelnen Fragen bzw. die jeweilige Bewertung der Regeleinhaltung muss dabei aber weiterhin manuell erfolgen.

Software zur Prüfung

Eine wachsende Anzahl Softwareunternehmen entdeckt den Compliance-Markt für sich und beginnt die Entwicklung von Prüfsoftware. „Ecora Auditor Pro“ beispielsweise wird mit vielversprechenden Aussagen wie „Einsparungen von 95 Prozent der Prüfzeit“ und besonderer Eignung für einzelne Regularien wie SOX beworben. In diesem wie in vielen anderen Fällen wird aber letztendlich nur ein Überblick über die Konfiguration bestehender technischer Maßnahmen gegeben, wie sie Teil jeder guten Systemadministrationslösung sind. Zugriffsrechte der Benutzer und Benutzergruppen sowie die protokollierten Zugriffe auf ausgewählte Ressourcen werden angezeigt.

Stark technisch orientierte Regularien wie PCI DSS (Payment Card Industry, 2009) lassen sich mit solchen Softwarelösungen weitgehend erfassen. Aber auch hier gilt das

allgemeine Problem der Analyse zur Logzeit, dass die Schwachstellen erst im bestehenden System nach der Ausführung entdeckt werden. Entsprechende Reporting-Ansätze sind ein wichtiger Bestandteil von Compliance-Bemühungen. Dem Reporting sollten aber Entwurfszeit-Prüfungen voraus gehen, um möglichst früh Schwachstellen entdecken zu können. Wie in Abschnitt 1.3.2 diskutiert, lassen sich Schwachstellen deutlich günstiger beheben, wenn sie früh erkannt werden.

Über Logging- und Reporting-Funktionalität hinausgehende Software zur Compliance-Prüfung gibt es bisher nicht für den industriellen Einsatz. Einzelne der in Abschnitt 2.4 vorgestellten theoretischen Ansätze zur automatisierten Prüfung wurden als Prototyp implementiert, aber - soweit aus den Angaben der jeweiligen Autoren entnehmbar - nur in Forschungsprojekten im Fallstudienrahmen eingesetzt.

Der aktuelle Stand der IT-Unterstützung bei der Compliance-Prüfung ist also der, dass einerseits Software zur Erleichterung der Dokumentation existiert und eingesetzt wird. Andererseits gibt es bisher im industriellen Umfeld zur manuellen Prüfung keine Alternativlösung, die mehr als Logging und Reporting bietet. Es besteht daher gerade in dem aus Prüfungssicht so besonders attraktiven Feld der Entwurfszeit-Prüfung von Workflows Bedarf an neuen Lösungen.

1.3.2 Prüfungszeitpunkte und Prüfungsebenen

Prüfungszeitpunkte

Für die Prüfung auf Regeleinhaltung kommen grundsätzlich drei verschiedene Zeitpunkte in Frage: *vor* der Ausführung, *während* der Ausführung und *nach* der Ausführung. Abbildung 1.1 veranschaulicht die Dreiteilung der Prüfungszeitpunkte und zeigt typische Verfahren.

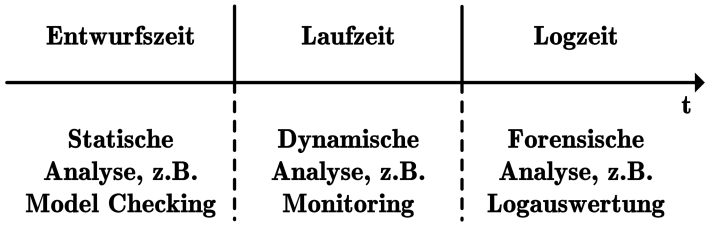


Abbildung 1.1: Prüfungszeitpunkte

Zur **Entwurfszeit**, also *vor* der Ausführung eines Prozesses oder einer Software, kann eine statische Analyse erfolgen. Statisch heißt diese Analyse deswegen, weil ihr keine Informationen über die dynamischen Aspekte der Ausführung vorliegen. Die Kontextinformationen eines Prozesses oder einer Software, wie z.B. genaue Ausführungszeitpunkte oder Variablenbelegungen, werden erst bei der Ausführung bekannt und können demnach vor der Ausführung nur statisch approximiert werden. Für eine Bestellung in einem Geschäftsprozess beispielsweise kann die genaue Rechnungssumme

erst während der Ausführung ermittelt werden, es kann aber eine statische Wertebereichsschätzung von z.B. 10 bis 5000 EUR erfolgen. Häufig ergeben sich aus der Ungenauigkeit der statischen Schätzung falsche Warnungen, sogenannte *false positives*, oder falsche Entwarnungen, sogenannte *false negatives*. Welche Herausforderungen daraus im industriellen Umfeld erwachsen, zeigen Bessey et al. (2010).

Ein Beispiel für typischerweise zur Entwurfszeit angewendete Prüfverfahren ist das Model Checking. Auch das von Accorsi und Wonnemann (2011) vorgeschlagene Verfahren ist ein Beispiel für eine Prüfung zur Entwurfszeit. So wie der Mangel an Kontextinformationen einen der größten Nachteile der Entwurfszeit-Analysen darstellt, so stellt die bereits durch den Entwurf erreichbare Vermeidung von Verstößen einen der größten Vorteile dar.

Zur **Laufzeit**, also *während* der Ausführung, werden die Kontextinformationen bekannt. Mithilfe eines sogenannten Monitors (Schneider, 2000; Accorsi et al., 2008), wohl den typischsten Fall der Laufzeitanalyse darstellend, kann während der Ausführung überwacht werden, ob sich das ausführende System in einem regelkonformen Zustand befindet. Sobald ein Regelverstoß erkannt wird, kann die Ausführung abgebrochen oder eine Warnung ausgegeben werden. Einige Monitore verwenden aus der statischen Analyse gewonnene Informationen, um die wahrscheinlichen Systemzustände nach jedem Ausführungsschritt vorherzusagen. Anhand solcher Monitore ist es möglich, die Ausführung (kurz) vor einem Regelverstoß abzubrechen, was einen Vorteil gegenüber der Logzeitanalyse bedeutet. Es besteht dann allerdings die Gefahr, wegen einer falschen Warnung eine regelkonforme Ausführung irrtümlich zu stoppen, oder - je nach Anwendungsfall schlimmer - eine regelwidrige Ausführung wegen einer falschen Entwarnung fortzusetzen. Sackmann et al. (2009) beschreiben ein Verfahren, anhand dessen während der Workflow-Ausführung eine risikobasierte Dienstauswahl umgesetzt werden kann, das heißt je nach Art und Eintrittswahrscheinlichkeit eines drohenden Verstoßes die Ausführung mit Ersatzdiensten fortzusetzen.

Ein Nachteil der Laufzeitanalyse ist die Tatsache, dass das ausführende System seine Ressourcen nicht mehr alleinig der Ausführung des Prozesses widmen kann, sondern einen Teil vor allem der Rechenzeit zur Überwachung des Prozesses benutzen muss. Als klassisches Beispiel sei hier die Virenprüfung auf Desktopcomputern genannt, die je nach Implementierung einen großen Teil der Ressourcen beansprucht.

Zur **Logzeit**, also *nach* der Ausführung, liegt die Maximalmenge an Informationen über die Ausführung vor, nämlich nicht nur die Kontextinformationen der Laufzeit sondern auch zusätzliche Informationen beispielsweise über die Ergebnisse parallel abgelaufener Prozesse. Verfahren wie das von Accorsi (2008) vorgeschlagene können daher im Falle von Regelverstößen den „Tathergang“ oftmals präzise rekonstruieren und teilweise sogar Folgen der Verstöße aufzeigen. Dieser aus technischer Sicht hochinteressante Zeitpunkt der Analyse hat jedoch einen entscheidenden Nachteil: Verstöße müssen erst eingetreten sein, um sie entdecken zu können.

Der Ansatz des Penetration Testings fällt insoweit in die Logzeit, dass dabei ein bereits implementiertes, häufig im Produktivbetrieb befindliches System bzw. eine Re-

produktion dessen untersucht wird. Im Gegensatz zu den während einer Entwurfszeit-Prüfung im Modell entdeckten Schwachstellen eines dann noch nicht existenten Systems oder Prozesses können gefundene Schwachstellen beim Penetration Testing bereits ausgenutzt worden sein.

Unabhängig von der Prüfungsebene ist die Fehler- bzw. Schwachstellenbehebung immer günstiger, je eher die Entdeckung erfolgt. Auf der Software-Ebene beispielsweise hat eine Untersuchung gezeigt, dass die spätere Behebung bis zu 200-mal teurer als die frühere Behebung war (Boehm, 1981). Vorausgesetzt, dass sich die Prüfkosten zu allen Zeitpunkten gleichen, ist damit der frühestmögliche Zeitpunkt wirtschaftlich gesehen der beste Prüfungszeitpunkt. In der vorliegenden Arbeit wird deswegen die Entwurfszeit als Prüfungszeitpunkt gewählt.

Prüfungsebenen

Orthogonal zu den Prüfungszeitpunkten lassen sich Prüfungsebenen definieren. Geschäftsprozesse bzw. Workflows stellen die oberste Ebene dar. Diese haben einen klar spezifizierten Anfangs- und Endzeitpunkt und bestehen aus einer festgelegten Anzahl und Reihenfolge ausgewählter Aktivitäten. Die mittlere Ebene beschreibt Anwendungen bzw. je nach Sichtweise (Stichwort Service-Orientierung) Dienste, die die vom Unternehmen benötigten Aktivitäten umsetzen oder steuern. Bei den Aktivitäten kann es sich sowohl um rein manuell, gemischt manuell-automatisiert oder um rein automatisiert ausgeführte handeln.

Zwischen den Anwendungen und der Hardware liegt die Software- bzw. Quellcode-Ebene. Eine Anwendung kann aus verschiedenen Software-Paketen zusammengesetzt sein; die Software-Ebene enthält die Implementierungsdetails dieser Pakete. Darunter liegt als unterste Ebene die Hardware, die letztlich für die Ausführung der teil- oder vollautomatisierten Aktivitäten zuständig ist. Abbildung 1.2 verdeutlicht die Einteilung. Wie sich Schwachstellen aus den unteren Ebenen auf die oberen Ebenen auswirken können, untersuchen Lowis et al. (2008). Einen Klassifikationsansatz für Schwachstellen in den auf eine service-orientierte Architektur (SOA) aufbauenden Diensten bieten Lowis und Accorsi (2009).

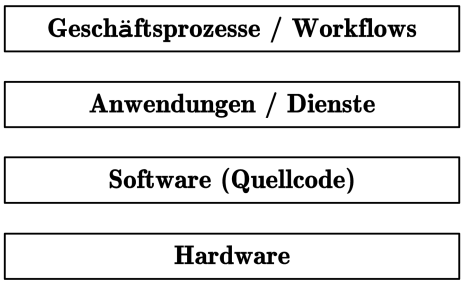


Abbildung 1.2: Prüfungsebenen

Die für eine automatisierte Prüfung nötigen Formalismen sind erstmalig auf Workflow-

Ebene vorhanden. Abgesehen vom Versuch einer automatisierten Analyse natürlichsprachlicher Texte, wie sie eventuell für Geschäftsprozesse vorhanden sind, stellen Workflows die ersten automatisiert analysierbaren Beschreibungen der Abläufe in einem Unternehmen dar. Die darunterliegenden Ebenen Anwendungen, Software und Hardware setzen diese Workflows „nur“ um. Die Workflow-Ebene als oberste bzw. „früheste“ analysierbare Ebene wird daher in der vorliegenden Arbeit als Prüfungsebene gewählt.

1.3.3 Schwachstellen im Informationsfluss

Workflow-Modelle beschreiben hauptsächlich, aus welchen Aktivitäten ein Workflow besteht und welche Daten zwischen diesen Aktivitäten ausgetauscht werden. Vom Start bis zum Ende des Workflows gesehen stellt der Datenaustausch einen **Datenfluss** dar. Aus der gleichen Sicht ergibt sich auf Aktivitäten bezogen der **Kontrollfluss**, der vor allem die Reihenfolge der Aktivitäten beschreibt. Der passende Oberbegriff lautet **Informationsfluss (IF)**, und die Prüfung des Informationsflusses, also sowohl des Datenflusses wie auch des Kontrollflusses, wird als Informationsflussanalyse bezeichnet. Diese Klarstellung ist notwendig, weil in manchen Arbeiten (z.T. auch in den in Abschnitt 2.4 zitierten) stellenweise *nur* der Datenfluss oder *nur* der Kontrollfluss als IF bezeichnet werden.

Grundsätzlich wird bei der Informationsflussanalyse zwischen **impliziten** und **expliziten Informationsflüssen** unterschieden (Sabelfeld und Myers, 2003). Implizite Flüsse entstehen aus all jenen Informationskanälen, die ursprünglich nicht zur Informationsübertragung gedacht waren, den sogenannten *covert channels* (Lampson, 1973). Abbildung 1.3 zeigt die typischerweise betrachteten Arten von Informationsflüssen.

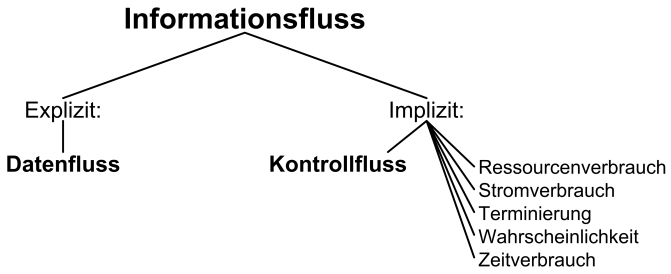


Abbildung 1.3: Arten von Informationflüssen (nach Sabelfeld und Myers (2003))

Zur Veranschaulichung der ursprünglich für Quellcode aufgestellten Einteilung soll pro Informationsflussart ein Workflow-Beispiel aus dem Umfeld medizinischer Behandlungen dienen. Bei den impliziten Informationsflüssen ist zusätzlich je ein Beispiel für mögliche Schlussfolgerungen angegeben, die aus den jeweiligen Informationen gezogen werden könnten.

- Expl. IF - Datenfluss: Patientendaten aus der Untersuchung.

- Impl. IF - Kontrollfluss: Abfolge der Behandlungsschritte. Erfolgt ohne weitere Schritte sofort die Operation, ist es ein Notfall.
- Impl. IF - Wahrscheinlichkeit: mit anderer Wahrscheinlichkeit als nur 0 oder 1 ausgeführte Aktivitäten. Gibt der Arzt zu 90% den Patienten die Hand, bei Norovirus-Befall aber nicht, kann der Beobachter auf den vermuteten Befund schließen. Wird die Hand nicht gegeben, ist die Folgerung aber nicht sicher.
- Impl. IF - Terminierung: Beendigung bzw. Abbruch einer Behandlung. Abbruch einer Behandlung deutet auf Änderung des Gesundheitszustands hin.
- Impl. IF - Zeitablauf: Zeitpunkt oder Dauer bestimmter Behandlungsschritte. Dauert eine Routineoperation länger, weist das auf Komplikationen hin.
- Impl. IF - Ressourcenverbrauch: Menge des eingesetzten Materials oder Personals. Schwerkranke brauchen häufig mehr Medikamente. (Bei Software eher im Sinne des Aufbrauchens von Ressourcen, z.B. „out of memory“-Fehler.)
- Impl. IF - Stromverbrauch: Mehrverbrauch an Energie bei der Ausführung einzelner Behandlungsschritte. (Bei Software besonders zur Kryptoanalyse von Smartcards relevant.)

Unabhängig davon, ob es sich um Quellcode oder einen Workflow handelt, soll es einem Angreifer nicht möglich sein, unerlaubten Zugriff auf verwendete Daten zu bekommen oder Rückschlüsse aus dem beobachteten Ablauf zu ziehen. Typischerweise gewinnt ein Angreifer aus einem impliziten IF bzw. *covert channel* nur ein Bit Information, was einer Ja/Nein-Entscheidung entspricht (das obige Notfallbeispiel aufgreifend also: handelt es sich um einen Notfall oder nicht). Selbst wenn es möglich ist, mehr Informationen zu gewinnen, so gelingt dies dem Angreifer nur mit großem Aufwand in überproportional langer (nicht-polynomialer) Zeit im Verhältnis zur Größe des Geheimnisses bzw. der Menge gewonnener Informationen (Askarov et al., 2008). Außerdem ist es schwierig, wenn nicht sogar unmöglich, in einem ressourcenteilenden Computersystem, wie es die typische Grundlage für Workflows mit IT-Unterstützung ist, solche verdeckten Informationskanäle zu schließen (Lipner, 1975). IF-Analyse wurde deswegen lange als theoretische Spielerei ohne praktische Einsatzfähigkeit betrachtet (Emerson, 2008).

Über die Zertifizierung von Software hinsichtlich sicherer Informationsflüsse haben Denning und Denning (1977) schon 1977 diskutiert. Lange Zeit wurden die entsprechenden Analysen aber nur für Hochsicherheitsumgebungen als relevant eingestuft. Von Brumley und Boneh (2003) wurde dann im Jahr 2003 beschrieben, wie es anhand Zeitmessungen bei Protokollinteraktionen möglich ist, private OpenSSL-Schlüssel aus einem Server im lokalen Netzwerk zu extrahieren. Trotz dieser praktischen Demonstration, dass implizite Informationsflüsse auch in Nicht-Hochsicherheitsumgebungen ein Sicherheitsrisiko darstellen, werden weiterhin explizite Datenflüsse bei der Sicherheitsanalyse wichtiger eingestuft als implizite (Huynh und Miller, 2010).

Explizite Informationsflusskanäle wurden bereits im Jahr 1973 von Lampson (1973) als *legitimate channels* kategorisiert. Der Kürze halber werden explizite Informationsflüsse in dieser Arbeit als **Datenfluss** bezeichnet, während die impliziten Informationsflüsse unter dem Begriff **Kontrollfluss** zusammengefasst werden. Soweit nicht anders angegeben, bezieht sich die Workflow-Prüfung im Folgenden immer auf die Prüfung der durch ein Workflow-Modell festgelegten Daten- und Kontrollflüsse.

Worauf sollen der Daten- und der Kontrollfluss nun untersucht werden? Avizienis et al. (2004) zufolge werden „faults, errors and failures“ unterschieden, übersetzt etwa „Mängel, Abweichungen und Versagen“. Ein Mangel führt möglicherweise zu einer Abweichung, die wiederum ein Versagen des untersuchten Systems auslösen kann. In gleicher Weise definiert Neuman Levine (2009) „defect (flaw), error and failure“ bezüglich Software-Diensten. Den Fehler-Einteilungen gegenüber steht die Definition von Bishop (2004) (Kap. 20, S. 389), der eine **Schwachstelle** (*vulnerability*) als das Versagen von Kontrollen bezeichnet und die Ausnutzung einer solchen Schwachstelle zur Verletzung der aufgestellten Sicherheitsregeln als *exploit*.

Die Bedeutung der Regeln und damit der Compliance-Regularien als Regelquellen hervorhebend wird in dieser Arbeit der Begriff der Schwachstelle bevorzugt. Anstelle von Software-Schwachstellen, die auf Quellcode-Ebene schon häufig klassifiziert wurden (siehe Abschnitt 2.4), geht es im vorliegenden Fall um strukturelle Schwachstellen in Workflows. Eine strukturelle Schwachstelle besteht dort, wo der modellierte Daten- oder Kontrollfluss eines Workflows einen Regelverstoß ermöglicht. Im Unterschied zu einem reinen Entwurfsfehler, bei dem beispielsweise einige Schritte im Workflow fehlen, steht hier also der Regelbezug im Vordergrund. Welche Regeln sich aus Compliance-Regularien ergeben, wird im zweiten Kapitel behandelt. Strukturelle Schwachstellen sind diesbezüglich all jene Stellen, an denen ein Element des Workflow-Modells auf einem, mehreren oder sogar allen Pfaden durch den Workflow eine der Compliance-Regeln verletzen würde, wenn diese Stelle zur Ausführung gelänge.

1.3.4 Anwendungsfall „Modell-Prüfung vor der Ausführung“

Aller Wahrscheinlichkeit nach wird die Anzahl und Strenge der Compliance-Regularien weiter zunehmen (Müller und Terzidis, 2008). Zusätzlich zu den bereits jetzt geltenden Regeln werden Unternehmen also neue Regeln einhalten bzw. Workflows auf Regeleinhaltung prüfen müssen. Dieser Anwendungsfall lässt sich kurzgefasst als „Alte Workflows, neue Regeln“ bezeichnen und ist der für diese Arbeit gewählte. Ziel ist es, für eine möglicherweise große Anzahl bestehender Workflows effektiv und effizient zu prüfen, ob neu hinzugekommene Regeln eingehalten werden. Die Prüfungsergebnisse bilden dann die Grundlage für die Entscheidung, welche Prozesse unverändert ausgeführt werden und an welchen Prozessen Änderungen vorgenommen werden, um Compliance zu den jeweils aktuellen Regularien zu erreichen.

Einer Vorhersage von Gartner zufolge wird sich zwar die Zahl der Regulierungen mit direktem Einfluss auf die Informationstechnologie zwischen den Jahren 2006 und

2012 verdoppeln, aber weniger als ein Drittel der Unternehmen wird Compliance- und Automatisierungsbemühen vereinen (Eid und Caldwell, 2006). Um bei einer steigenden Anzahl von Regeln und automatisierten Geschäftsprozessen (laut Padovan et al. (2005) in Deutschland bereits im Jahr 2005 über 66%, Tendenz steigend) die Compliance-Prüfung dennoch wirtschaftlich durchführen zu können, werden automatisierte Verfahren benötigt, die die von Eid und Caldwell (2006) wie folgt formulierten fünf Compliance-Schritte unterstützen:

1. Know who wants you to do things
   (identify the appropriate regulations that apply to your organization)

2. Know what to do
   (interpret the regulation for the organization's environment)

3. Know what you do
   (understand and document the organization's processes and policies)

4. Do what you say
   (monitor for compliance and changes)

5. Say what you know
   (report as required)

Bereits das häufige Auftreten des Wortes „know“ weist daraufhin, dass die Automatisierung der Compliance-Schritte ein schwieriges Unterfangen ist: Wissen beinhaltet Semantik und allen Bemühungen der Forschung zu künstlicher Intelligenz (KI) zum Trotz sind Computer weiterhin in der Verarbeitung von Syntax weiter fortgeschritten als bezüglich der Semantik. Die ersten drei Schritte werden ihrer semantischen Komplexität wegen manuelle bleiben, bis die KI große Weiterentwicklungen verzeichnet. Sobald aber - ganz gleich ob wie heute manuell oder zukünftig womöglich automatisiert - maschinell auswertbare Regeln abgeleitet wurden, können die Schritte 4 und 5 automatisiert werden. Der betrachtete Anwendungsfall „Alte Workflows, neue Regeln“ entspricht Schritt 4, nämlich der Prüfung, ob bestehende Workflows neu hinzugekommene Regeln einhalten.

Vor dem Hintergrund dieses Anwendungsfalls werden im zweiten Kapitel die Anforderungen an ein automatisiertes Compliance-Prüfverfahren diskutiert. Es zeigt sich aber, dass die Anforderungen auch für den umgekehrten Fall gelten, nämlich die Prüfung von neuen Workflows auf die Einhaltung bestehender Regeln. Hier ist aus der Sicht des Workflow-Entwurfs besonders an Workflow-Editorsoftware zu denken, anhand derer Workflow-Designer neue Workflows erstellen oder bestehende Workflows abändern. Erfolgt die automatisierte Prüfung auf Regeleinhaltung genügend schnell, können die Designer bereits während der Modellierung darüber informiert werden, inwieweit neue oder geänderte Elemente des Workflow-Modells regelkonform sind. Damit lassen sich Kosten für ansonsten später nötige Workflow-Änderungen einsparen.

Auch Schritt 5, der spätere Nachweis der Regeleinhaltung, kann automatisiert werden sofern die Regeln entsprechend formalisiert wurden. In dieser Arbeit liegt jedoch, dem beschriebenen Anwendungsfall entsprechend, der Fokus auf der Prüfung *vor* der Ausführung.

Für den beschriebenen Anwendungsfall „Alte Workflows, neue Regeln“ erscheint von den in Abschnitt 1.3.2 vorgestellten Prüfungszeitpunkten und -ebenen die Entwurfszeit-Prüfung auf Modellebene wie oben diskutiert am besten geeignet. Da die bisherige IT-Unterstützung keine Lösungen für eine solche Compliance-Prüfung bietet, und bestehende (Forschungs-) Ansätze wie im zweiten Kapitel gezeigt bisher nicht den für Compliance-Prüfungen nötigen Regelumfang erreichen, besteht ein Bedarf an neuen Verfahren. Diese müssen strukturelle Schwachstellen in Workflow-Modellen aufdecken können, und zwar *sowohl* im Daten- *als auch* im Kontrollfluss.

Bei der folgenden Diskussion von Compliance-Regeln und ihrer Klassifikation gilt die bereits von (Sackmann und Kähmer, 2008) beschriebene Begrenzung, dass sich nicht alle Teile der Compliance-Regularien (siehe Abschnitt 2.1) in automatisiert prüfbare Regeln übertragen lassen. Die auf sogenannte *Safety*-Eigenschaften abbildbaren Teile lassen sich beispielsweise mit Zugriffskontrollregeln erfassen. Safety bedeutet dabei, dass keine verbotenen Aktivitäten ausgeführt werden. Die sogenannten *Liveness*-Eigenschaften sind durch die klassische Zugriffskontrolle nicht erfassbar und können im allgemeinen Fall zur Entwurfszeit nicht geprüft werden; eine näherungsweise Prüfung kann durch Kontrollaktivitäten erreicht werden (für Zeitlimits in Abschnitt 3.3.7 dargestellt). Liveness-Eigenschaften beschreiben, dass geforderte Aktivitäten stattfinden müssen, allerdings zu einem möglicherweise weit in der Zukunft liegenden Zeitpunkt. Um ein Beispiel zu nennen: die Regel, dass Daten innerhalb von zwei Jahren nach Gebrauch gelöscht werden müssen, kann letztendlich erst nach zwei Jahren ausgewertet werden. Nach einem Jahr und 364 Tagen besteht noch immer die Möglichkeit, dass die Löschaktivität fristgerecht stattfindet. Usage Control (Park und Sandhu, 2004; Pretschner et al., 2006; Katt et al., 2009) bzw. Nutzungskontrolle umfasst neben den Safety-Eigenschaften der Zugriffskontrolle auch die nach einem Zugriff relevanten Liveness-Eigenschaften. Was nicht durch Regeln der Nutzungskontrolle erfassbar ist, kann durch die in dieser Arbeit beschriebene Informationsflussanalyse nicht zur automatisierten Prüfung verwendet werden.

1.4 Problemstellung und Lösungsansatz

Compliance-Regularien beinhalten eine Vielzahl von Vorschriften, die teils rechtlicher, teils organisatorischer und teils technischer Natur sind (Sackmann et al., 2008). Viele dieser Vorschriften lassen sich als Regeln formulieren, die für eine automatisierte Prüfung verwendet werden können.

Workflows als eine Abfolge von Aktivitäten, die einen Geschäftsprozess fallbezogen automatisiert umsetzen, stellen typischerweise die höchste Ebene dar, auf der sich Regeln automatisiert prüfen lassen. Wie im Abschnitt 1.3.2 gezeigt wird, können theore-

tisch auch darunter liegende Ebenen wie z.B. die zur Workflow-Ausführung verwendete Software geprüft werden. Hier gilt aber der Umstand, dass sich je mehr Kosten einsparen lassen, umso früher Probleme erkannt und behoben werden. Deswegen soll die Prüfung frühestmöglich im Sinne der Prüfungsebenen erfolgen und auch frühestmöglich im Sinne der Prüfungszeitpunkte, hier also vor der Ausführung der Workflows bzw. zur Entwurfszeit.

Das Problem ist, dass bisherige Prüfverfahren den Datenfluss und den Kontrollfluss eines Workflows nicht in dem zur Compliance-Prüfung nötigen Umfang analysieren (siehe zweites Kapitel, Abschnitt 2.4 zu bisherigen Verfahren und Abschnitt 2.5 zum nötigen Umfang).

Die resultierende Aufgabe besteht folglich darin, einen Workflow automatisiert auf die Einhaltung von Regeln zu prüfen und dabei den für Compliance-Prüfungen nötigen Umfang der Datenfluss- *und* Kontrollfluss-Analyse zu erreichen.

Für die Analyse von Workflows haben sich Petrinetze bewährt (Peterson, 1977). Sie bieten eine graphisch leicht verständliche Notation und gleichzeitig eine formale Grundlage für die Analyse (Adam et al., 1998). Für die Darstellung von Regeln bzw. sogenannten Policies gibt es eine Vielzahl von Möglichkeiten wie XACML, WS-Policy, Ponder und ExPDT (OASIS, 2005; W3C, 2007; Sloman, 2000; Sackmann und Kähmer, 2008). Um eine generell einsetzbare Lösung zu bieten, verwendet die vorliegende Arbeit Petrinetze nicht nur für die Workflow-Notation, sondern auch für die Regel-Notation. Dazu werden Vorlagen definiert, die eine Übertragung der Regeln aus verschiedenen Ausgangsnotationen in die Petrinetz-Notation erlauben.

Der verfolgte Lösungsansatz besteht darin, Workflows und Regeln in Petrinetze zu überführen und dann anhand einer vorlagenbasierten Analyse der Petrinetze eine automatisierte Compliance-Prüfung durchzuführen.

1.5 Aufbau der Arbeit

Das erste Kapitel enthält eine wirtschaftliche Gegenüberstellung der manuellen und der automatisierten Prüfung und steckt bezüglich der Prüfungszeitpunkte und -ebenen den Rahmen für das zu entwickelnde automatisierte Prüfverfahren. Innerhalb dieses Rahmens steht auch die Definition für strukturelle **Schwachstellen in Workflows**, die zu finden Aufgabe des Verfahrens sein wird. Die Beschreibung des Problems und des Lösungsansatzes beenden das erste Kapitel.

Die Anforderungen an die Prüfung werden im zweiten Kapitel aufgestellt. Dazu werden zunächst ausgewählte Compliance-Regularien vorgestellt, um die Frage zu klären, welche Regeln für Compliance typischerweise erfüllt werden müssen. Um eine allgemeine, nicht auf einzelne Regularien beschränkte Einordnung zu erzielen, wird auf bestehende Compliance- bzw. Regelklassifikationen zurückgegriffen. Es folgt eine kurze Einführung in die Modellierung von Workflows. Zur vorliegenden Arbeit verwandte Arbeiten, also Prüfverfahren für Workflows und ähnliche Ansätze, werden ebenso wie die Prüfungsanforderungen beschrieben. Das Kapitel abschließend werden die bestehenden

Verfahren bezüglich ihrer Eignung für die automatisierte Compliance-Prüfung bewertet. Es zeigt sich, dass einige der gewünschten Analyseziele bereits von existierenden Verfahren erreicht werden. Die Prüfung von Daten- *und* Kontrollfluss wird bisher aber nicht in dem aus Compliance-Sicht nötigen Ausmaß geboten.

Für diese Prüfung wird im dritten Kapitel REVEAL vorgestellt, ein neues Prüfverfahren. REVEAL steht für *p**R**oc**E**ss **V**uln**E**rability **A**na**L**ysis*. Wichtiger Bestandteil des Verfahrens sind die Regelvorlagen, anhand derer Compliance-Regeln formalisiert und dann automatisiert geprüft werden können.

Zur Evaluation des neu entwickelten Verfahrens dient ein Fallbeispiel aus dem Gesundheitssektor. Das vierte Kapitel zeigt, wie REVEAL einen Workflow auf die Einhaltung der durch die im dritten Kapitel eingeführten Regel-Vorlagen abgedeckten Regeln überprüft und erfolgreich alle enthaltenen strukturellen Schwachstellen meldet.

Zusammengefasst wird die Arbeit im fünften Kapitel. Dabei werden auch weitere Einsatzmöglichkeiten des in dieser Arbeit entwickelten Prüfverfahrens aufgezeigt, die über die reine Suche nach möglichen Compliance-Verstößen hinausgehen.

2 Regeleinhaltung in Workflows

Nach einer kurzen Vorstellung ausgewählter Regelquellen wird die Modellierung von Regeln und Workflows behandelt. An einem Fallbeispiel werden dann die Ansatzpunkte für die Analyse gezeigt. Schließlich wird die beim Vergleich bestehender Verfahren sichtbar werdende Prüfungslücke diskutiert.

2.1 Regeln

2.1.1 Compliance-Regularien

Unternehmen sehen sich einer Vielzahl von Regeln gegenüber, so etwa aus internen Vorgaben, aus Verträgen mit anderen Unternehmen, aus sektorspezifischen oder allgemeinen „best practices“ sowie aus gesetzlichen Vorgaben. Stellvertretend für diese Arten von Regelquellen werden nachfolgend einige Regularien exemplarisch vorgestellt.

Die Finanzskandale der Firmen Enron und Worldcom waren nicht nur Auslöser für den Sarbanes-Oxley Act (SOX), sondern insgesamt für eine Welle von neuen oder verschärften Regularien. Nach der in Abbildung 2.1 gezeigten Entwicklung des SOX werden bis heute fortlaufend neue Regelwerke aufgestellt und bestehende erweitert, um ähnliche Vorfälle zu verhindern.

Die Einhaltung der Regeln wird je nach Regelquelle unternehmensintern, von Partnerunternehmen oder vom Gesetzgeber gefordert. Darüberhinaus legen Sicherheitsrahmenwerke wie ISO 17799 (International Organization for Standardization, 2005) fest, welche Standards und Verfahren anzuwenden sind. Für eine entsprechende Zertifizierung ist es unabdingbar, die für das jeweilige Unternehmen geltenden Regeln einzuhalten.

Compliance-Regularien als Regelquellen existieren etwa in eher organisatorischer Art wie SOX (107th Congress of the USA, 2002), oder eher technischer Art wie der Payment Card Industry Data Security Standard (PCI DSS), der Datensicherheitsstandard der Bezahlkartenindustrie (Payment Card Industry, 2009). Wie nachfolgend gezeigt wird, ist der Inhalt der Regularien unterschiedlich abstrakt formuliert. So sind die Regeln der OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data (OECD, 1980) eher allgemeiner Natur und fordern etwa eine Beschränkung von Datensammlungen. Hingegen enthält der Health Information Portability and Accountability Act (HIPAA) (104th Congress of the USA, 1996) viele konkrete Regeln für Gesundheitsdaten und schreibt beispielsweise vor, dass bei anonymisierten Patientendaten die Postleitzahl gekürzt werden muss, wenn in diesem Postleitzahlenbereich

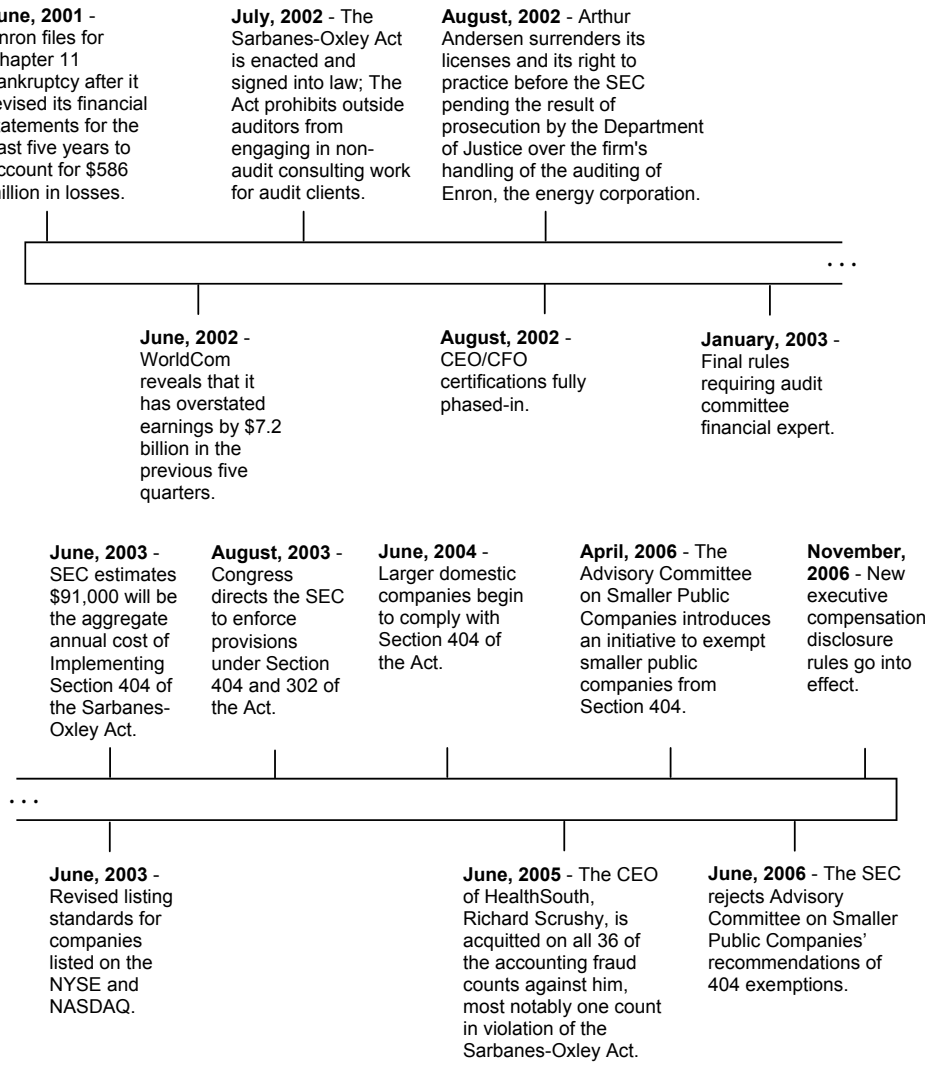


Abbildung 2.1: Die Entwicklung des Sarbanes-Oxley-Acts (aus Hartman (2007))

weniger als 20.000 Personen leben.

Die Vorstellung der Regularien beginnt mit SOX, dem wohl am meisten zitierten Regularium. Nicht wie SOX allgemein für börsennotierte Unternehmen, sondern für den Finanzmarkt aufgestellt, reguliert der anschließend vorgestellte Gramm-Leach-Bliley-Act (GLB) unter Anderem den Umgang mit nicht-öffentlichen, persönlichen Daten. Deutlich stärker fokussieren die OECD-Guidelines auf persönliche Daten, wie auch das ebenfalls vorgestellte Bundesdatenschutzgesetz (BDSG). Abschließend und am ausführlichsten wird das HIPAA-Regularium vorgestellt, weil es nachfolgend immer wieder für Beispiele herangezogen wird.

Sarbanes-Oxley-Act (SOX)

Der im Jahr 2002 in Kraft getretene SOX (107th Congress of the USA, 2002) soll Bilanzskandale verhindern, wie sie bei Enron oder Worldcom aufgetreten sind. Unternehmen müssen dazu in regelmäßigen Berichten ihre Finanzdaten offenlegen, die einem speziellen Audit unterzogen werden. Außerdem müssen interne Kontrollen eingerichtet werden, über deren Erfolg ebenfalls berichtet werden muss.

Das erste Kapitel, *Public Company Accounting Oversight Board*, legt die Aufgaben einer einzurichtenden Aufsichtsbehörde fest. Im zweiten Kapitel, *Auditor Independence*, werden Tätigkeiten aufgeführt, die Auditoren nicht durchführen dürfen, da sie unabhängig und unvoreingenommen arbeiten sollen. Das dritte Kapitel, *Corporate Responsibility*, erlegt Unternehmen die Verantwortung für die einwandfreie Erstellung von Finanzberichten auf. Eine erweiterte Offenlegung der Finanzen wird im vierten Kapitel, *Enhanced Financial Disclosures*, gefordert. Interessenskonflikte bei Auditoren bzw. sogenannten Analysten sollen durch die Vorgaben im fünften Kapitel, *Analyst Conflicts of Interest*, ausgeschlossen werden. Das sechste Kapitel, *Commission Resources and Authority*, beschreibt die für die Überwachung der SOX-Einhaltung zur Verfügung stehenden Mittel. Eine Reihe von zu erstellenden Berichten wird im siebten Kapitel, *Studies and Reports*, beschrieben. In den letzten vier Kapiteln schließlich werden einige Betrugsarten samt ihrer zugehörigen Strafmaße beschrieben: *Corporate and Criminal Fraud Accountability*, *White-Collar Crime Penalty Enhancements*, *Corporate Tax Returns* und *Corporate Fraud and Accountability*.

Workflow-Bezug: Die SOX-Forderungen enthalten keine direkt auf Workflows anwendbaren Regeln. Sie müssen erst durch Interpretation auf Workflow-Ebene anwendbar gemacht werden. Nur an einer Stelle, nämlich im dritten Kapitel, Abschnitt a.5.A, gibt es annähernd technische Forderungen. Und zwar müssen Finanzberichte eine Bestätigung enthalten, dass nicht nur interne Kontrollen eingerichtet wurden, sondern dass auch alle Mängel im Entwurf oder im Betrieb solcher Kontrollen, die die Aufzeichnung, Bearbeitung, Zusammenfassung und Berichterstattung über Finanzdaten beeinträchtigen könnten, an die Auditoren gemeldet wurden: „[require that] (5) the signing officers have disclosed to the issuer's auditors and the audit committee of the board of directors (or persons fulfilling the equivalent function) (A) all significant deficiencies in the design or operation of internal controls which could adversely affect the issuer's ability to record, process, summarize, and report financial data and have identified for the issuer's auditors any material weaknesses in internal controls".

Gramm-Leach-Bliley Act (GLB)

Der GLB (106th Congress of the USA, 1999) aus dem Jahr 1999 erlaubt bestimmte, vorher verbotene Zusammenschlüsse von Unternehmen im Finanzsektor und erleichtert damit die Zusammenarbeit von Banken, Versicherungen und dem Wertpapierhandel.

Im ersten Kapitel, *Facilitating Affiliation among Banks, Securities Firms, and Insurance Companies*, werden einige Regelungen des Glass-Steagall-Acts aufgehoben, der

zuvor die strikte Trennung der betroffenen Einrichtungen vorgeschrieben hatte. Nach weiteren Vorgaben für Zusammenschlüsse von Einrichtungen folgen Regeln für die Finanzaufsicht. Das zweite Kapitel, *Functional Regulation*, beschreibt die zulässigen Aktivitäten von Börsenmakler und Wertpapierhändlern sowie Kapitalanlagegesellschaften. Selbiges erfolgt im dritten Kapitel, *Insurance*, für Versicherungen und Versicherungsvertreter. Das vierte Kapitel, *Unitary Savings and Loan Holding Companies*, verbietet den Zusammenschluß von Spar- und Darlehenskassen mit kommerziellen Partnern. Im fünften Kapitel, *Privacy*, wird der Umgang mit sogenannter „nonpublic personal information“ (NPI) und mit Finanzdaten geregelt. Für Bausparkassen werden im sechsten Kapitel, *Federal Home Loan Bank System Modernization*, Modernisierungsmaßnahmen vorgeschrieben. Das letzte Kapitel, *Other Provisions*, behandelt Geldautomatengebühren, Wirtschaftsauskunfteien und einige thematisch gemischte, kleinere Änderungen der Regeln für die Finanzwirtschaft.

Workflow-Bezug: Für Workflows und aus Sicherheits- und Privatsphärensicht interessant ist vor allem das fünfte Kapitel *Privacy*. Darin wird der Schutz von NPI sowie die Offenlegung der jeweiligen Institutsregeln zur Erhebung und Weitergabe dieser Daten vorgeschrieben. Darüber hinaus wird der Versuch, durch falsche Angaben persönliche bzw. finanzielle Daten zu erfahren, als Straftat eingestuft (*social engineering*-Angriffe, in dieser Art auch als *pretexting* bekannt).

Die Forderungen sind so abstrakt definiert, dass sie erst nach Interpretation als Regeln für Workflows formuliert werden können, wie das folgende Beispiel zeigt.

Beispiel. Paragraph 501 *Protection of Nonpublic Personal Information* aus dem Kapitel zu Privacy lautet:

> a Privacy Obligation Policy. - It is the policy of the Congress that each financial institution has an affirmative and continuing obligation to respect the privacy of its customers and to protect the security and confidentiality of those customers' nonpublic personal information.
>
> b Financial Institutions Safeguards. - In furtherance of the policy in subsection (a), each agency or authority described in section 505(a) shall establish appropriate standards for the financial institutions subject to their jurisdiction relating to administrative, technical, and physical safeguards -
>
> 1. (1) to insure the security and confidentiality of customer records and information;
> 2. (2) to protect against any anticipated threats or hazards to the security or integrity of such records; and
> 3. (3) to protect against unauthorized access to or use of such records or information which could result in substantial harm or inconvenience to any customer.

Neben der üblichen Zugriffskontrolle lässt sich daraus auch die Forderung nach einer Zweckbindung ableiten. Dafür muss außer den typischerweise in Zugriffskontrolllisten (access control lists, ACL) mitgeführten Berechtigungen der zugreifenden Angestellten auch der Zweck eines Datensatzes bzw. einer Aktivität maschinell auswertbar sein. ■

OECD-Guidelines

Die 1980 von der OECD herausgegebenen Guidelines on the Protection of Privacy and Transborder Flows of Personal Data (OECD, 1980) enthalten acht Prinzipien, die dank ihrer Kürze nachfolgend im Original wiedergegeben werden können.

1. **Collection Limitation Principle**: There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.

2. **Data Quality Principle**: Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.

3. **Purpose Specification Principle**: The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.

4. **Use Limitation Principle**: Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with Paragraph 9 except: a) with the consent of the data subject; or b) by the authority of law.

5. **Security Safeguards Principle**: Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data.

6. **Openness Principle**: There should be a general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller.

7. **Individual Participation Principle**: An individual should have the right: a) to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him; b) to have communicated to him, data relating to him within a reasonable time; at a charge, if any, that is not excessive;

in a reasonable manner; and in a form that is readily intelligible to him; c) to be given reasons if a request made under subparagraphs (a) and (b) is denied, and to be able to challenge such denial; and d) to challenge data relating to him and, if the challenge is successful to have the data erased, rectified, completed or amended.

8. **Accountability Principle**: A data controller should be accountable for complying with measures which give effect to the principles stated above.

Workflow-Bezug: Aus den OECD-Guidelines ergeben sich Datenschutzregeln, wie sie vielfach auch in anderen Regularien zu finden sind. Es handelt sich um die Zweckbindung, das Erforderlichkeitsprinzip, den Einwilligungsbedarf und die Nachfrage- und Änderungsberechtigung für Betroffene. Spezielle Regeln zur Umsetzung werden - der Richtliniennatur gemäß - nicht vorgegeben.

Bundesdatenschutzgesetz (BDSG)

Das 1990 in Kraft getretene und zuletzt im Jahr 2009 geänderte BDSG (Bundesministerium für Justiz (BMJ), 2009) enthält Vorschriften für die Erhebung, Verarbeitung und Nutzung personenbezogener Daten (PBD), darunter auch sogenannter „besonderer Arten personenbezogener Daten“ wie ethnische Herkunft und religiöse Überzeugungen. In der novellierten Fassung setzt das BDSG die Richtlinie 95/46/EG Europäisches Parlament (1995) des Europäischen Parlaments um, die zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten dient.

Der erste Abschnitt *Allgemeine und gemeinsame Bestimmungen* klärt zunächst die verwendeten Begriffe. In §3a wird dann allgemein Datenvermeidung und Datensparsamkeit gefordert. Ebenso wird in §4 allgemein jede nicht explizit erlaubte Verwendung von Daten verboten. §4a verlangt die Einholung der Einwilligung des Betroffenen, und vor der Übermittlung von PBD ins Ausland oder an über- und zwischenstaatliche Stellen wird in §4b eine Prüfung des Schutzniveaus der Gegenseite verlangt. Eine automatisierte Einzelentscheidung darf laut §6a nur dann erfolgen, wenn der Betroffene die Möglichkeit des Widerspruchs hat.

§9 fordert Zutrittskontrolle, Zugangskontrolle, Zugriffskontrolle, Weitergabekontrolle, Eingabekontrolle, Auftragskontrolle und Verfügbarkeitskontrolle, sowie darüber hinaus auch die getrennte Verarbeitung je nach Zweck der erhobenen Daten.

Der zweite Abschnitt *Datenverarbeitung der öffentlichen Stellen* ergänzt die allgemeinen Bestimmungen des ersten Abschnitts um die Erlaubnis, PBD erheben zu dürfen, wenn dies aus übergeordneten Gründen wie zur Gefahrenabwehr, zur Gesundheitsvorsorge, für die wissenschaftliche Forschung oder aus über- oder zwischenstaatlichen Verpflichtungen heraus nötig ist. Die Speicherung, Veränderung und Nutzung von PDB wird in diesem Sinne auch zur Strafverfolgung erlaubt. Der Rest des Abschnitts beschreibt die Rechte des Betroffenen und organisatorische Vorgaben für öffentliche Stellen.

Der dritte Abschnitt *Datenverarbeitung nicht-öffentlicher Stellen und öffentlich-rechtlicher Wettbewerbsunternehmen* erlaubt zusätzlich zu den Bestimmungen des ersten Abschnitts die Verarbeitung zusammengefasster (z.B. in Listenform verkürzter) PBD für Adresshandel oder Werbung, wenn die Verarbeitung für bestimmte Zwecke erfolgt und die Daten auf eine Branchenbezeichnung, den Namen, Titel, akademischen Grad, die Anschrift und das Geburtsjahr beschränkt werden (§28). Außer den Rechten des Betroffenen beschreibt der Abschnitt weiterhin die Aufgaben der Aufsichtsbehörde für die nicht-öffentlichen Stellen und öffentlich-rechtlichen Wettbewerbsunternehmen.

Der vierte Abschnitt *Sondervorschriften* behandelt eine besondere Zweckbindung bei PBD unter Berufs- oder Amtsgeheimnis in §39, Vorgaben für Forschungseinrichtungen (PBD sind möglichst zu anonymisieren) in §40 sowie Zusatzangaben für die Datenverarbeitung und Nutzung von PBD bei der Deutschen Welle in §41 und §42. Gelangt eine nicht-öffentliche Stelle unrechtmäßig in Kenntnis von PBD, so muss sie laut §42a die Aufsichtsbehörde und den Betroffenen darüber informieren.

Der fünfte Abschnitt *Schlussvorschriften* enthält in §43 Bußgeld-, und in §44 Strafvorschriften, der sechste Abschnitt *Übergangsvorschriften* regelt den Übergang vom vorherigen Stand zu den Bestimmungen des geänderten Gesetzes.

Workflow-Bezug: Besonders aus §9 ergeben sich Vorgaben für Workflows. Zutritts-, Zugangs-, Weitergabe-, Eingabe- und Verfügbarkeitskontrollen lassen sich datenunabhängig für alle Workflows umsetzen. Für die Zugriffskontrolle müssen hingegen für jeden Datensatz die zugehörigen Berechtigungen abrufbar sein, um festzustellen, ob ein bestimmter Benutzer (bzw. eine bestimmte Rolle) auf den Datensatz zugreifen darf. Für die Auftragskontrolle, die sicherstellen soll, dass PBD nur gemäß den Weisungen des Auftraggebers verarbeitet werden, müssen ebendiese Weisungen abrufbar sein. Für die zweckgetrennte Verarbeitung schließlich muss nicht eine Berechtigung oder eine Weisung, sondern der Zweck eines Datensatzes feststellbar sein.

Health Information Portability and Accountability Act (HIPAA)

Mit dem amerikanischen HIPAA (104th Congress of the USA, 1996) von 1996 soll einerseits erreicht werden, dass bei einem Arbeitsplatzwechsel oder -verlust zumindest Teile der Arbeitnehmerkrankenversicherung bestehen bleiben. Gleichzeitig soll die Sicherheit der Gesundheitsdaten und die Privatsphäre der Patienten besser geschützt werden, und Datenübertragungsstandards sollen Mehraufwand bei der Verwaltung vermeiden. Im Jahr 2009 wurden durch den Health Information Technology for Economic and Clinical Health (HITECH) Act einige der HIPAA-Regeln verschärft, ohne aber grundlegend neue Regeln einzuführen.

HIPAA besteht aus fünf Teilen, im englischen Original als „Titles“ bezeichnet.

Der erste Teil, *Title I: Health Care Access, Portability, and Renewability*, beschreibt rechtliche und organisatorische Änderungen bei der Beantragung, Übertragung und Erneuerung von Krankenversicherungen. Der zweite Teil wird nachfolgend im Detail beschrieben. Der dritte Teil, *Title III: Tax-related Health Provisions*, erlaubt die steuerliche Absetzung gewisser Gesundheitskosten. Im vierten Teil, *Title IV: Application and*

Enforcement of Group Health Plan Requirements, werden Bedingungen für die Krankenversicherung bei Vorerkrankungen und für die Fortsetzung von Krankenversicherungen festgelegt. Der fünfte Teil, *Title V: Revenue Offsets*, regelt unternehmenseigene Lebensversicherungen und die Behandlung von Nicht-U.S.-Bürgern.

Der zweite Teil, *Title II: Preventing Health Care Fraud and Abuse; Administrative Simplification; Medical Liability Reform*, definiert einerseits Standards für Transaktionen und Code Sets sowie einheitliche, eindeutige Identifikatoren. Andererseits legt er Regeln für den Umgang mit schützenswerten Gesundheitsdaten, sogenannter **protected health information (PHI)** fest. Dementsprechend besteht der zweite Teil hauptsächlich aus fünf als „Rule“ bezeichneten Regelsätzen, und zwar *Transactions and Code Sets Rule, Unique Identifiers Rule, Security Rule, Privacy Rule* und *Enforcement Rule*.

Der letztgenannte Regelsatz steckt den Rahmen nicht etwa für das informatische Enforcement, sondern die rechtliche Durchsetzung. Neben Strafmaßen wird dort beispielsweise die Durchführung von Anhörungen bei Verstößen beschrieben. Die Regelsätze zu Transaktionen, Code Sets und eindeutigen Identifikatoren dienen allein administrativen Verbesserungen. Im Folgenden liegt der Fokus daher auf den *Security*- und vor allem auf den *Privacy*-Regelsätzen.

Der *Security*-Regelsatz *Security Standards for the Protection of Electronic Protected Health Information* (Subpart C, §164.302 bis §164.318) behandelt sechs Themengebiete (von den Begriffsklärungen und Angaben zur Gültigkeit der Paragraphen absehend).

- **Security standards general rules**: Verlangt die Einhaltung der Vertraulichkeit, Integrität und Verfügbarkeit beim Erstellen, Empfangen, Aufbewahren und Übertragen von PHI und dass Schutzmaßnahmen gegen vorhersehbare Bedrohungen dieser Daten getroffen werden.

- **Administrative safeguards**: Verlangt die Einrichtung eines Sicherheitsmanagementprozesses, der Risikoanalyse und Risikomanagement beinhaltet. Desweiteren wird ein Zugriffsmanagement gefordert sowie Sicherheitstraining, Incidentmanagement, Backup-Pläne, Notfallpläne und regelmäßige Tests der Maßnahmen.

- **Physical safeguards**: Verlangt Zugangskontrolle zu Räumen und Geräten, die PHI enthalten.

- **Technical safeguards**: Verlangt Zugriffskontrolle inklusive eindeutiger Benutzerkennungen und einem Notfallzugriffsplan. Logging und Audit wird gefordert, und die Verwendung von Verschlüsselung zur sicheren Übertragung von PHI wird empfohlen.

- **Organizational requirements**: Verlangt die Einhaltung der HIPAA-Regeln auch von an der PHI-Verarbeitung beteiligten Dritten und dass die Einhaltung geprüft werden muss.

- **Policies and procedures and documentation requirements**: Verlangt die Aufstellung und Durchführung von Plänen zur Einhaltung der HIPAA-Regeln und formuliert eine Dokumentationspflicht für beides.

Der *Privacy*-Regelsatz *Privacy of Individually Identifiable Health Information* (Subpart E, §164.500 bis §164.534) umfasst elf Themengebiete (ebenfalls von den Begriffsklärungen und Angaben zur Gültigkeit der Paragraphen absehend).

- **Uses and disclosures of protected health information general rules**: Formuliert ein grundlegendes *default deny* (Zugriff nur mit expliziter Erlaubnis) und *minimum necessary* (Zugriff nur auf benötigte Daten, auch als *need to know*-Prinzip bekannt). Außerdem wird die Verwendung ano- oder pseudonymisierter Daten (*de-identified PHI*) erlaubt und die Datenweitergabe an Dritte unter gewissen Umständen.
- **Uses and disclosures organizational requirements**: Enthält Bedingungen für Verträge mit Dritten und für Krankenversicherungen.
- **Uses and disclosures to carry out treatment, payment, or health care operations**: Erteilt die Erlaubnis, PHI zu Zwecken der Behandlung, Abrechnung und zum Betrieb der Gesundheitseinrichtung (beispielsweise Patientenverzeichnis) zu verwenden.
- **Uses and disclosures for which an authorization is required**: Legt fest, dass immer eine Berechtigung zur Verwendung von PHI vorliegen muss, sofern die Verwendung nicht explizit durch eine andere HIPAA-Regel erlaubt oder gefordert wird. Darüberhinaus wird die Verwendung und Weitergabe von psychotherapeutischen Aufzeichnungen eingeschränkt, ebenso die von PHI zu Zwecken des Marketings.
- **Uses and disclosures requiring an opportunity for the individual to agree or to object**: Erlaubt die Verwendung und Weitergabe von PHI, wenn die Betroffenen informiert wurden und Gelegenheit hatten, ihre Zustimmung oder ihren Widerspruch auszudrücken.
- **Uses and disclosures for which an authorization or opportunity to agree or object is not required**: Definiert Umstände, unter denen keine Berechtigung zur Verwendung oder Weitergabe von PHI nötig ist. Übergeordnete Ansprüche entstehen dabei aus Gesetzen, der öffentlichen Gesundheit, Fällen von Gewalt oder Missbrauch, der Gesundheitsaufsicht, bei Straftaten, Organspenden, Forschungsprojekten, Militärangehörigen und Fragen der inneren Sicherheit.
- **Other requirements relating to uses & disclosures of protected health information**: Beschreibt im Detail, welche PHI verwendet oder weitergegeben werden dürfen, je nachdem, ob sie anonymisiert wurden oder nicht.

- **Notice of privacy protection for protected health information**: Schreibt vor, dass Gesundheitseinrichtungen ihre Besucher bzw. Patienten über die Verwendung und Weitergabe von PHI vorab informieren müssen, und zwar in Form einer Bekanntmachung wie z.B. durch einen Aushang.

- **Rights to request privacy protection for protected health information**: Zwingt Gesundheitseinrichtungen, ihren Patienten die Möglichkeit zu geben, eine Beschränkung der Verwendung und Weitergabe von PHI zu beantragen. Das Vorgehen im Falle der Antragszustimmung oder -ablehnung wird ebenfalls festgelegt.

- **Access of individuals to protected health information**: Erlaubt Patienten eine Abfrage der über sie gespeicherten PHI und bestimmt das Vorgehen bei der Abfrage, der Gewährung des Zugriffs und der Verweigerung des Zugriffs auf gewisse PHI.

- **Amendment of protected health information**: Gibt Patienten das Recht, die über sie gespeicherten PHI bei Bedarf anpassen zu lassen. Gesundheitseinrichtungen werden verpflichtet, nicht nur diese Anpassung durchzuführen, sondern auch möglichen Dritten, die die PHI vorher erhalten haben, die Anpassungen weiterzugeben.

- **Accounting of disclosures of protected health information**: Verlangt, dass Gesundheitseinrichtungen auf Patientenanfrage hin eine Auflistung der Fälle von PHI-Weitergaben zur Verfügung stellen, und beschreibt detailliert den Inhalt der zu erstellenden Auflistung.

- **Administrative requirements**: Beschreibt einige Anforderungen an das Personal, die Sicherung von PHI, Beschwerden und deren Dokumentation sowie die Einrichtung und Änderung von internen, HIPAA-bezogenen Verfahrensweisen.

Workflow-Bezug: Die HIPAA Privacy Rule beschreibt ganz konkret, wie mit PHI umzugehen ist. Außer Angaben im Sinne der Zugriffskontrolle (wer darf zu welchen Zwecken auf welche Daten zugreifen) wird festgelegt, welche Auskunfts- und Zugriffsrechte die Patienten haben und wie von den betroffenen Unternehmen (*covered entities*) mit Patientenanfragen umzugehen ist. Vorschriften bezüglich der Dokumentationspflicht und zur Verwaltung (z.B. Personal) bilden den dritten und kleinsten Teil der *HIPAA Privacy Rule*.

Einige Beispiele für Regeln mit direktem Workflow-Bezug sind nachfolgend aufgelistet.

- Abschnitt 164.502 a schreibt *default deny* für PHI vor. Aktivitäten, die nicht explizit erlaubt sind, dürfen demnach nicht ausgeführt werden.

- Abschnitt 164.502 b schreibt *minimum necessary* für PHI vor. Es dürfen also immer nur genau die Daten ausgetauscht werden, die für eine bestimmte Aktivität gebraucht werden.

- Abschnitt 164.506 a 1 fordert das Einholen der Patienteneinwilligung, bevor Patientendaten verarbeitet oder weitergegeben werden.

- Abschnitt 164.512 beschreibt, an wen bzw. welche Personengruppen Patientendaten herausgegeben werden dürfen.

- Abschnitt 164.524 b 2 i legt eine maximale Reaktionsfrist auf Patientenanfragen fest.

- Abschnitt 164.526 c 3 schreibt vor, dass Änderungen an Patientendaten an alle betroffenen Parteien weitergegeben werden müssen.

2.1.2 Regelsprachen zur Modellierung

Die klassische Erfassung von Zugriffskontrollregeln erfolgt über eine Zugriffskontrollmatrix. In einer solchen Matrix ist für jeden Benutzer festgelegt, welche Aktion (typischerweise Erzeugen, Lesen, Schreiben, Löschen) er auf welchem Objekt (bzw. z.B. Dateitypen) ausführen darf. Neuere Modelle, wie sie durch sogenannte Policy-Sprachen angeboten werden, sollen höhere Flexibilität bieten und besonders im Internet beiden Seiten einer Geschäftsbeziehung gerecht werden.

Unternehmen auf der einen, Partnerunternehmen oder Kunden auf der anderen Seiten sollen ihre (Datenschutz-) Regeln formulieren und idealerweise automatisiert abgleichen können. Von Kähmer (2009) wurden die Policy-Sprachen P3P, APPEL, EPAL und XACML in Bezug auf ihre Ausdrucksstärke, Flexibilität, Berechenbarkeit und Modularität verglichen. Keine der Sprachen erlaubt die mathematisch fundierte Kombination von Regeln, die aber zur automatisierten Zusammenführung von Regeln zweier Parteien nötig ist. In der Kähmer'schen Dissertation wird deswegen eine neue Sprache, ExPDT (Extended Privacy Definition Tools) vorgeschlagen, die diese Kombination von Regelsätzen erlaubt.

Für die vorliegende Arbeit ist die Wahl der Policy-Sprache zweitrangig. Die genannten Sprachen, aber auch Sprachen wie WS-Policy aus dem Bereich der Web Services, sind alle XML-basiert und lassen sich durch eine schemagestützte XML-Transformation bei Bedarf in andere XML-Dialekte übertragen. Für die Prüfung auf Einhaltung von Compliance-Regeln müssen nur die nachfolgend vorgestellten Regel-Elemente erfasst werden können. Dazu ist beispielsweise sowohl die umfangreiche ExPDT-Sprache in der Lage wie auch die erweiterbare WS-Policy-Sprache.

2.1.3 Regel-Elemente

Der Verschiedenheit der Compliance-Regularien zum Trotz sind zur Erfassung der jeweils enthaltenen Regeln nur einige wenige Elemente nötig, wie die nachfolgend vorgestellten Klassifikationen zeigen.

Sadiq, Governatori und Namiri (2007)

In ihrem Artikel „Modeling Control Objectives for Business Process Compliance“ identifizieren Sadiq et al. (2007) vier sogenannte *control tags* bzw. Bausteine, anhand derer Geschäftsprozesse auf Regeleinhaltung überprüft werden können:

- **Data control tags**: Welche Daten müssen aufbewahrt werden, woher stammen Daten.
- **Flow control tags**: Modalität von Aktivitäten und ggf. gewünschte Reihenfolge der Ausführung.
- **Resource control tags**: Wer bekommt unter welchen Umständen Zugriff auf Daten (z.B. für Vier-Augen-Prinzip).
- **Time control tags**: Zeitvorgaben wie z.B. einzuhaltende Fristen.

Im Artikel von Sadiq et al. fehlen präzise Angaben zu den Regularien, die untersucht wurden. Eingangs werden SOX, Basel II und HIPAA erwähnt, später wird noch die Erfahrung aus verschiedenen SAP-Projekten als Grundlage genannt. Der aus den Regularien zu den Bausteinen führende Ableitungsvorgang wird von Sadiq et al. nicht beschrieben.

Svirskas, Courbis, Molva und Bedžinskas (2007)

Svirskas et al. (2007) identifizieren sieben Fragen, die bezüglich der Compliance von SOA-basierten Interaktionen zu beantworten sind:

1. Have the necessary actions been performed or not at all? Is the whole result of collaboration valid or not?
2. Have the actions been performed in the right order and accordingly to the regulations?
3. Were these persons authorized to authorize the actions?
4. Who certified the correctness and compliance of the actions?
5. At what time exactly the actions have been performed?
6. Has the evidence and/or timestamp itself not been tampered with?

7. Is the evidence ready to be used in the court of law, according to the legislation in effect?

Die Fragen 4 und 7 sind organisatorischer Natur, die Frage 6 ist auf das Logging bezogen. Aus den restlichen Fragen ergibt sich im Hinblick auf Workflow-Modelle der Bedarf nach Vorlagen für die Prüfung auf nötige Aktivitäten, die Reihenfolge der Aktivitäten, die Berechtigung für die Ausführung von Aktivitäten sowie den Zeitpunkt der Ausführung. Auf der Modellebene lässt sich der Zeitpunkt der Ausführung allerdings nicht direkt bestimmen, weil er sich erst zur Laufzeit herausstellt. Wie im Kapitel 3 diskutiert, kann diese Prüfung aber indirekt auf Modellebene dadurch erfolgen, dass das Vorhandensein von Kontrollaktivitäten geprüft wird.

Svirskas et al. nennen die Datengrundlage ihrer Untersuchung nicht eindeutig, zitieren aber unmittelbar vor der Ableitung ihrer Fragen die Regeln der OMG Regulatory Compliance Alliance (ORCA, Vorgänger von Governance, Risk Management and Compliance Roundtable, GRC-RT) und der Global Regulatory Information Database (Compliance GRID, Vorgänger von Governance, Risk Management and Compliance GRID, GRC-GRID) (Object Management Group (OMG)). Auch hier wird der Ableitungsprozess nicht beschrieben, weswegen beispielsweise unklar bleibt, welche anderen Fragen ermittelt aber als nicht relevant eingestuft wurden.

COMPAS (2008)

Für das von der EU geförderte Projekt „Compliance-driven Models, Languages, and Architectures for Services“ (COMPAS) wurde im Deliverable D2.1 „State of the art in the field of compliance languages“ (Tilbug University, 2008) eine Übersicht über Compliance-Rahmenwerke, -regeln und -sprachen erstellt. Dabei wurden fünf *basic compliance concerns* bzw. Bausteine identifiziert, die aus Compliancesicht in einem Geschäftsprozess geprüft werden müssen:

- **Control flow**: Welche Aktivitäten sind (nicht) erlaubt und/oder nötig, in welcher Reihenfolge dürfen/müssen sie ausgeführt werden.
- **Information**: Daten, die zwischen Aktivitäten (nicht) ausgetauscht werden dürfen oder müssen.
- **Locative**: Ort der Ausführung der Aktivitäten.
- **Resource**: Wer oder was ist an der Ausführung beteiligt, also ausführende Rollen im Falle von Menschen oder ausführende Dienste im Falle von Computern.
- **Temporal**: Zeitliche Vorgaben, ab/bis wann eine Aktivität auszuführen ist.

Darüber hinaus werden sieben weitere *advanced compliance concerns* genannt: *monitoring, payment, privacy, quality, retention, security* und *transaction*. Da es sich dabei um Kombinationen der fünf *basic compliance concerns* handelt, werden diese sieben

Aspekte hier nicht weiter diskutiert. Ein Prüfverfahren, welches die ersten fünf Bausteine prüfen kann, kann entsprechend auch die darauf aufbauenden *advanced compliance concerns* prüfen.

Die für die Untersuchung verwendeten Regularien sind die Vorgaben der Bankenaufsicht in Basel II, die amerikanischen HIPAA-Regeln für das Gesundheitswesen, die International Financial Reporting Standards (IFRS), die Regeln der Financial Industry Regulatory Authority (FINRA), die EU-Finanzmarktrichtlinie *Markets in Financial Instruments Directive* (MIFID), das amerikanische Finanzgesetz SOX und *Tabaksblat*, die niederländischen Regeln zur Unternehmensführung (*corporate governance*). Auf welche Art und Weise die Bausteine abgeleitet wurden haben die Autoren in ihrem Bericht nicht erklärt.

Breaux (2009)

Breaux (2009) schlägt eine domänenunabhängige Ontologie für Compliance vor. Wie in Abbildung 2.2 gezeigt, enthält die Ontologie einerseits Regelelemente (links in der Abbildung), andererseits Workflow-Elemente (rechts oben in der Abbildung).

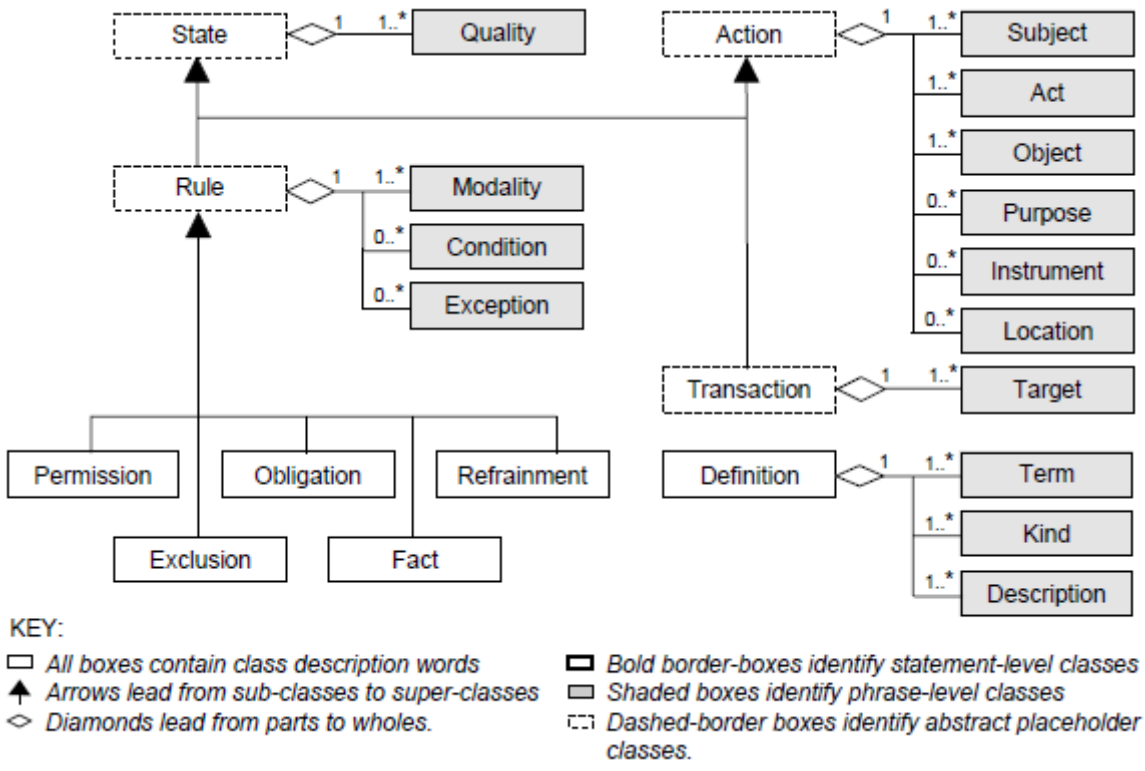


Abbildung 2.2: Compliance-Ontologie von Breaux (2009)

In der Ontologie gibt es zu einer Aktivität immer ein Subjekt, welches eine Aktion mit bzw. auf einem Objekt ausführt. Optional kann es dazu einen Zweck geben, zu

dem die Aktivität ausgeführt wird, ein Instrument, wie die Aktivität ausgeführt wird und einen Ort, an dem die Aktivität ausgeführt wird.

Zur Erstellung der Ontologie hat Breaux hauptsächlich HIPAA analysiert. Zusätzlich wurden über einhundert *Internet privacy policies* aus dem Finanz- und Gesundheitssektor betrachtet, die ETOPS-Regularien zur Flugsicherheit (U.S. 14 CFR §121.374) sowie Regularien zur Barrierefreiheit (U.S. 36 CFR §1194).

Cabanillas, Resinas und Ruiz-Cortés (2010)

Cabanillas et al. (2010) präsentieren eine Liste der vier relevantesten Elemente für die Compliance-Prüfung von Geschäftsprozessen:

- **Control flow**: Die Ausführungsreihenfolge von Aktivitäten.
- **Data**: Der Fluss von Datenobjekten durch den Prozess.
- **Resources**: Menschen die mit dem Prozess interagieren.
- **Time**: Fristen für die Ausführung von Aktivitäten.

Cabanillas et al. halten sich bei ihrer Einteilung stark an die in ihrem Artikel zitierte Klassifikation von Sadiq et al. (2007), die in Abschnitt 2.1.3 vorgestellt wird. Die einzelnen Elemente werden in selbigem Abschnitt 2.1.3 näher beleuchtet. An dieser Stelle sei nur angemerkt, dass bei den von Cabanillas et al. definierten Elementen einige der von Sadiq et al. beschriebenen Unterelemente fehlen. *Control flow* umfasst dort nämlich auch die Modalität von Aktivitäten, und *Resources* beinhaltet dort auch die Computersysteme, die manche Aktivitäten ausführen.

Eine direkte Angabe der Quellen fehlt bei der von Cabanillas et al. vorgestellten Untersuchung. Ebensowenig wird der Ableitungsprozess beschrieben.

2.2 Workflows

In diesem Abschnitt werden die Modellierungssprachen ARIS, Aristaflow, BPEL und BPMN sowie vor allem Petrinetze vorgestellt. Schließlich werden die grundlegenden Elemente eines Workflows beschrieben.

2.2.1 Standardmodelle für Workflows

ARIS

Die von Scheer vorgeschlagene „Architektur integrierter Informationssysteme“ (ARIS) findet besonders in Deutschland vielfach Anwendung. Nicht zuletzt weil sie als Grundlage für viele Produkte der SAP AG dient, wird ARIS auch weltweit eingesetzt. Aus

Workflow-Sicht ist vor allem die ARIS-Steuerungssicht interessant (Hansen und Neumann, 2005, Kap. 2.2.5, S. 233). Workflows werden für ARIS als ereignisgesteuerte Prozessketten (EPK) modelliert. Ein EPK-Modell besteht aus Ereignissen, Funktionen, Prozessschnittstellen, Konnektoren und dem Kontrollfluss (Hansen und Neumann, 2005, Kap. 2.2.5, S. 236). Durch die Hinzunahme von Angaben aus der Daten-, Organisations- und/oder Leistungssicht entstehen sogenannte erweiterte EPK (eEPK). Seit der Konferenz ARIS 2007 wird meist nicht mehr zwischen EPK und eEPK unterschieden. Neben dem Kontrollfluss können EPK demnach auch den Datenfluss eines Workflows beschreiben, wie die Abbildung 2.3 an einem Beispiel zeigt.

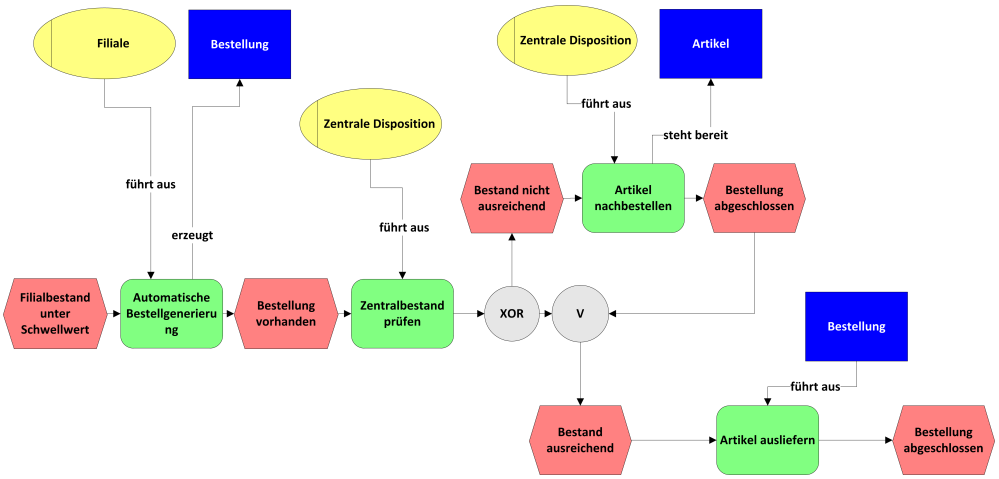


Abbildung 2.3: Beispiel einer EPK (vereinfacht aus (Hansen und Neumann, 2005, Kap. 2.2.5, S. 237))

Aristaflow (ehemals ADEPT)

Aristaflow (Dadam et al., 2009) ist eine Anwendung für das Workflow-Management, beinhaltet aber auch ein (proprietäres) Workflow-Modell. Der Fokus liegt auf flexiblen Änderungen an - möglicherweise bereits laufenden - Workflows, das heißt auf dem Einfügen, Verschieben oder Entfernen von Aktivitäten. Wie Dadam et al. (2009) schreiben, gibt es weder andere Produkte noch andere Forschungsprototypen, die eine ähnliche Workflow-Flexibilität bieten. Das zugrundeliegende Workflow-Modell (Reichert und Dadam, 1998) erfasst den Daten- und Kontrollfluss eines Workflows und wird von Aristaflow z.B. dazu genutzt, bei einer lesenden Aktivität zu prüfen, ob die zu lesenden Daten an dieser Stelle im Workflow überhaupt verfügbar sind. Die Abbildung 2.4 wurde aus (Reichert und Dadam, 1998) reproduziert und zeigt einen Workflow mit zwei Aktivitäten, dessen zweite Aktivität auf ein Datum zugreifen will, welches ihr nicht zur Verfügung steht.

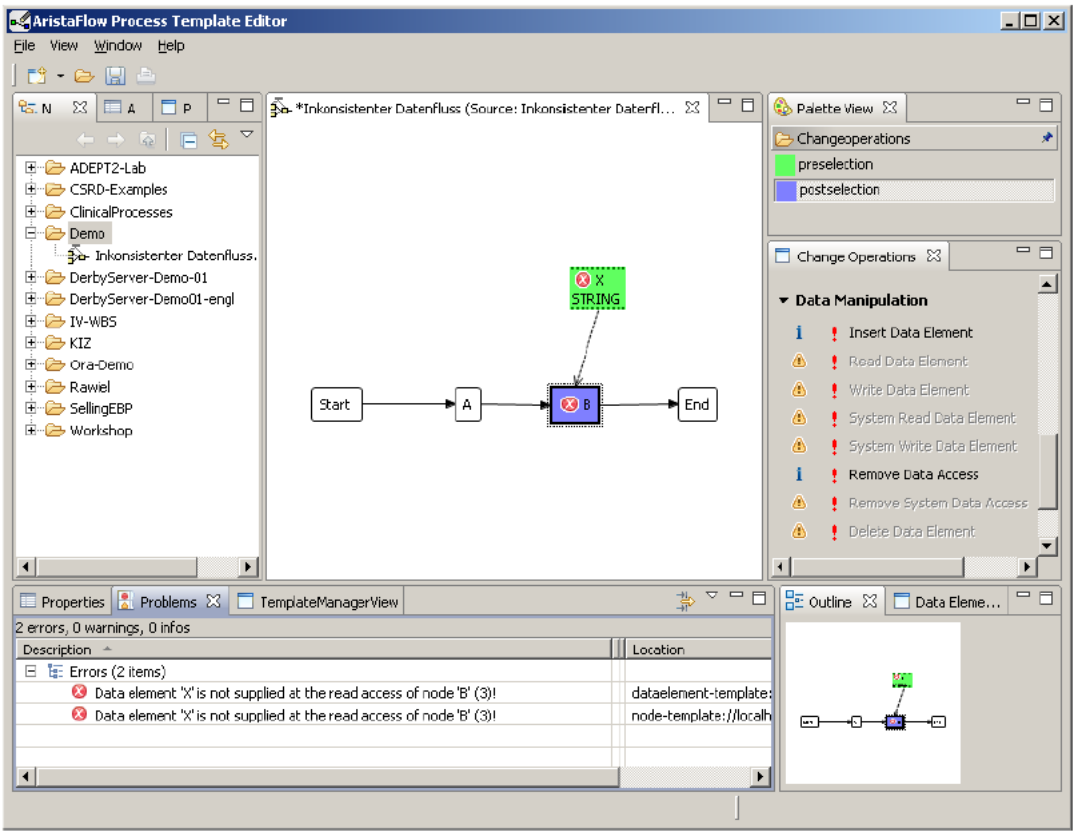


Abbildung 2.4: Workflow-Beispiel in Aristaflow (aus (Dadam et al., 2009))

Business Process Execution Language (BPEL)

Anhand von BPEL (Organization for the Advancement of Structured Information Standards (OASIS), 2007) lassen sich Geschäftsprozesse in Form einer XML-basierten Programmiersprache notieren. Neben der Wertzuweisung durch *assign* gibt es dazu Grundaktivitäten und strukturierte Aktivitäten. Die leere Aktivität *empty* dient als Platzhalter, zum Aufruf von Aktivitäten wird *invoke* verwendet, mit *receive* kann auf den Eingang einer Nachricht gewartet werden, mit *reply* wird eine Nachricht als Antwort auf einen Aufruf versendet, und durch *wait* kann eine Pause festgelegt werden, die durch einen Trigger (z.B. Ablauf einer Zeitspanne) beendet wird. Über diese Grundaktivitäten hinaus wird die Struktur durch sequentielle Ausführung (*sequence*), parallele Ausführung (*flow*), Verzweigung (*pick*, *switch*) und Schleifen (*while*) modelliert.

Die schriftliche Notation wird naturgemäß eher von Programmierern verwendet. Für den Workflow-Entwurf und Besprechungen mit verschiedenen Zielgruppen kommt eher die graphische Darstellung zum Einsatz, wie sie in Abbildung 2.5 an einem einfachen Beispiel-Workflow gezeigt wird.

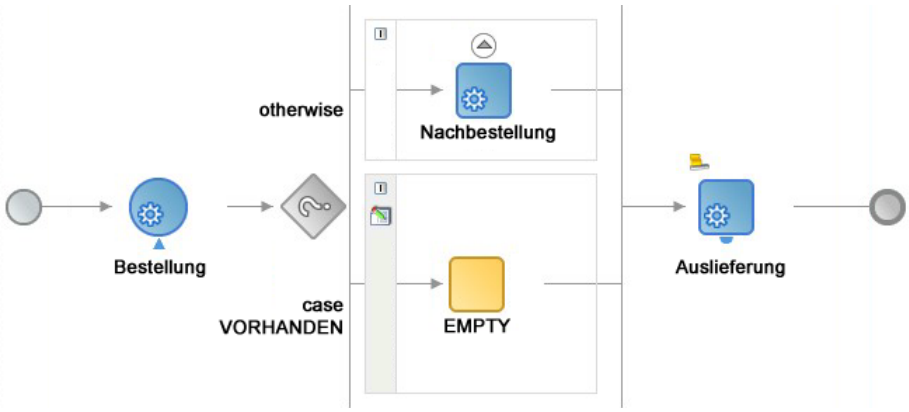


Abbildung 2.5: Beispiel eines Workflows in BPEL

Business Process Modeling Notation (BPMN)

Ganz ähnlich dem BPEL-Standard ist der BPMN-Standard, bei dem es ursprünglich weniger um die direkte Ausführbarkeit als vielmehr um die graphische Darstellung ging. Wie aus dem Vergleich von Abbildung 2.6 mit Abbildung 2.5 ersichtlich, werden BPEL und BPMN sehr ähnlich visualisiert.

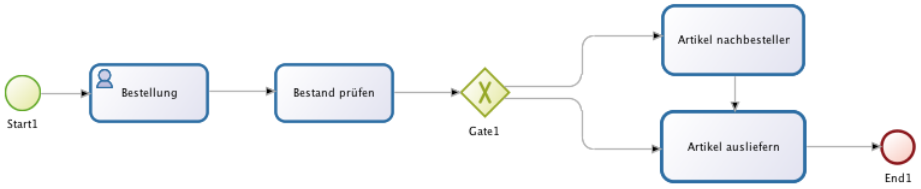


Abbildung 2.6: Beispiel eines Workflows in BPMN

Ein Großteil der BPMN-Angaben läßt sich in entsprechende BPEL-Angaben übersetzen (Ouyang et al., 2006). Im Januar 2011 wurde BPMN 2.0 veröffentlicht, womit BPMN und BPEL noch näher aneinander rücken (OMG, 2011). Einerseits ist BPMN 2.0 im Gegensatz zu BPMN 1.x ausführbar, andererseits wurde als Teil der Bemühungen, BPMN und BPEL zusammenzuführen, eine Untermenge von BPMN 2.0 so definiert, dass die Transformation in BPEL einfacher wird. Allerdings ist BPMN 2.0 aufgrund einer Vielzahl neuer Symbole unübersichtlicher als BPMN 1.x oder BPEL.

2.2.2 Modellierung mit Petrinetzen

Für die Analyse von Workflows haben sich Petrinetze bewährt (Peterson, 1977). Sie bieten eine graphisch leicht verständliche Notation sowie eine formale Grundlage für die Analyse (Adam et al., 1998) bzw. dienen als mathematisches Modellierungswerkzeug (Murata, 1989). In der einfachsten Form bestehen sie aus **Plätzen**, die ein **To-**

ken enthalten können, **Transitionen**, die Tokens verbrauchen und neu erzeugen, und **Kanten**, die Plätze und Transitionen verbinden. Transitionen können **feuern**, wenn sie **aktiv** sind, das heißt wenn in den direkt vor ihnen liegenden Plätzen Tokens enthalten sind. Dabei verbrauchen sie die Tokens dieser Plätze und erzeugen neue Tokens in den hinter ihnen liegenden Plätzen.

Plätze und Transitionen sind miteinander durch gerichtete Kanten verbunden, wobei ein Platz nur mit Transitionen und eine Transition nur mit Plätzen verbunden werden darf. Es sind also niemals zwei Plätze oder zwei Transitionen direkt durch eine Kante verbunden; Petrinetze stellen bipartite Graphen dar. Die Plätze direkt vor einer Transition sind die **Eingangsplätze** dieser Transition, die Plätze direkt hinter einer Transition sind deren **Ausgangsplätze**.

Graphisch werden Plätze als Kreise dargestellt, Tokens als Punkte in den Kreisen, Transitionen als Rechtecke und Kanten als Linien. Abbildung 2.7 zeigt ein einfaches Petrinetz. Die Transition *t1* ist aktiv, da ihr Eingangsplatz *p1* ein Token enthält. Wenn die Transition *t1* feuert, konsumiert sie das Token im Platz *p1* und erzeugt ein neues Token im Platz *p2*, wie im rechten Teil der Abbildung dargestellt.

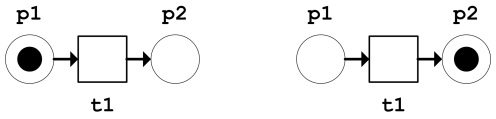


Abbildung 2.7: Ein einfaches Petrinetz vor und nach dem Feuern

Gibt es mehrere solcher Eingangsplätze für eine Transition, kann diese in der ursprünglichen Form eines Petrinetzes nur dann feuern, wenn in *jedem* Eingangsplatz ein Token liegt. Demgemäß erzeugt dann die Transition in *jedem* Ausgangsplatz ein Token.

Eine übliche Erweiterung dieser einfachen Form der Petrinetze besteht darin, mehrere Tokens pro Platz zuzulassen. Plätze bekommen dazu eine **Markierung**, die aus einer Ganzzahl besteht und anzeigt, wieviele Tokens im jeweiligen Platz enthalten sind. Das Feuerverhalten der Transitionen kann dann verschieden definiert werden, so dass wahlweise weiterhin nur jeweils ein Token pro Eingangsplatz verbraucht wird oder aber mehrere bzw. alle. Entsprechend können in den Ausgangsplätzen ein oder mehrere Tokens erzeugt werden.

Farbige Petrinetze (Colored Petri nets, CPN (Jensen, 1996)) erlauben darüber hinaus die Unterscheidung von Tokens. Dazu werden verschiedene Typen von Tokens definiert, die jeweils eine Farbe zugewiesen bekommen. Die Farbe steht dabei stellvertretend für den Typ und dient der einfachen graphischen Darstellung; verschiedene geometrische Formen von Tokens wären zur Unterscheidung ebenso denkbar. Formal wird bei farbigen Petrinetzen häufig eine Nummerierung der Tokens gewählt, die dann erst zur graphischen Darstellung in unterschiedliche Farben übersetzt wird.

Benutzt man sowohl die Erweiterung um mehrere Tokens pro Platz als auch die um farbige Tokens, so lassen sich durch entsprechend angepasste Regeln für das Feuerverhalten der Transitionen recht komplexe Abläufe anhand vergleichsweise kleiner

Petrinetze modellieren. Eine Transition kann dann beispielsweise in Abhängigkeit der Eingabetokens eine unterschiedliche Anzahl verschiedenfarbiger Ausgabetokens pro Ausgangsplatz erzeugen. Ohne solche Erweiterungen müssen zur Modellierung komplexer Schaltbedingungen eine Vielzahl von Hilfsplätzen und -transitionen eingeführt werden, die das Petrinetz anwachsen lassen und oftmals unübersichtlich machen (Abschnitt 3.2.2 enthält ein Beispiel).

Aus der erweiterten Modellierung ergeben sich erweiterte Analysemöglichkeiten. Diese wurden bisher kaum erforscht, auch wenn Ansätze wie Fan et al. (2007) erste Schritte in diese Richtung darstellen. Ein Hinderungsgrund liegt möglicherweise in der Analysekomplexität, die exponentiell wächst (Murata (1989) bzw. Lipton (1975)): Die Anzahl Pfade durch ein Petrinetz steigt exponentiell mit der Anzahl der verzweigenden Elemente, so dass im allgemeinen Fall auch die benötigte Rechenzeit und der benötigte Speicherplatz exponentiell wachsen (Stichwort „state space explosion“, der zu bearbeitende Zustandsraum hat explosionsartiges Wachstum). Für „kleine“ Petrinetze mit einer zweistelligen Anzahl Elemente lassen sich dennoch viele erweiterte Analysen durchführen. Ein großer Teil der in der Praxis modellierten Workflows fällt in diese noch analysierbare Größenordnung (Accorsi und Wonnemann, 2011).

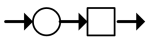
Die von einer Ausführungsumgebung zu unterstützenden Workflow-Muster beschreiben van der Aalst et al. (2003). Diese 20 Muster werden in Abbildung 2.8 zusammengefasst. Zur Erfassung der Muster haben van der Aalst et al. (2003) 15 industrielle Workflow-Systeme (Staffware, COSA, InConcert, Eastman, FLOWer, Domino, Meteor, Mobile, MQSeries, Forté, Verve, Visual WorkFlow, Changengine, I-Flow, SAP R/3) untersucht. Damit beschreiben die Muster wenn nicht alle, dann doch die große Mehrheit der in der Praxis vorkommenden Workflow-Konstrukte.

Theoretisch lassen sich alle diese Muster mit einfachen Petrinetzen modellieren. Das Schaltverhalten ist dann aber nicht deterministisch. Das Muster Multi-Choice (Nr. 6 in Abbildung 2.8) beispielsweise aktiviert eine gewisse Anzahl der Folgepfade. Im nichtdeterministischen Fall müssen alle möglichen Kombinationen analysiert werden. Daraus ergeben sich unter Umständen falsche Warnungen (*false positives*), also fälschlicherweise als Verstoß angezeigte Aktivitäten bzw. Pfade. Angenommen, von fünf Folgepfaden wird die Kombination Pfad 2+Pfad 5 als regelverletzend eingestuft, ist aber wegen der im Workflow festgelegten Schaltlogik gar nicht aktivierbar. Damit wäre die Meldung praktisch gesehen ein *false positive*, auch wenn es theoretisch zum Verstoß kommen könnte. Für die praktische Analyse ist es also sinnvoll, durch ein erweitertes Feuerverhalten die Schaltlogik des Workflows präziser nachbilden zu können. Dazu können die Transitionen der Workflow- und Regel-Petrinetze deterministisch definiert werden, wie im dritten Kapitel gezeigt werden wird.

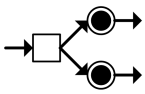
2.2.3 Workflow-Elemente

Bereits im Jahr 1992 haben Curtis et al. (1992) die Elemente IT-unterstützter Prozesse sehr treffend klassifiziert. Von Stohr und Zhao (2001) wurde die Klassifikation im Jahr

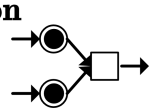
1: Sequence



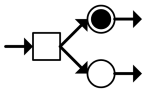
2: Parallel Split



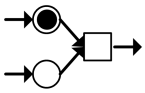
3: Synchronization



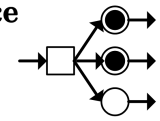
4: Exclusive Choice



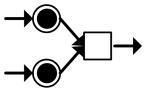
5: Simple Merge



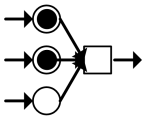
6: Multi-Choice



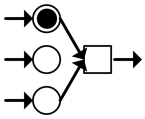
7: Synchronizing Merge



8: Multi-Merge



9: Discriminator



10: Arbitrary Cycles

11: Implicit Termination

12: Multiple Instances Without Synchronization

13: Multiple Instances With A Priori Design Time Knowledge

14: Multiple Instances With A Priori Runtime Knowledge

15: Multiple Instances Without A Priori Runtime Knowledge

16: Deferred Choice

17: Interleaved Parallel Routing

18: Milestone

19: Cancel Activity

20: Cancel Case

(Patterns 10-20 do not have a unique Petri net representation)

Abbildung 2.8: Workflow-Patterns von van der Aalst et al. (2003)

2001 nur geringfügig erweitert.

Curtis, Kellner und Over (1992)

Curtis et al. (1992) zufolge wurden bei der Modellierung von Informationssystemen vor 1992 hauptsächlich Datenflüsse und Transformationen berücksichtigt. Unabhängig vom Einsatzgebiet besteht ein Prozess laut den Autoren aus den vier folgenden Elementen, die in dem Artikel als Grundlage der Prozessmodellierung beschrieben werden:

- **Aktivitäten**: Einzelne Schritte eines Prozesses (*process steps*).
- **Akteure**: Personen, die Aktivitäten ausführen.
- **Rollen**: Funktional zugeordnete Menge von Prozesselementen wie Aktivitäten und Artefakte, die ein Akteur zur Erfüllung seiner Aufgabe benötigt.
- **Artefakte**: Prozesselemente, die durch Aktivitäten erzeugt oder bearbeitet werden.

Bei der Modellierung dieser Elemente können verschiedene Perspektiven eingenommen werden. Die Herausforderung besteht laut Curtis et al. darin, ein einheitliches Modell für alle Perspektiven zu finden, bzw. eine gemeinsame Grundlage zu definieren, die die Informationen aller Perspektiven vereint. Die vier im Artikel diskutierten Perspektiven lauten wie folgt:

- **Functional perspective**: Welche Aktivitäten werden ausgeführt, und welche Artefakte werden zur Ausführung benötigt.
- **Behavioral perspective**: In welcher Reihenfolge und unter welchen Bedingungen werden die Aktivitäten ausgeführt.
- **Organizational perspective**: Welche Akteure führen welche Aktivitäten wo aus. Auch die Übertragungsart und der Speicherort von Artefakten gehört zu dieser Perspektive.
- **Informational perspective**: Welche Artefakte werden im Prozess erzeugt oder bearbeitet und in welchem Bezug stehen sie zueinander.

Sowohl die eher auf reine Prozessbeschreibung zielenden wie auch die auf Prozessautomatisierung gerichteten Ansätze, die im Artikel gegenübergestellt werden, lassen sich einer oder mehreren dieser Perspektiven zuordnen. Die vier Modellierungselemente Aktivitäten, Akteure, Rollen und Artefakte sind sowohl vom Einsatzgebiet wie von der Perspektive unabhängig.

Stohr und Zhao (2001)

Stohr und Zhao (2001) greifen die obige Arbeit von Curtis et al. auf und fügen noch eine fünfte Perspektive hinzu, nämlich die *operational perspective*. Aus dieser Perspektive stehen die Werkzeuge und die Software im Vordergrund, mit der die Aktivitäten eines Workflows ausgeführt werden.

Basierend auf der Workflow Management Coalition (WfMC) definieren Stohr et al. einen Geschäftsprozess als Abfolge von Aktivitäten mit klar definierten Ein- und Ausgaben. Eine Aktivität ist ein einzelner, von einem Menschen oder einer Maschine ausgeführter Schritt des Prozesses. Ein Workflow wird definiert als „the automation of a business process in whole or in part, during which documents, information or tasks are passed from one participant to another according to a set of procedural rules“. Daraus ergeben sich als Workflow-Elemente:

- **Aktivitäten**: Einzelne Schritte im Workflow.
- **Akteure**: Menschen oder Maschinen, die Aktivititäten ausführen.
- **Daten**: Dokumente oder Informationen im Workflow.
- **Regeln**: Vorgaben („Schaltregeln“) bezüglich des Workflow-Ablaufs.
- **Rollen**: Sammlung von Aktivitäten und Verantwortlichkeiten, die einem Akteur zugewiesen werden können.

Die Regeln beschreiben nur, welche Daten zwischen welchen Aktivitäten ausgetauscht werden (Ist-Zustand). Es handelt sich dabei *nicht* um Compliance-Regeln (Soll-Zustand).

Der Begriff der Verantwortlichkeiten wie sie Teil einer Rolle sein können bezieht sich auf die noch zu erledigenden Aktivitäten, die in der einem Akteur zugewiesenen Aufgabenliste enthalten sind.

2.3 Fallbeispiel

Der in Abbildung 2.9 dargestellte, allgemeine Ablauf einer Behandlung im Krankenhaus dient im Folgenden als Fallbeispiel. Nach der Anmeldung wird der Patient untersucht und durchläuft dann je nach Schwere der erkannten Erkrankung einen von drei Workflow-Pfaden. Wird keine oder nur eine leichte Erkrankung erkannt, die weder eine Operation noch Medikation erfordert, wird der Patient beraten und dann entlassen. Bei einer mittelschweren Erkrankung, die keine Operation, wohl aber Medikation erfordert, erfolgt die Medikament-Auswahl und eine Unterrichtung, wie das Medikament einzunehmen ist. Bei einer schweren Erkrankung erfolgt eine Operation. Dazu wird zunächst der Patient über die Risiken der Operation aufgeklärt. Parallel

wird dann der Operationssaal vorbereitet und der Patient auf direkte Operationsrisiken (wie z.B. Intubationsrisiken) untersucht. Sollten sich bei der an die Operation anschließenden Überwachung Komplikationen zeigen, wird erneut operiert. Ansonsten werden die zur Weiterbehandlung nötigen Medikamente ausgewählt und der Patient, genau wie im Falle einer mittelschweren Erkrankung, über die Einnahme der Medikamente unterrichtet. In allen Fällen werden die Beratungs- bzw. Behandlungsdaten in die elektronische Patientenakte (EPA) eingetragen. Die EPA muss dann von einem Arzt abgezeichnet werden. Nach der Entlassung endet der Workflow, sofern sich der Patient nicht wegen weiterhin bestehender Probleme erneut zur Untersuchung meldet.

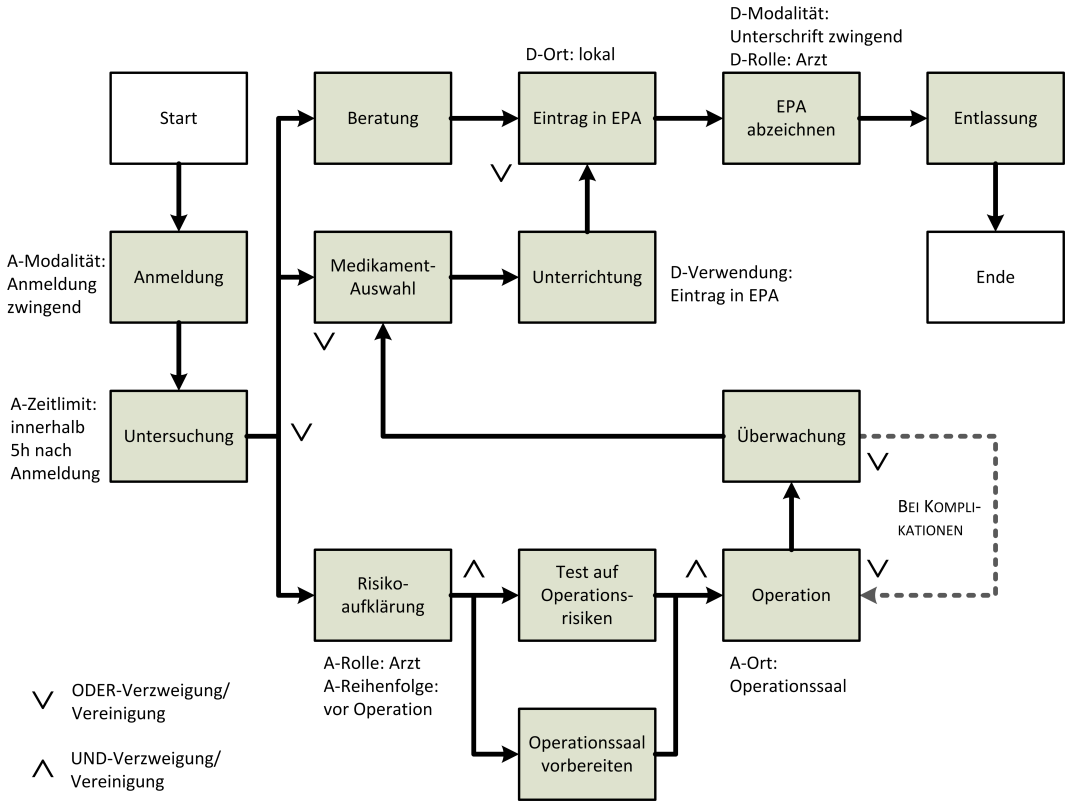


Abbildung 2.9: Fallbeispiel: Workflow für Krankenhaus-Behandlung

Jedem der in Abschnitt 2.1.3 vorgestellten Regel-Elemente entsprechend findet sich in Abbildung 2.9 eine Regel-Angabe im Workflow. Die Anmeldung ist zwingender Bestandteil des Workflows (Modalität für Aktivitäten). Zwischen der Anmeldung und der Untersuchung dürfen nicht mehr als fünf Stunden verstreichen (Zeitlimit für Aktivitäten). Die Daten der EPA dürfen nur lokal abgelegt (Ort für Daten) und nur von einem Arzt abgezeichnet werden (Rolle für Daten). Die Unterschrift ist zwingender Bestandteil des Workflows (Modalität für Daten). Die Angaben über das ausgewählte Medikament müssen in der EPA abgelegt werden (Verwendung für Daten). Die Risikoaufklärung darf nur durch den Arzt geschehen (Rolle für Aktivitäten) und muss vor der

Operation erfolgen (Reihenfolge für Aktivitäten). Die Operation darf nur im OP-Saal stattfinden (Ort für Aktivitäten).

Über die Regel-Angaben hinaus sind im Workflow die Typen der Verzweigungen und Vereinigungen von Workflow-Pfaden markiert. Für die Korrektheitsprüfung von Workflows ist es wichtig, die Unterscheidung zwischen dem UND-Typ und dem ODER-Typ treffen zu können. Beim UND-Typ werden alle Pfade einer Verzweigung aktiviert, oder es wird auf die Beendigung aller Pfade einer Vereinigung gewartet. Beim ODER-Typ wird einer oder werden mehrere Pfade einer Verzweigung aktiviert, oder es wird auf die Beendigung eines oder mehrerer Pfade einer Vereinigung gewartet. In einem korrekten Workflow sind immer nur Verzweigungen und Vereinigungen vom selben Typ kombiniert. Die Kombination einer ODER-Verzweigung mit einer UND-Vereinigung führt zum Deadlock, die Kombination einer UND-Verzweigung mit einer ODER-Vereinigung zu potentiell bei Workflow-Ende noch nicht beendeten Teilpfaden.

2.4 Bestehende Analyseverfahren

Die automatisierte Suche nach Schwachstellen in Workflows erfährt wegen der fortschreitenden Automatisierung der Workflow-Ausführung mehr und mehr Aufmerksamkeit. Im Bereich der Software-Entwicklung wird die (automatisierte) Schwachstellenanalyse schon deutlich länger betrieben und durch eine Vielzahl von Verfahren unterstützt. Den Anfang bildeten eher manuelle Ansätze wie die Angriffsbäume und die FMEA. Angriffsbäume (*Attack Trees* wie von Schneier (1999) beschrieben) bzw. ursprünglich Fehlerbäume (*Fault Trees* (Vesely et al., 1981); eine gute Einführung geben Moore et al. (2001)) gehen von einem Angriff aus und suchen rückwärts nach möglichen Angriffsschritten. Die Fehler-Möglichkeits-und-Einfluss-Analyse (FMEA, (Department of Defense, 1980)) hingegen beginnt mit einem Fehler und sucht vorwärts nach möglichen Konsequenzen für das untersuchte System. Sie wurde ursprünglich in der Luft- und Raumfahrttechnik eingesetzt und findet mittlerweile auch im Softwarebereich Verwendung (Georgieva, 2010). Über Angriffsbäume und die FMEA hinaus sind inzwischen hochgradig automatisierte Ansätze entwickelt worden. Beispiele dafür sind die Taint-Analyse für Java-Quellcode durch „Griffin“ (Livshits, 2006; Martin et al., 2005) und die Programmabhängigkeitsgraphen wie sie von (Hammer, 2009; Hammer und Snelting, 2009) zur Informationsflusskontrolle für Java-Bytecode vorgeschlagen werden.

Workflows und Quellcode beschreiben beide eine Abfolge von auszuführenden Aktivitäten, die bestimmte Regeln einhalten sollen. Regeln für Workflows beschreiben beispielsweise, welche Rolle welche Aktivitäten ausführen oder auf welche Daten zugreifen darf. Organisatorischen Anforderungen entsprechend werden Workflow-Regeln häufig explizit festgelegt, während zumindest zu Beginn der Software-Entwicklung die Regeln für Quellcode nur implizit als Annahmen in den Köpfen der Softwareentwickler vorhanden waren (Krsul, 1998). Im Laufe der Zeit entstand eine Vielzahl von Klassifikationen, die beschreiben, auf welche Arten im Quellcode die Einhaltung der im-

pliziten und expliziten Regeln durch Schwachstellen gefährdet werden. Im Rahmen der vorliegenden Arbeit stellte sich daher die Frage, ob die in solchen Klassifikationen gesammelten Beschreibungen möglicher Regelverstöße nicht zur Suche nach Schwachstellen in Workflows genutzt werden können. Bei der Analyse der in Tabelle 2.1 aufgelisteten Klassifikationen von Quellcode-Schwachstellen zeigte sich aber dann, dass die beschriebenen Schwachstellen entweder ganz spezifisch nur auf der Software-Ebene auftreten (vgl. Ebenen-Modell in Abbildung 1.2) oder aber nur zu eher allgemeinen Hinweisen wie „Zugriffskontrolle im Workflow ist nötig“ führen. Die Übertragung der Quellcode-Schwachstellen-Beschreibungen auf mögliche Workflow-Schwachstellen wurde daher nicht weiter verfolgt.

Nachfolgend werden Prüfverfahren vorgestellt, mit denen Schwachstellen auf Workflow-Ebene entdeckt werden können. Zur Vorbereitung bzw. Einführung der Verfahren werden zwei Arbeiten kurz behandelt, die allgemeine Daten- bzw. Kontrollfluss-Probleme klassifizieren.

Petrinetze sind zur Modellierung von Workflows besonders gut geeignet, wie etwa Curtis et al. (1992); van der Aalst (1996); Stohr und Zhao (2001) zeigen. Deswegen beruht auch das neu entwickelte, in Kapitel 3 beschriebene Prüfverfahren auf Petrinetzen. Ebenfalls vor diesem Hintergrund werden in Abschnitt 2.4.1 nur einige wenige nicht auf Petrinetzen basierende Verfahren vorgestellt, bevor in Abschnitt 2.4.2 die auf Petrinetze aufbauende Mehrheit der zum Vergleich herangezogenen Verfahren diskutiert wird.

Sadiq et al. (2004) diskutieren sieben mögliche Datenfluss-Probleme, die bei der Validierung eines Workflow-Modells entdeckt werden müssen:

- Redundant Data: Ein Element der Ausgabe-Daten einer Aktivität entspricht keinem Element der Eingabe-Daten folgender Aktivitäten.
- Lost Data: Zwei Aktivitäten in parallelen Pfaden erzeugen das gleiche Datenelement. Beim Zusammenführen der Pfade wird nur ein Element verwendet, das andere geht verloren.
- Missing Data: Ein Element der Eingabe-Daten einer Aktivität entspricht keinem Element der Ausgabe-Daten vorheriger Aktivitäten.
- Mismatched Data: Ein Element der Ausgabe-Daten einer Aktivität ist anders formatiert als für die Eingabe-Daten einer folgenden Aktivität benötigt.
- Inconsistent Data: Verschiedene Aktivitäten beschreiben das gleiche Datenelement und aufgrund unklarer Zugriffsreihenfolge wird nicht der aktuellste Wert weiterverwendet.
- Misdirected Data: Die für eine Aktivität benötigten Daten werden erst von einer folgenden Aktivität erzeugt.

Tabelle 2.1: Übersicht über Schwachstellenklassifikationen

Arbeit	**Jahr**	**Fokus**	**C**	**O**	**A/W**	**V**
RISOS (Abbott et al., 1976)	1976	OS	-	-	✓	✓
PA (Bisbey II und Hollingworth, 1978)	1978	OS	✓	-	-	-
Hogan (1988)	1988	-	-	✓	-	-
Neumann und Parker (1989)	1989	Syst./Net	-	-	✓	-
NRL (Landwehr et al., 1994)	1994	Software	✓	✓	-	✓
Brinkley und Schell (1995)	1995	System	-	-	✓	-
Aslam (1995)	1996	OS	✓	-	-	-
Cohen (1997)	1997	System	-	-	✓	-
Lindqvist und Jonsson (1997)	1997	-	-	✓	-	-
Du und Mathur (1997)	1997	Software	✓	-	✓	-
Howard und Longstaff (1998)	1998	-	✓	-	-	-
Krsul (1998)	1998	Software	✓	✓	-	-
Piessens et al. (2001)	2001	Sw./Net	✓	✓	-	-
Jiwnani und Zelkowitz (2002)	2002	Software	✓	-	✓	✓
Alvarez und Petrovic (2003)	2003	Web	✓	-	✓	-
Web Appl. Sec. Consortium (2004)	2004	Web	✓	-	✓	-
Langweg und Snekkenes (2004)	2004	App. Sw.	✓	-	✓	✓
Lindstrom (2004)	2004	WS	-	-	✓	-
19 Sins (Howard et al., 2005)	2005	Software	✓	-	✓	-
7 Kingdoms (Tsipenyuk et al., 2005)	2005	Software	✓	-	-	-
Weber et al. (2005)	2005	Software	✓	-	-	-
Vanden Berghe et al. (2005)	2005	WS	✓	-	-	-
PLOVER (Christey, 2006)	2006	-	✓	✓	-	-
Parrend und Frenot (2007)	2007	Java	✓	-	✓	✓
OWASP (OWASP Foundation, 2009)	2009	Web	✓	-	-	-
SANS Top 20 (SANS, 2009)	2009	Sw./Web	-	-	-	✓
24 Sins (Howard et al., 2009)	2009	Software	✓	-	✓	✓
CAPEC (MITRE Corporation, 2011a)	2011	Sw./Web	-	-	✓	✓
CWE (MITRE Corporation, 2011b)	2011	Software	✓	-	✓	✓
OVAL (MITRE Corporation, 2011c)	2011	Software	-	-	✓	✓

C=Codeproblem, O=Organisatorische Ursache, A/W=Ausnutzung/Wirkung, V=Verwundbare Stelle. OS=Operating System, WS=Web Services

- Insufficient Data: Die vorliegenden Daten sind „nicht ausreichend“ um eine Aktivität erfolgreich zu beenden („nicht ausreichend“ wird im Artikel nicht erklärt).

Zur Aufdeckung von Kontrollfluss-Problemen verwenden Liu und Kumar (2005) das Prinzip der korrespondierenden Paare. Demnach sind nur bestimmte Kombinationen von Verzweigungsoperatoren zulässig, wenn es sich um einen strukturierten Workflow handelt; unstrukturierte Workflows dürfen zusätzliche Kombinationen enthalten, aber nicht alle beliebigen, weil sonst Deadlocks entstehen oder Aktivitäten unnötig mehrfach ausgeführt werden.

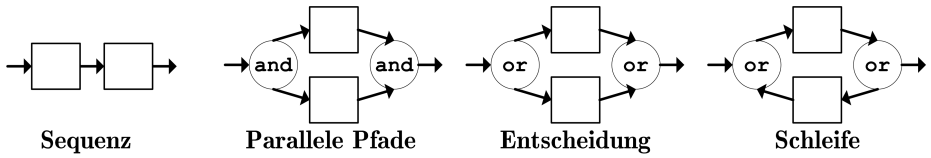


Abbildung 2.10: Vier Kontrollflussmuster strukturierter Workflows (Liu und Kumar, 2005)

Strukturierte Workflows lassen sich in die in Abbildung 2.10 gezeigten Muster zerlegen: die Sequenz als Abfolge von Aktivitäten, parallele Pfade als pseudo-gleichzeitige Ausführung von Aktivitäten (Systeme die keine echte Parallelität erlauben führen entweder die eine oder die andere Aktivität zuerst aus), die Entscheidung als Auswahl aus einer Menge von Ausführungspfaden, und die Schleife als Wiederholung einer Aktivität oder eines ganzen Ausführungspfads.

Wie in Abbildung 2.10 müssen die Pfade nach einer Und-Verzweigung (*AND split*) mit einer Und-Vereinigung (*AND join*) wieder zusammengeführt werden, ebenso nach einer Oder-Verzweigung (*OR split*) mit einer Oder-Vereinigung (textitOR join). Unstrukturierte Workflows, die dem Beitrag zufolge nur von wenigen Praxislösungen unterstützt werden, bieten dem Workflow-Designer weitere Kombinationsmöglichkeiten. Davon führen aber mehr als die Hälfte zu Deadlocks, wie Liu et al. durch vollständige Enumeration der Kombinationen zeigen.

2.4.1 Nicht-Petrinetz-basierte Verfahren

Sadiq und Orlowska (1999, 2000)

Auf Basis des WfMC-Workflow-Modells der Workflow Management Coalition (1999) schlagen Sadiq und Orlowska (1999) ein Verfahren zur Graph-Reduktion vor. Workflows werden dabei als Graph dargestellt und durch wiederholte Anwendung von Reduktionsregeln soweit wie möglich reduziert. Korrekte Workflows, also ohne Deadlocks und mit Synchronisation, werden auf zwei Elemente reduziert. Sind Deadlocks vorhanden oder ist die Synchronisation unzureichend, bleiben mehr Elemente übrig. Das Verfahren dient somit der „visuellen Identifikation struktureller Konflikte“ (Sadiq und Orlowska, 1999).

In einer überarbeiteten Version erschien der Artikel ein Jahr später wieder (Sadiq und Orlowska, 2000). Da die Unterschiede zwischen den Versionen hauptsächlich graphischer Natur sind und der Wortlaut überwiegend exakt bestehen blieb, wird die spätere Version nicht als eigenständiges Verfahren diskutiert.

Giblin, Liu, Müller, Pfitzmann und Zhou (2005, 2006)

Giblin et al. stellen mit REALM einen Ansatz vor, der die Formulierung von Compliance-Regeln vor allem für zeitliche Belange als logische Ausdrücke erlaubt (Giblin et al., 2005, 2006). Die drei diskutierten Regeltypen sind:

- y Within(c) After x: wenn Aktivität X ausgeführt wird, muss anschließend innerhalb einer gewissen Frist Aktivität Y ausgeführt werden.
- x Within(c) Before y: wenn Aktivität Y ausgeführt werden soll, muss vorher innerhalb einer gewissen Zeitspanne Aktivität X ausgeführt worden sein.
- x AtLeast(c) Before y: wenn Aktivität Y ausgeführt werden soll, muss Aktivität X mindestens eine gewisse Zeitspanne im Voraus ausgeführt worden sein.

Die logische Ergänzung der drei Fälle um den vierten Fall, **y AtLeast(c) After x**, wird von Giblin et al. nicht diskutiert.

REALM verwendet temporale Logik, womit die laut Giblin et al. häufig in Compliance-Regeln geforderten zeitlichen Aspekte der Reihenfolge von Aktivitäten und der Einhaltung von Fristen überprüft werden können. In welcher Form die Ergebnisse ausgegeben werden, bleibt unklar. Sollte die Ergebnisanzeige untransformiert erfolgen, könnte diese nur von in temporaler Logik bewanderten Benutzern interpretiert werden. Die Rückführung der logischen Ausdrücke auf problematische Stellen im Workflow müsste dann manuell erfolgen.

Anders als die in der vorliegenden Arbeit angestrebte Prüfung zur Entwurfszeit zielt REALM auf das Monitoring, also die Überwachung von Workflows zur Laufzeit. Es handelt sich also um einen ergänzenden Ansatz, der die bei der Entwurfszeit-Prüfung nicht endgültig entscheidbaren Fragen wie z.B. nach der tatsächlichen Einhaltung von Fristen für die Aktivitätsausführung dann während der Laufzeit überwachen kann.

Liu, Müller und Xu (2007)

Liu et al. (2007) stellen in ihrem *Open Process AnaLyzer* (OPAL)-Framework eine Methode vor, mit der BPEL-Geschäftsprozesse auf die Einhaltung von in der Business Property Specification Language (BPSL) formulierten Compliance-Regeln geprüft werden können. Die Prozesse werden dabei zunächst in den Pi-Calculus transformiert und dann als endliche Automaten (finite state machines, FSM) dargestellt. Die Regeln werden in linearer temporaler Logik (LTL) formuliert. Die eigentliche Prüfung erfolgt mittels Model Checking.

Zur Identifikation von Fehlern bzw. Regelverstößen im Geschäftsprozess greifen Liu et al. auf die von Xu et al. definierten *business bug patterns* (Xu et al., 2006) zurück. Diese Fehlermuster wiederum stellen die Gegenstücke zu den Workflow-Patterns von van der Aalst et al. (2003) dar (siehe dazu Abschnitt 3.2) und beschreiben, welche Kombinationen von Aktivitäten bzw. Verzweigungselementen in einem Workflow *nicht* auftreten dürfen.

Beispiel. Der *Sequential Bug* (Xu et al., 2006) betrifft die Fälle, in denen das Sequenz-Muster aus van der Aalst et al. (2003) verletzt wird. In einer Sequenz muss B auf A folgen. Eine Verletzung liegt dann vor, wenn A und B gleichzeitig ausgeführt werden oder wenn A ausgeführt wird und das anschließende B fehlt. Das Fehlermuster ist demnach definiert als „$SimultaneousStart(A, B) \vee NoResponse(A, B)$“ (Xu et al., 2006). ■

Xu et al. (2006) weisen darauf hin, dass es in der Praxis nützlicher ist, anhand solcher Muster nach Fehlern zu suchen, als die Korrektheit von komplexen Geschäftsprozessen zu beweisen. Weiterhin argumentieren sie, dass Model Checking sich besser zur Aufdeckung von Fehlern in einem System eignet als zum Beweis der Korrektheit des Systems. Das Verfahren von Liu et al. (2007) setzt daher die Suche anhand von Fehlermustern mit Model Checking um. Der besseren Praxistauglichkeit wegen werden nach dem Suchlauf die Ergebnisse, die zunächst nur mit Kenntnis der Theorie endlicher Automaten interpretierbar sind, zu ihren Ursprüngen im Geschäftsprozess zurückverfolgt und von OPAL in der BPEL-Darstellung angezeigt. Um die Analyse nachvollziehen zu können, sind allerdings Detailkenntnisse mehrerer, für sich genommen bereits komplexer Formalismen nötig.

2.4.2 Petrinetz-basierte Verfahren

Bereits 1992 haben Curtis et al. (1992) die besonders gute Eignung von Petrinetzen zur Workflow- bzw. Prozessmodellierung hervorgehoben. Auch van der Aalst (1996) schlägt die Verwendung von Petrinetzen als Standard für Workflows bzw. Workflow-Mangementsysteme vor. Wie van der Aalst schlagen Stohr und Zhao (2001) Petrinetze nicht nur zur Modellierung, sondern besonders auch zur Analyse von Workflows vor.

Nachfolgend werden Analyseverfahren vorgestellt, die auf der Basis von Petrinetzen den Daten- und/oder Kontrollfluss eines Workflows prüfen. Der Abschnitt 2.5 bewertet die Verfahren hinsichtlich ihrer Eignung zur Compliance-Prüfung.

Adam, Atluri und Huang (1998)

Adam et al. (1998) greifen drei von Rusinkiewicz und Sheth (1995) diskutierte Kontrollflussabhängigkeiten auf:

- **control-flow dependencies**: Von Rusinkiewicz und Sheth (1995) als *execution dependencies* bezeichnet. Geben an, in welcher Reihenfolge Aktivitäten ausgeführt werden dürfen oder müssen.

- **value dependencies**: Von Rusinkiewicz und Sheth (1995) als *data or value dependencies* bezeichnet. Geben an, welche Ausgaben von anderen Aktivitäten für die Ausführung bestimmter Aktivitäten vorliegen müssen.

- **external dependencies**: Von Rusinkiewicz und Sheth (1995) als *temporal dependencies* bezeichnet, geben an, von welchen externen Variablen der Kontrollfluss abhängt. Ein prominentes Beispiel für solche Variablen ist die Zeit.

Um einen Workflow auf die Erfüllung der verschiedenen Abhängigkeiten zu prüfen, bauen Adam et al. auf die von van der Aalst (1996) in Petrinetze übertragenen sechs Workflow-Primitive der Workflow Management Coalition auf: AND join, AND split, OR join, OR split, Iteration, Causality.

Weiterhin formalisieren sie, Georgakopoulos et al. (1995) entsprechend, einen Workflow als eine Menge von Aktivitäten inklusive der Aufrufreihenfolge und dem Informationsfluss zwischen den Aktivitäten.

Jede Aktivität schaltet darüber hinaus ein Element der Menge *{begin, abort, precommit, commit}* . Die Ausführung beginnt jeweils nach *begin* und endet mit *commit* oder *abort*. Darauf und auf den Zwischenzuständen aufbauend kann dann die Erfüllung der Abhängigkeiten zwischen den Aktivitäten geprüft werden. So muss beispielsweise jede Aktivität bei Beendigung des Workflows ebenfalls beendet worden sein; falls für eine Aktivität nur *begin*, aber dann weder *commit* noch *abort* ausgelöst wurde, liegt ein Fehler vor.

Van der Aalst, Hirnschall und Verbeek (2002)

Die Definition der Workflow-Korrektheit von Sadiq und Orlowska (1997) aufgreifend stellen van der Aalst et al. (2002) ein Verfahren vor, dass die Abwesenheit von Deadlocks und die Wahrung der Synchronisation in einem Workflow bestätigen soll. Im Gegensatz zu dem Verfahren von Sadiq und Orlowska (1999), welches - wie im Artikel durch ein Gegenbeispiel gezeigt - nicht alle Workflows korrekt analysiert, werden dabei anhand weniger Reduktionsregeln alle Workflows richtig untersucht.

Wie van der Aalst et al. (2002) in der Zusammenfassung ihres Artikels beschreiben, ist die Analysekomplexität des auf Petrinetzen beruhenden Verfahrens geringer als die des nicht mit Petrinetzen arbeitenden Verfahrens von Sadiq und Orlowska (1999). Die Petrinetz-Modellierung erlaubt hier also eine präzisere Analyse bei höherer Performance, was die Attraktivität von Petrinetzen zur Workflow-Analyse unterstreicht.

Sun, Zhao, Nunamaker und Sheng (2006)

Aufbauend auf der Beobachtung, dass vorherige Analyseverfahren zur weit überwiegenden Mehrheit nur den Kontroll-, aber nicht den Datenfluss betrachten, haben Sun et al. (2006) die Verwendung einer Datenflussmatrix zur Analyse vorgeschlagen und ein entsprechendes Verfahren vorgestellt. Dabei werden drei Fälle behandelt: fehlende,

mehrfache und verlorene Datenelemente. Die von Sadiq (siehe Kapitel 2.4) beschriebenen sieben Fälle werden in dem Artikel diskutiert und als aus den drei Fällen zusammensetzbar dargestellt. Neben einer ausführlichen Formalisierung von Regeln zur Datenflussverifikation beschreiben Sun et al. jeweils einen Verifikationsalgorithmus für die Prüfung auf fehlende, mehrfache und verlorene Daten. Der Artikel stellt eine Verfeinerung des bereits 2004 erschienen Artikels von Sun et al. (2004) dar.

Vanhatalo, Völzer und Leymann (2007)

Um die Kontrollflussanalyse zu beschleunigen, verfolgen Vanhatalo et al. (2007) den Ansatz der Workflow-Dekomposition. Sie zerlegen einen Workflow in *single entry, single exit* (SESE) - Komponenten, also Ausführungsabschnitte, in die und aus denen jeweils nur ein einzelner Pfad führt. Diese Komponenten werden dann auf Korrektheit (*soundness*) geprüft, im Artikel definiert als die Abwesenheit von Deadlocks und die Anwesenheit von synchronisierenden Elementen nach Verzweigungen.

Vanhatalo et al. heben die Bedeutung einer schnellen und mögliche Fehler präzise identifizierenden Prüfung hervor. In vielen Fällen lässt sich der präsentierten Fallstudie zufolge die Analysekomplexität von der sonst meist quadratischen, kubischen oder sogar exponentiellen Stufe auf eine lineare senken. Die Schnelligkeit des Verfahrens rührt neben der Dekompositionsstrategie auch von der Verwendung einer Heuristik her, mit der Komponenten in korrekte und nicht korrekte eingeteilt werden. Für manche Komponenten muss dennoch eine vollständige Analyse mit den aufwändigeren Verfahren durchgeführt werden. Bei der Fehleridentifikation ist das Verfahren insofern vorteilhaft, als es Fehler nicht nur grundsätzlich erkennt, sondern auch die betroffene SESE-Komponente identifiziert. Innerhalb dieser Workflow-Fragmente muss der Fehler allerdings manuell lokalisiert werden.

Die Analyse basiert auf Workflow-Graphen, die von Vanhatalo et al. in starker Anlehnung an Petrinetze definiert werden. Aus den Workflow-Graphen wird durch die Dekomposition ein sogenannter *process structure tree* erzeugt, wofür linearer Aufwand nötig ist. Die Dekomposition stellt durch diese Aufwandsreduktion einen interessanten Ansatz für die Workflow-Analyse dar. Aufgrund der je nach Analyseziel möglichen Abhängigkeiten zwischen Komponenten ist die Dekomposition aber nicht in jedem Fall sinnvoll einsetzbar, da auf den kompletten Workflow bezogene Eigenschaften nicht oder nur umständlich erfasst werden können.

Fan, Dou und Chen (2007)

Fan et al. (2007) heben nicht nur die Bedeutung der Analyse sowohl des Kontrollflusses wie auch des Datenflusses hervor, sondern zeigen auch, dass durch Abhängigkeiten zwischen den beiden Flüssen selbst dann Fehler auftreten können, wenn die Flüsse für sich genommen fehlerfrei sind.

Als Beispiel für einen solchen Fehler, der bei der Kombination eines widerspruchsfreien Kontrollflussmodells mit einem widerspruchsfreien Datenflussmodell auftreten kann,

verwenden Fan et al. einen einfachen Workflow für die Entwicklung eines Krans. Der in Abbildung 2.11 gezeigte Workflow besteht aus vier Aktivitäten, nämlich *t1*, Anforderungserhebung beim Kunden, *t2*, Entwurf des Auslegers, *t3*, Entwurf der Stromversorgung und *t4*, Test des Krans. Für den Entwurf des Auslegers werden die Anforderungen des Kunden benötigt. Für den Entwurf der Stromversorgung werden neben den Anforderungen des Kunden auch einige Ergebnisse des Auslegerentwurfs benötigt, da die Stromversorgung für den Ausleger genügend stark sein muss. Für den Test schließlich werden die Ergebnisse aus allen vorherigen Aktivitäten benötigt. Diese Datenabhängigkeiten werden im Datenflussmodell *d* in Abbildung 2.11 festgehalten.

Das ursprüngliche Kontrollflussmodell *cA* enthält keine Fehler und erfüllt zusammen mit dem Datenflussmodell *d* sowohl die Kontrollfluss- wie auch die Datenflussabhängigkeiten. Tauscht man nun die Reihenfolge der Aktivitäten *t2* und *t3*, so ergibt sich das Kontrollflussmodell *cB*. Für sich genommen ist *cB* korrekt und erfüllt die Kontrollflussabhängigkeiten. Das unveränderte *d* ist ebenfalls korrekt und erfüllt die Datenflussabhängigkeiten. Bei der Kombination zeigt sich aber, dass zwischen Kontroll- und Datenfluss ein Widerspruch besteht. Die Aktivität *t2* soll laut *cB* **nach** *t3* stattfinden, muss aber laut *d* **vor** *t3* ausgeführt werden.

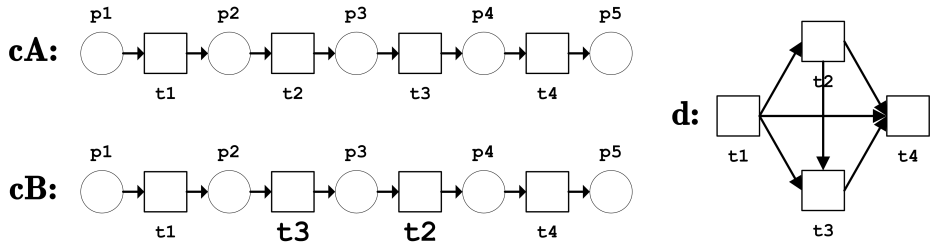


Abbildung 2.11: Beispiel: Widerspruchsfreie Flüsse ergeben widersprüchliche Kombination (nach Fan, Dou und Chen Fan et al. (2007))

Fan et al. bezeichnen diese Kombinationsproblematik als *control/data flow inconsistency* und eröffnen damit neben reinen Kontroll- und reinen Datenflussfehler noch eine dritte Kategorie. Zur Erfassung dieser aus der Kombination resultierenden Fehler schlagen sie die Verwendung verschiedener Tokens in dem den Workflow modellierenden Petri- bzw. Workflow-Netz vor. Damit kann der Kontrollfluss vom Datenfluss unterschieden und die Kombination der Flüsse auf Fehler hin untersucht werden.

Für die Umsetzung wählen Fan et al. komplexe Zahlen, bei denen der Realteil die Anzahl der Kontrollflusstokens in einem Platz wiedergibt und der Imaginärteil die Anzahl der Datenflusstokens. Zur benutzerfreundlichen graphischen Darstellung empfiehlt sich hingegen die Verwendung farbiger oder unterschiedlich geformter Tokens.

Trcka, van der Aalst und Sidorova (2008, 2009)

Trcka et al. beschreiben in zwei Artikeln (Trcka et al., 2009, 2008) einen Ansatz zur Prüfung der Kontroll- und teilweise auch der Datenflüsse in einem Workflow. Sie de-

finieren dafür zunächst Anforderungen an den Kontrollfluss und *Antipatterns* für den Datenfluss. Anschließend verwenden sie temporäre Logik (CTL*, bzw. CTL und LTL) für das Model Checking. Die Workflows werden dabei als Petrinetz-Variante namens „Workflow-Net“ (van der Aalst, 1998) modelliert.

Die Korrektheitseigenschaften von Trcka et al. (2008) sind in Tabelle 2.2 aufgelistet. Die in beiden Artikeln beschriebenen *Antipatterns* für den Datenfluss zeigt Tabelle 2.3.

Tabelle 2.2: Korrektheitseigenschaften aus Trcka et al. (2008)

No-dead-transitions	jede Aktivität ist erreichbar
Strong completion	alle Pfade führen immer zum Ende
Strong completion *with fairness*	alle Pfade führen *bei Fairness* zum Ende
Optional completion	aus jedem Zustand kann das Ende erreicht werden
Eventual completion *with* ϕ	es gibt einen Zustand ϕ, von dem aus das Ende erreicht werden kann
Eventual completion	es gibt einen Pfad vom Start zum Ende

Tabelle 2.3: Datenfluss-*Antipatterns* aus Trcka et al. (2008)

Missing data	Daten werden benötigt, sind aber nicht vorhanden
Strongly redundant d.	D. w. erzeugt, aber nie verwendet
Weakly redundant d.	wie *strongly*, aber nur auf manchen Pfaden
Strongly lost data	D. w. überschrieben, ohne gelesen worden zu sein
Weakly lost data	wie *strongly*, aber nur auf manchen Pfaden
Inconsistent data	D. w. parallel ohne Synchronisation verwendet
Never destroyed	erzeugte Daten werden nie gelöscht
Twice destroyed	erzeugte Daten werden mehrfach gelöscht
Not destroyed on time	erzeugte Daten werden nicht rechtzeitig gelöscht

Wie auch Fan et al. (2007) weisen Trcka et al. auf mögliche Abhängigkeiten zwischen Kontroll- und Datenfluss hin. Weil beispielsweise der Wert bestimmter Datenelemente die Wahl des Ausführungspfads bestimmen kann, ist es notwendig, beide Flussarten gemeinsam zu analysieren.

Umgesetzt wird die Analyse als Model Checking mit Petrinetzen, wobei die Antipatterns in temporaler Logik formuliert werden. Als Zwischenschritt werden dabei Kripke-Strukturen erzeugt. Bis auf diese Erzeugung verwenden Trcka et al. bereits bestehende Werkzeuge für die Analyse.

Meda, Sen und Bagchi (2007, 2010)

Das Verfahren von Sundari et al. (2007) greift die Einteilung von Sun et al. (2006) auf und unterscheidet hauptsächlich zwischen fehlenden, mehrfachen und verlorenen

Datenelementen. Neu ist die Einführung der Kategorie *redundant data in loops*, die das Problem des Überschreibens von Datenelementen bei der Ausführung von Schleifen betrifft. Weil das Verfahren abbricht, sobald es bei der Prüfung einen Fehler gefunden hat, muss zur Identifikation von n Fehlern im Workflow das Verfahren n-mal gestartet werden, ohne dass die vorher ermittelten Analysedaten weiterverwendet werden.

Meda et al. (2010) erweitern ihr bereits im Jahr 2007 vorgestelltes Verfahren „GTforDF“ (aus dem Zusammenhang geschlossen für „Graph Traversal for Data Flow“ stehend, die Autoren erklären die Abkürzung nicht) um die Erkennung des als *inconsistent data* bezeichneten Problems. Sie konzentrieren sich damit auf vier Arten von Datenflussschwachstellen:

- **Missing data**: Ein Datenelement wird zur Ausführung benötigt, liegt aber bei Erreichen der betroffenen Aktivität (noch) nicht vor.

- **Inconsistent data**: Eine Aktivität A erzeugt ein Datenelement D, welches von der Aktivität B in einem parallelen Ausführungspfad benötigt wird. Je nachdem, ob A vor B ausgeführt wird oder nicht, ist D vorhanden (bzw. aktualisiert) oder nicht, wenn B erreicht wird.

- **Lost data**: Die beiden parallel ausgeführten Aktivitäten A und B greifen schreibend auf das Datenelement D zu. Abhängig von der Ausführungsreihenfolge geht der von A oder von B geschriebene Wert verloren.

- **Redundant data**: Ein erzeugtes Datenelement wird weder innerhalb des Workflows verwendet noch dient es als Ausgabe des Workflows.

Die von Meda et al. gewählte Darstellung eines Workflows ist sehr eng an Petrinetze angelehnt und entspricht weitgehend der *Workflow Net*-Definition, wie sie van der Aalst und van Hee (2002) vorgestellt haben.

Das von Liu und Kumar (2005) beschriebene Prinzip der korrespondierenden Paare wird aufgegriffen. Die Definition wird für die Datenflussanalyse auf AND-Splits und AND-Joins beschränkt, wobei darauf hingewiesen wird, dass für die Kontrollflussanalyse auch XOR-Verzweigungen berücksichtigt werden müssen.

In der vorgestellten Variante ihres Analysealgorithmus werden zunächst die einzelnen Ausführungspfade generiert, also alle verschiedenen Pfade vom Start bis zum Ende des Workflows. Erst danach erfolgt die Analyse der Pfade. Meda et al. merken an, dass sich das Verfahren effizienter gestalten ließe, wenn die Analyse bereits beim Durchlaufen der Pfade geschehen würde. Eine Beschreibung oder grobe Skizze eines solchen effizienteren Ansatzes ist im Artikel jedoch nicht enthalten.

2.5 Eignung bestehender Verfahren zur Compliance-Prüfung

2.5.1 Zu erfassende Informationsfluss-Eigenschaften

Um den Informationsfluss eines Workflow-Modells auf Regeleinhaltung zu prüfen, müssen - den oben vorgestellten Arbeiten folgend - vier Eigenschaften analysiert werden.

- Korrektheit: der Workflow darf keine Deadlocks oder „Ausführungsreste“ verursachen
- Kontrollfluss: Modalität, Ort, Reihenfolge, Ressourcen und Zeitlimits müssen den Regeln entsprechen
- Datenfluss: Modalität, Ort, Ressourcen und Verwendung müssen den Regeln entsprechen
- Widerspruchsfreiheit der DF-KF-Kombination: Daten- und Kontrollfluss dürfen sich nicht gegenseitig behindern

Die Korrektheit von Workflow-Modellen wurde bereits im Jahr 1997 von Sadiq und Orlowska (1997) definiert. Weil die restlichen Eigenschaften nur dann sinnvoll geprüft werden können, wenn ein korrekter Workflow vorliegt, ist die Korrektheitsprüfung unabdingbar.

Die Analyse des Daten- und Kontrollflusses ist zur Prüfung der Einhaltung der oben diskutierten Regeln zwingend erforderlich. Wie die oben vorgestellten Arbeiten zeigen, wird allerdings von den meisten bestehenden Verfahren nur *entweder* der Datenfluss *oder* der Kontrollfluss geprüft, so dass bisher zur Analyse beider Flüsse unter Umständen zwei Verfahren eingesetzt werden müssen. Meda et al. (2010) sowie Trcka et al. (2008) beschreiben die Vorteile der gemeinsamen Prüfung und zeigen erste Schritte in Richtung eines Analyseverfahrens für beide Flüsse.

Die Widerspruchsfreiheit der DF-KF-Kombination wird vor allem von Fan et al. (2007) beschrieben und gefordert. Zwar wird auch von Meda et al. (2010) und Trcka et al. (2008) die gemeinsame Analyse vorgeschlagen, jedoch ohne speziell auf die Widerspruchsfreiheit einzugehen.

Nachfolgend werden die vier IF-Eigenschaften näher vorgestellt.

Prüfung auf Korrektheit des Workflows

Die Korrektheitsprüfung besteht typischerweise darin, ungültige Kombinationen von verzweigenden und vereinigenden Elementen zu suchen. Von Trcka et al. (2008) werden die Korrektheitsanforderungen in zwei Anforderungen zusammengefasst: *no-dead-transitions* und *completion*. Die erste Anforderung verlangt, dass jede Aktivität im

Workflow zur Ausführung gelangen kann, also vom Start des Workflows aus erreichbar ist. Die zweite Anforderung betrifft die Beendigung des Workflows und kann in verschiedenen Stärken formuliert werden.

Die stärkste Variante ist die Forderung, dass **alle** Pfade des Workflows zum Ende führen müssen und dass beim Erreichen des Workflow-Endes **keine** Aktivitäten mehr in der Ausführungsphase sein dürfen. Schwächere Varianten fordern nur, dass **zumindest ein** Pfad zum Workflow-Ende führen muss oder sogar nur führen **kann**, abhängig von bestimmten Bedingungen wie etwa der Ausführung anderer Aktivitäten. Bei einigen schwächeren Varianten dürfen sich außerdem beim Erreichen des Workflow-Endes noch manche Aktivitäten in der Ausführungsphase befinden.

Die für die Compliance-Prüfung relevanten Eigenschaften von Workflow-Modellen lassen sich nur dann sinnvoll prüfen, wenn der zu prüfende Workflow korrekt ist. Ein nicht korrekter, also beispielsweise Deadlocks enthaltender Workflow, führt zwar nach Erreichen des Deadlocks keine verbotenen Aktivitäten mehr aus, aber auch keine der geforderten Aktivitäten mehr. Die Korrektheit des Workflow-Modells ist daher auch für die Compliance-Prüfung relevant und wird durch zusätzliche Korrektheitsanforderungen abgedeckt.

Beispiel. Die Korrektheitsprüfung des Workflow-Modells in Abbildung 2.12 muss in Fehlermeldungen resultieren.

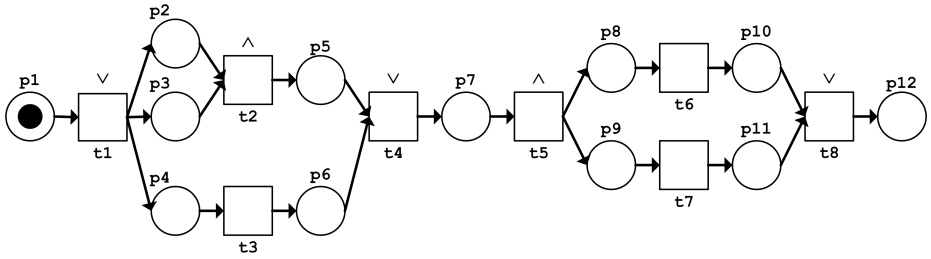


Abbildung 2.12: Beispiel für nicht korrekten Workflow

Die Erreichbarkeitsforderung wird im linken Abschnitt durch die auf die ODER-Transition *t1* folgende UND-Transition *t2* verletzt. Da *t1* nur genau ein Token erzeugt, kann niemals sowohl in *p2* als auch in *p3* ein Token liegen, so dass *t2* nicht erreichbar ist.

Die Fertigstellungsforderung wird in ihrer starken Variante im mittleren Abschnitt verletzt, da der obere Pfad nicht erreichbar ist (*p5* wird nie ein Token enthalten, da *t2* nie feuern kann). Die schwächere Variante, „es muss mindestens einen Pfad geben, der das Ende erreichen kann“, wird hingegen erfüllt, da *p6* ein Token erhält, wenn bei *t1* der untere Pfad gewählt wird.

Die Anforderung, dass bei Workflow-Ende alle Aktivitäten ebenfalls beendet wurden, wird im rechten Abschnitt verletzt, denn nach der parallel verzweigenden Transition *t5* kann sich die Transition *t6* noch in der Ausführung befinden während *t7* bereits

durchlaufen wurde (oder umgekehrt, so dass *t7* noch aktiv ist) wenn über die ODER-Transition *t8* das Workflow-Ende erreicht wird. ■

Trcka et al. (2008) stellen (wie in Abschnitt 2.4.2 vorgestellt) Anforderungen an die Prüfung des Kontroll- und Datenflusses in Workflows auf. Diese in Tabelle 2.4 gezeigten Anforderungen lassen sich durch Kombinationen aus den obigen Anforderungen abdecken, analog zu Tabelle 2.5.

Tabelle 2.4: Kontroll- und Datenflussanforderungen aus Trcka et al. (2008)

Korrektheit (Kontrollfluss)
no-dead-transitions, strong completion, strong completion with fairness, optional completion, eventual completion with ϕ, eventual completion
Datenfluss
missing, strongly redundant, weakly redundant, strongly lost, weakly lost, inconsistent, never destroyed, twice destroyed, not destroyed on time

Prüfung des Daten- und Kontrollflusses

Die oben vorgestellten Klassifikationen benennen die zentralen Elemente für die Prüfung von Workflows auf Regeleinhaltung. Aus Workflow-Sicht stehen typischerweise die Aktivitäten im Vordergrund (vgl. Modellierungssprachen wie BPMN und BPEL). Aus Regelsicht liegen stattdessen die Daten im Fokus (vgl. die zentrale Bedeutung geschützter Gesundheitsdaten in HIPAA). Beide Sichten sind für die Prüfung von Workflows bedeutend, da sowohl der durch die Abfolge von Aktivitäten bestimmte Kontrollfluss, als auch der durch die ausgetauschten Daten bestimmte Datenfluss analysiert werden müssen. Dieser im Abschnitt 1.3.3 diskutierten Beobachtung folgend werden die soeben klassifizierten Elemente entsprechend der Abbildung 2.13 eingeteilt.

Aktivitäten im Workflow müssen darauf geprüft werden, ob sie erlaubt oder sogar gefordert sind, am geforderten Ort und in der richtigen Reihenfolge stattfinden, nur von berechtigten Ressourcen (Rollen/Diensten) ausgeführt werden und ob aufgestellte Zeitlimits eingehalten werden.

Daten im Workflow müssen darauf geprüft werden, ob ihr Auftreten im Workflow erlaubt oder sogar gefordert ist und ob sie nur am geforderten Ort und nur von berechtigten Ressourcen (Rollen/Diensten) verwendet werden. Auch die Vorgaben bezüglich der Verwendung, also für welche Aktivitäten die Daten bestimmt sind, müssen eingehalten werden.

Aktivitäten verwenden Daten. In einem Workflow sind für praktisch jede Aktivität Daten nötig bzw. werden von ihr erzeugt. Auch in einem weitgehend manuell ausgeführten Workflow muss beispielsweise stets der Name des Patienten oder die Bestellnummer eines Kunden vorliegen.

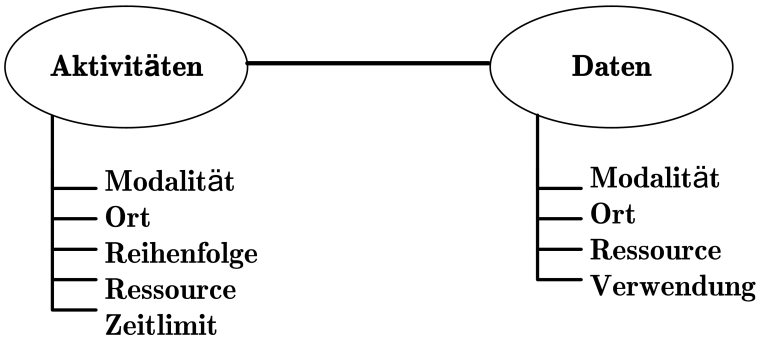


Abbildung 2.13: Elemente für die Prüfung des Daten- und Kontrollflusses

Daten müssen verarbeitet werden, benötigen also Aktivitäten. Ohne Aktivitäten, die Daten verwenden, sind Daten sinnlos. Selbst wenn Daten „nur“ gespeichert werden, um sie später zu verarbeiten, ist erstens die Speicherung selbst eine Aktivität und zweitens stellt auch die spätere Verwendung eine Aktivität dar.

Die **Modalität** beinhaltet zunächst die grundlegende Aussage, ob ein Element im Workflow enthalten sein darf oder nicht. Darüberhinaus legt sie auch fest, ob das Element im Workflow enthalten sein *muss*.

Der **Ort** der Ausführung von Aktivitäten oder der Verwendung von Daten ist besonders vor dem Hintergrund der aktuellen Entwicklungen im sogenannten *Cloud Computing* wichtig (vgl. Analyse von Accorsi et al. (2011)). Für unternehmensübergreifende, aber auch schon für unternehmensinterne Workflows, Teile derer ausgelagert werden, muss nachvollziehbar sein, an welchem (Stand-) Ort die Ausführung von Aktivitäten oder Verwendung von Daten erfolgt. Die Firma Microsoft beispielsweise speichert momentan (Dezember 2010, Details einer Firmenpräsentation) einen Großteil der über die Azure-Plattform abgelegten Daten in Amsterdam. Bei einer umfassenden Compliance-Prüfung müssen daher die dort geltenden rechtlichen Datenschutzbestimmungen mit denen des Herkunftslandes verglichen werden. Sind sie inkompatibel, muss ggf. eine andere Plattform zum Einsatz kommen.

Die **Reihenfolge** von Aktivitäten spielt besonders im Gesundheitsbereich eine große Rolle. Nicht nur aufgrund medizinischer Notwendigkeiten, sondern auch wegen expliziter Vorgaben (z.B. in HIPAA), wann beispielsweise ein Patient über mögliche Operationsrisiken etc. aufzuklären ist, muss geprüft werden, ob alle vor der Ausführung einer Aktivität nötigen Aktivitäten im Workflow an der richtigen Stelle stehen und ob alle nach der Ausführung einer Aktivität vorgeschriebenen Aktivitäten modelliert wurden. Vielfach erfolgt die Angabe der Reihenfolge zusammen mit einem Zeitlimit. Dass auf eine Patientenanfrage innerhalb von 30 Tagen eine Reaktion erfolgen muss, ist ein Beispiel aus HIPAA.

Die **Ressource**, also im Falle menschlicher Ausführung die Rolle, im Falle maschineller Ausführung der Dienst, muss bekannt sein um die klassische Zugriffskontrolle umsetzen zu können. Hat die Rolle oder der Dienst im Workflow-Modell nicht das

Recht, bestimmte Aktivitäten auszuführen oder bestimmte Daten zu verwenden, muss das bei der Prüfung entdeckt werden.

Die **Verwendung** von Daten kann vorgeschrieben werden, was häufig in Kombination mit einem Zeitlimit geschieht. So kann für Verbindungsdaten eines Telekommunikationsunternehmens vorgeschrieben sein, dass diese nicht vor Ablauf einer gewissen Frist gelöscht werden dürfen oder nach Ablauf eines Zeitlimits gelöscht werden müssen.

Das **Zeitlimit** erlaubt, anhand von absoluten oder relativen Zeitpunkten festzulegen, innerhalb welcher Zeit eine Aktivität ausgeführt werden muss. Ebenso lässt sich die Ausführung für einen gewissen Zeitraum verbieten. Die Patientenbenachrichtigung innerhalb von 30 Tagen oder die Löschung von Telefonverbindungsdaten nach 90 Tagen sind Beispiele für solche Fristen.

Die in manchen Regularien unterschiedenen Begriffe der Erhebung, Verarbeitung, Speicherung und Löschung von Daten werden in dieser Arbeit unter dem Begriff **Verwendung** zusammengefasst. Tatsächlich lässt sich auf der Workflow-Modellebene nicht prüfen, ob beispielsweise die Löschung eines Datensatzes sachgemäß erfolgt. Dazu müsste das gesamte IT-System inklusive der verwendeten Soft- und Hardware geprüft werden (Lowis und Accorsi (2011) beschreiben ein für SOA-basierte Workflows geeignetes Verfahren). Bei der in dieser Arbeit verfolgten Prüfung auf Modellebene wird geprüft, ob im Workflow die von Compliance-Regeln geforderten Aktivitäten modelliert sind. Finden sich im Modell also einerseits nur erlaubte, andererseits alle geforderten Aktivitäten, bedeutet das - auf Modellebene - die Einhaltung der Regeln zur Datenverwendung, unabhängig davon ob es sich um datenerhebende, verarbeitende, speichernde oder löschende Aktivitäten handelt.

In einigen Regularien wird der Zweck festgelegt, zu dem Daten erhoben, verarbeitet und gespeichert werden dürfen. Dieser Zweck ist wegen der nötigen semantischen Interpretation für eine automatisierte Analyse nicht direkt erfassbar. In der vorliegenden Arbeit wird davon ausgegangen, dass der Zweck durch die Abfolge der vorhergehenden und nachfolgenden Aktivitäten formalisiert werden kann. Demnach wird der Zweck nicht als eigenständiges Attribut angegeben, sondern durch die Angabe erlaubter oder verbotener Aktivitäten abgedeckt.

Vielfach finden sich in Regularien auch Bedingungen. HIPAA etwa unterscheidet bezüglich der Gesundheitsdaten den normalen Behandlungsfall vom Notfall und erlaubt, im Notfall auch ohne Einwilligung des (bewusstlosen) Patienten Gesundheitsdaten zu verarbeiten. Als Zusatzbedingung muss der Patient nach Wiedererlangung des Bewusstseins über die Verarbeitung informiert werden. Eine solche Bedingung erfordert, ähnlich wie der Zweck, eine semantische Interpretation, so dass ihr Zutreffen im allgemeinen Fall nicht automatisiert entschieden werden kann. Bedingungen werden deswegen im Folgenden als Prädikate angesehen, so dass bei der Workflow-Analyse beispielsweise der Notfallzustand aus einer booleschen Variable ausgelesen werden kann. Für die durch die erfüllte Bedingung ausgelösten Aktivitäten und Daten gelten dieselben Attribute wie in Abbildung 2.13.

Die obige Einteilung beinhaltet die von Sadiq et al. (2007) vorgeschlagenen Aspek-

te. Die COMPAS-Klassifikation (Tilbug University, 2008) enthält den Aspekt *locative compliance concern*, der durch das Ort-Attribut abgedeckt wird.

Die sieben Workflow-Elemente der Ontologie von Breaux (2009) lassen sich wie folgt den Elementen und Attributen zuordnen: *Subject* zu Ressourcen (Rolle), *Act* zu Aktivitäten, *Object* zu Daten, *Purpose* als Zweck wie unten diskutiert, *Instrument* zu Ressourcen (Dienst), *Location* zu Ort und *Ressource* zu Daten.

Die von Giblin et al. (2005, 2006) vorgestellten Forderungen (Details dazu siehe Abschnitt 2.4.1) bezüglich der Reihenfolge von Aktivitäten und der Einhaltung von Fristen wird mit den Anforderungen abgedeckt.

Von Sadiq et al. (2004) werden sieben mögliche Datenfluss-Probleme diskutiert, die bei der Validierung eines Workflow-Modells entdeckt werden sollten. Wie die (in Abschnitt 2.4 näher beschriebenen) Probleme durch die obigen Anforderungen abgedeckt werden, zeigt die Tabelle 2.5.

Tabelle 2.5: Zuordnung der Datenfluss-Probleme aus Sadiq et al. (2004)

Problem aus Sadiq et al. (2004)	Abdeckung durch...
Redundant data	Verwendung
Lost data	Reihenfolge, Verwendung
Missing data	Modalität
Mismatched data	Verwendung
Inconsistent data	Reihenfolge, Verwendung
Misdirected data	Verwendung
Insufficient data	Modalität

Einige Beispiele für die oben genannten Workflow-Eigenschaften schließen diesen Abschnitt ab.

Beispiel. Bei der Prüfung des in Abbildung 2.14 dargestellten Workflows gegen die folgenden Regeln muss ein Prüfverfahren Warnungen hinsichtlich der Aktivitäten ausgeben.

```
Regel 1. Erlaubte Aktivitäten: 'Inform patient',
         'Examine patient', 'Update patient record'.
Regel 2. Nach 'Examine patient' muss 'Update
         patient record' ausgeführt werden.
Regel 3. Vor 'Examine patient' muss 'Inform
         patient' ausgeführt werden.
```

Laut Regel 1 muss die Aktivität *Send advertisement* als verboten erkannt werden. Da *Examine patient* im Modell steht, muss für Regel 2 das fehlende *Update patient record* gemeldet werden. Entsprechend Regel 3 muss die falsche Reihenfolge von *Examine patient* und *Inform patient* entdeckt werden. ■

Abbildung 2.14: Beispiel-Workflow mit Aktivitäten, ohne Daten-Angaben

Beispiel. Bei der Prüfung des in Abbildung 2.15 dargestellten Workflows gegen die folgenden Regeln muss ein Prüfverfahren Warnungen hinsichtlich der Daten ausgeben.

```
Regel 3. Erlaubte Daten: Patient ID, ECG data, Examination
         results, Patient remarks, Blood values.
Regel 4. 'Take blood sample' erfordert Patient ID.
Regel 5. Für 'Create diagnosis' werden Patient ID, ECG data,
         Blood values und Examination results gefordert.
```

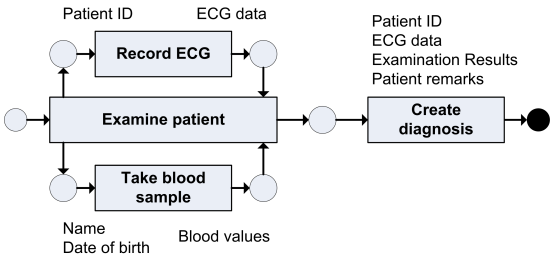


Abbildung 2.15: Beispiel-Workflow mit Daten-Angaben

Regel 3 bedingt die Markierung des Namens und Geburtsdatums als unzulässige Datenelemente. Regel 4 verlangt, das fehlende Datenelement Patient ID bei *Take blood sample* zu bemerken. Aufgrund von Regel 5 muss ein Hinweis auf das bei *Create diagnosis* fehlende Datenelement der Blutwerte erfolgen. Das Prüfverfahren muss erkennen können, dass das Workflow-Modell in Abbildung 2.15 keinen eindeutigen Datenfluss erzeugt, so ist z.B. unklar, ob zuerst die EKG-Daten, die Blutwerte oder keine dieser Daten zu *Examine patient* gelangt. Entsprechend Regel 5 muss bei *Create diagnosis* erkannt werden, dass die „patient remarks“ kein erforderliches Datenelement sind. Die Nicht-Verwendung der Blutwerte muss erkannt werden; dieses Datenelement dient keiner späteren Aktivität als Eingabe. ■

Beispiel. Bei der Prüfung des in Abbildung 2.16 dargestellten Workflows gegen die folgenden Regeln muss ein Prüfverfahren Warnungen hinsichtlich der Ressourcen ausgeben.

```
Regel 6. Erlaubte Rolle: Doctor.
         Erlaubte Orte: Doctor's office, Examination room.
Regel 7. 'Examine patient' muss im Examination room stattfinden.
```

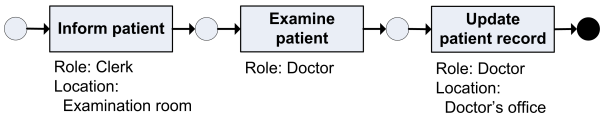


Abbildung 2.16: Beispiel-Workflow mit Ressourcen-Angaben

Der Regel 6 folgend muss die unzulässige Ressource bemängelt werden. Gemäß Regel 7 ist die fehlende Ortsangabe bei *Examine patient* zu bemerken. ■

Beispiel. Bei der Prüfung des in Abbildung 2.17 dargestellten Workflows gegen die folgenden Regeln muss ein Prüfverfahren Warnungen hinsichtlich der Zeitlimits ausgeben.

```
Regel 8. Mindestens 5 Tage vor der Untersuchung muss
         der Patient informiert werden.
Regel 9. Spätestens 3 Tage nach der Untersuchung muss
         der Patient an die Nachuntersuchung erinnert werden.
```

Abbildung 2.17: Beispiel-Workflow mit Zeitlimits

Zur Erfüllung der Zeitlimit-Anforderungen müssen Kontrollaktivitäten für Fristen im Workflow modelliert sein und an der passenden Stelle gefunden werden. Vorgeschriebene Start- und Endzeitpunkte, wie sie in Regel 8 und 9 für die Aktivität *Examine patient* festgelegt sind, müssen so kontrolliert werden. Das Fehlen von Kontrollaktivitäten im Workflow der Abbildung 2.17 ist zu melden. ■

Prüfung auf Widerspruchsfreiheit der DF-KF-Kombination

An einem Workflow mit nur vier Aktivitäten wird von Fan et al. (2007) gezeigt, dass die Kombination aus Datenfluss und Kontrollfluss selbst dann nicht widerspruchsfrei sein muss, wenn die Flüsse für sich genommen korrekt sind. Ein typisches Beispiel ist die Verfügbarkeit von Daten, die zur Ausführung einer bestimmten Aktivität benötigt werden. Liegen die Daten zum Zeitpunkt des Aufrufs der Aktivität noch nicht vor, kann die Aktivität nicht ausgeführt werden. An der Prüfung der einzelnen Flüsse lässt sich das aber nur bedingt bzw. anhand zusätzlich aufgestellter Regeln erkennen. Eine Korrektheitsprüfung des Datenflusses und des Kontrollflusses, jeweils für sich genommen, zeigt solche Widersprüche nicht an.

Das Fallbeispiel der Behandlung im Krankenhaus aufgreifend würde der Workflow-Ausschnitt A in Abbildung 2.18 einen Widerspruch auslösen. Der Kontrollfluss des

Abschnitts ist für sich genommen korrekt, da weder Deadlocks auftreten noch Aktivitäten nach dem Durchlauf aktiv bleiben. Der Datenfluss ist ebenfalls fehlerfrei, da beide Aktivitäten Zugriff auf die EPA haben dürfen und die zusätzlichen Vorgaben bezüglich der zugreifenden Rolle und des Speicherorts eingehalten werden. In der Kombination aus Kontroll- und Datenfluss zeigt sich aber, dass die Aktivität *EPA abzeichnen* nicht ausgeführt werden kann, wenn vorher kein Eintrag gemacht wurde. In der Variante des Workflow-Ausschnitts B besteht dieses Problem nicht.

Als zusätzliche Regel muss dann angegeben werden, dass *Eintrag in EPA* **vor** *EPA abzeichnen* ausgeführt werden muss. Alternativ kann der Widerspruch durch die Analyse der Ein- und Ausgabedaten von Aktivitäten aufgedeckt werden, was eben eine kombinierte Analyse des Daten- und Kontrollflusses erfordert.

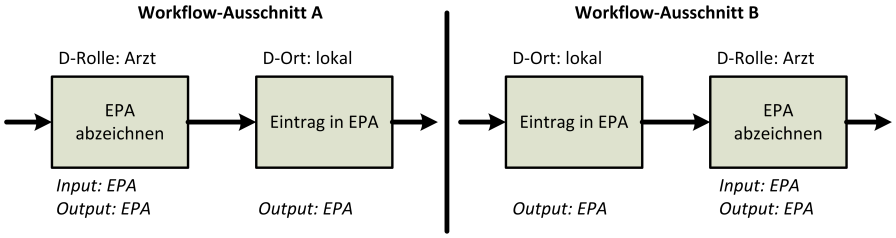


Abbildung 2.18: Beispiel für Widerspruch zwischen DF und KF

Über die grundlegende Korrektheitsprüfung und die jeweils einzelne Prüfung des Daten- und Kontrollflusses hinaus muss demnach eine Prüfung auf Widerspruchsfreiheit der Kombination aus DF und KF erfolgen.

Zusammenfassung der IF-Eigenschaften

Eine Übersicht über die insgesamt zu erkennenden Workflow-Eigenschaften gibt die Tabelle 2.6. Der Kürze halber wird in späteren Kapiteln auf die einzelnen Anforderungen durch eine Element-Kürzel-Kombination Bezug genommen, so z.B. mit *D.O* für Ortsangaben bei Daten.

2.5.2 Einordnung der bisherigen Verfahren

Es stellt sich nun die Frage, ob die vorgestellten Verfahren ausreichend sind, um alle in Abschnitt 2.5.1 zusammengefassten IF-Eigenschaften zu analysieren.

Giblin et al. stellen mit REALM ein auf temporaler Logik basierendes Verfahren zur Workflow-Prüfung vor. Das Verfahren läuft ständig, eben als Monitor, und entdeckt dabei die auftretenden Probleme. Schwachstellen, die nicht bei der überwachten Ausführung auftreten, bleiben unentdeckt und „schlummern“ weiterhin im Workflow. Wegen der mehreren zugrundeliegenden logischen Formalismen (temporale Logik und Prädikatenlogik höherer Stufe) ist die Komplexität der Verarbeitungsschritte vergleichsweise

Tabelle 2.6: Zu prüfende Workflow-Eigenschaften

Element	Kürzel	Einhaltung der geforderten...
	K	Korrektheit
A, D	M	Modalität
A, D	O	Ortsangaben
A	R	Reihenfolge von Aktivitäten
A, D	S	Ressourcen (Rolle/Dienst)
D	V	Verwendung von Daten
A	Z	Zeitlimits (durch Kontrollaktivitäten)
	W	Widerspruchsfreiheit der DF-KF-Kombination

A=Aktivität, D=Daten

hoch, ebenso der Zeitaufwand für Einarbeitung, Bedienung und Auswertung. Das Verfahren zielt hauptsächlich auf die Erkennung von Reihenfolgekonflikten ab, also die falsche Reihenfolge von Aktivitäten und die Nicht-Einhaltung spezieller Vorgaben für die Ausführungszeitpunkte. Der Datenfluss wird nicht analysiert.

Liu, Müller und Xu verwenden nicht nur einen, sondern gleich mehrere Formalismen für ihren auf dem Model Checking basierenden Ansatz. Workflows in Form von BPEL-Prozessen werden dabei zunächst in den Pi-Calculus und dann in endliche Automaten transformiert; die Regeln müssen im Format der Business Property Specification Language (BPSL) vorliegen und werden in lineare temporale Logik (LTL) transformiert. Wie (Liu et al., 2007, S. 339) selbst schreiben, ist temporale Logik ohne Hintergrundwissen nur schwer verständlich. Ihr in Abbildung 2.19 gezeigtes Framework enthält daher insgesamt sechs Adapter, Transformer und Tracer, die zwischen den verschiedenen Formaten übersetzen und die Ergebnisse der formalen Prüfung auf das ursprüngliche Workflow-Modell zurückführen. Entsprechend hoch ist die Einarbeitungs-, Bedien- und Auswertungszeit, und entsprechend viele Schritte hoher Komplexität sind zur Verarbeitung nötig. Der Fokus des Ansatzes liegt auf dem Kontrollfluss, besonders der Prüfung der Reihenfolge von Aktivitäten und der Einhaltung von zeitlichen Vorgaben. Der Datenfluss steht im Hintergrund.

Adam, Atluri und Huang stellen mit ihrem auf speziellen Petrinetzen mit zeitlichen Beschränkungen beruhenden Ansatz ein Verfahren vor, welches ebenfalls den Fokus auf die Reihenfolgeprüfung legt. Die von Rusinkiewicz und Sheth (1995) definierten drei Arten von Abhängigkeiten innerhalb eines Workflows werden zwar aufgegriffen, laut der Beschreibung erfasst das Analyseverfahren aber nur die Ausführungsabhängigkeiten sowie die zeitlichen Abhängigkeiten. Darüberhinaus wird nur auf Korrektheitseigenschaften geprüft, nicht auf Eigenschaften des Datenflusses.

Sun, Zhao, Nunamaker und Sheng setzen ein erfolgreich auf Korrektheit geprüftes Workflow-Modell voraus und bieten ein Prüfverfahren für den Datenfluss innerhalb dieses Modells. Das Verfahren setzt eine in der Praxis schwer aufzustellende vollständige Lese-/Schreib-Datenflussmatrix voraus, die für jedes Datenelement definiert, welche

Tabelle 2.7: Einordnung bestehender Workflow-Analyseverfahren

Eigenschaft / **Verfahren**	**K**	**DF**				**KF**					**W**
		M	O	S	V	M	O	R	S	Z	
Giblin et al.								✓		✓	
Liu et al.								✓		✓	
Sadiq et al.	✓										
Adam et al.						✓		✓		✓	
Fan et al.	✓							✓			✓
Meda et al.		✓			✓						✓
Sun et al.		✓			✓						✓
Trcka et al.	✓	✓			✓						✓
Van der Aalst et al.	✓										
Vanhatalo et al.	✓							✓			

K=Korrektheit,DF=Daten-,KF=Kontrollfluss,W=Widerspruchsfreiheit. M=Modalität,O=Orte,R=Reihenfolge,S=Ressourcen, V=Verwendung,Z=Zeitlimits

Aktivität schreibend und/oder lesend darauf zugreift. Der Kontrollfluss wird nicht analysiert.

Vanhatalo, Völzer und Leymann prüfen mit ihrem Verfahren „nur“ die Korrektheit eines Workflows. Durch die Einteilung eines Petrinetzes in *single entry, single exit (SESE)* genannte Fragmente (ähnlich der korrespondierenden Paare) wird diese Prüfung stark beschleunigt. Als Ergebnis identifiziert das Verfahren problematische Fragmente. Die jeweils auslösende Aktivität muss innerhalb eines solches Fragments manuell ermittelt werden. Der Kontrollfluss in Bezug auf einzelne Aktivitäten sowie der gesamte Datenfluss werden von diesem Verfahren nicht geprüft.

Fan, Dou und Chen beschreiben eines der bisher wenigen Verfahren, die neben dem Kontrollfluss teilweise auch den Datenfluss prüfen. Allerdings bricht das Verfahren bei der Entdeckung einer Schwachstelle ab, muss also zur Entdeckung von n Schwachstellen n-mal aufgerufen werden. Problematischer ist, dass die (Korrektheits-) Prüfung nicht genau die Stelle im Workflow lokalisiert, die eine Schwachstelle darstellt, sondern nur rückmeldet, dass eine Schwachstelle gefunden wurde. Bezüglich der Aktivitäten und Daten innerhalb eines Workflows prüft das Verfahren nur auf die vorgeschriebene Reihenfolge und erfasst damit nur einen Bruchteil der Daten- und Kontrollflusseigenschaften.

Trcka, van der Aalst und Sidorova setzen für ihr Prüfverfahren temporale Logik ein, und zwar CTL* (sowohl CTL als auch LTL). Der damit verbundene Aufwand und die Komplexität wurden bereits diskutiert. Die geprüften Workflow-Eigenschaften sind in den Tabellen 2.2 und 2.3 aufgelistet und beziehen sich auf die Korrektheit und den Datenfluss.

Meda et al. beschreiben ein Datenfluss-Analyseverfahren und heben hervor, dass

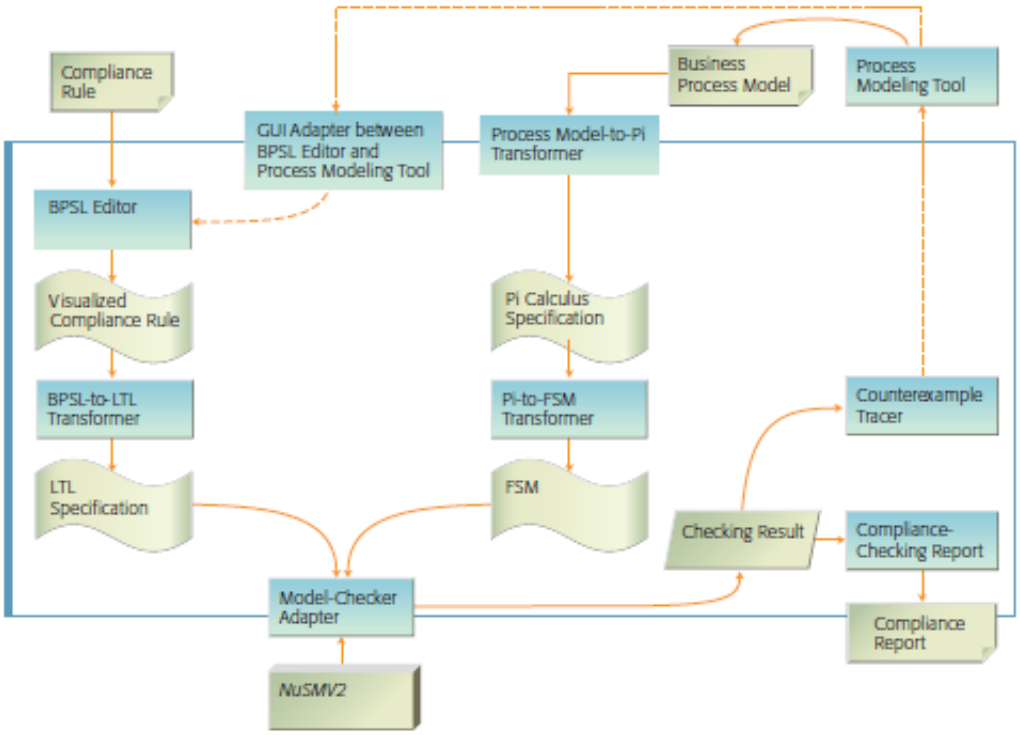


Abbildung 2.19: *Compliance-checking Framework* von Liu, Müller und Xu, Abbildung aus (Liu et al., 2007, S. 350)

ihr Verfahren zur kombinierten Kontrollfluss- und Datenfluss-Analyse erweitern werden kann. Hauptsächlich zielt das Verfahren auf die Entdeckung von Schwachstellen im Datenfluss. Wird eine solche gefunden, bricht die Analyse ab und gibt eine Warnung aus. Zur Aufdeckung mehrerer Schwachstellen muss das Verfahren daher mehrfach gestartet werden. In der beschriebenen Form muss jeweils die gefundene Schwachstelle behoben werden, bevor weitere Schwachstellen entdeckt werden können. Die präzise Lokalisierung der potentiell regelwidrigen Aktivitäten muss manuell durchgeführt werden. Über die Erwähnung der Wichtigkeit der Kontrollfluss-Analyse hinaus beschreiben die Autoren aber nicht, wie die entsprechende Erweiterung ihres Verfahrens aussehen könnte. Der Kontrollfluss wird durch ihr Verfahren nicht geprüft.

Während die Korrektheit von Workflows von vielen Verfahren untersucht wird, liegt der Schwerpunkt ansonsten entweder auf dem Datenfluss *oder* auf dem Kontrollfluss, also z.B. der Reihenfolge von Aktivitäten. Der **Ort** der Ausführung und die ausführende bzw. zugreifende **Ressource**, also die Rolle oder der Dienst, wird von keinem Verfahren explizit berücksichtigt.

Aus der Sicht der drei klassischen Schutzziele der Sicherheit (Müller und Pfitzmann, 1999), nämlich Vertraulichkeit, Integrität und Verfügbarkeit, ergibt sich das folgende

Bild. Die Verfügbarkeit von Daten wird von vielen Verfahren geprüft. Beispielsweise suchen Sun et al. (2006) nach „missing data“. Ebenso wird die Verfügbarkeit der einzelnen Aktivitäten geprüft, indem potentielle Deadlocks gesucht werden. Die Integrität von Daten wird von wenigen Verfahren geprüft. Meda et al. (2010) etwa suchen nach „inconsistent data“, also Daten, die möglicherweise durch andere überschrieben wurden.

Die Vertraulichkeit von Daten hingegen wird von keinem der Verfahren geprüft. Eine mögliche Erklärung ist die Tendenz, dass Vertraulichkeit in der Praxis typischerweise durch Zugriffskontrolle während der Laufzeit angestrebt wird und nicht bereits zur Entwurfszeit im Modell untersucht wird. Eine andere mögliche Erklärung liegt darin, dass die Verfahren meist aus Korrektheits-Analysen erwachsen sind und eher allgemeingültige Regeln wie „jede Aktivität muss erreichbar sein“ einsetzen als zu parametrisierende Regeln wie „Aktivität X darf nicht auf Datum Y zugreifen“.

Zur Einhaltung von Compliance-Regeln gehört die Prüfung auf Vertraulichkeit von Daten aber insofern zwingend dazu, als viele Regularien entsprechende Vorschriften für den Umgang mit (persönlichen) Daten enthalten. Die von den oben beschriebenen Verfahren gebotene Prüfung des Workflow-Informationsflusses bzw. des Daten- und Kontrollflusses auf Korrektheit, Verfügbarkeit und teilweise auch Integrität ist somit nicht ausreichend und muss um die Prüfung auf Vertraulichkeit ergänzt werden.

3 Workflow-Analyse mit REVEAL

Nach einer kurzen Übersicht des Prüfungsablaufs (Abschnitt 3.1) stellt dieses Kapitel Petrinetze als Grundlage der Prüfung bzw. zur Modellierung von Workflows und Regeln vor (Abschnitt 3.2). Darauf aufbauend werden Prüfungsvorlagen definiert (Abschnitt 3.3), anhand derer sich Compliance-Regeln für Workflows formulieren lassen. Der Algorithmus zur eigentlichen Prüfung wird im Abschnitt 3.4 beschrieben, gefolgt von einer Übersicht über die Implementierung.

3.1 Ablauf

REVEAL prüft einen als Petrinetz vorliegenden Workflow auf die Einhaltung von ebenfalls als Petrinetz vorliegenden Regeln. Der erste Schritt zur Prüfung ist daher einerseits die Petrinetz-Modellierung des zu prüfenden Workflows, andererseits die Petrinetz-Modellierung der für diesen Workflow geltenden Regeln, wie in Abbildung 3.1 dargestellt.

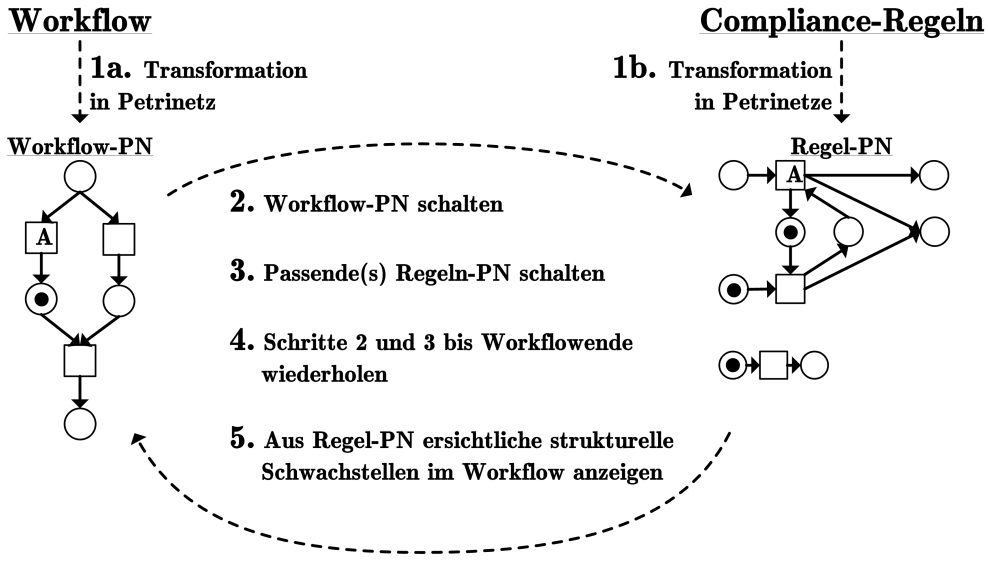


Abbildung 3.1: Ablaufübersicht des REVEAL-Verfahrens

Die Regeln werden anhand von Vorlagen (Beschreibung im Abschnitt 3.3) formalisiert, die alle im Abschnitt 2.5 bereits vorgestellten Regelelemente abdecken, nämlich

Aktivitäten mit den Zusatzangaben Modalität, Ort, Reihenfolge, Ressource, Zeitlimit, und Daten mit den Zusatzangaben Modalität, Ort, Ressource, Verwendung.

Liegen Workflow und Regeln als Petrinetze vor, beginnt der Prüfalgorithmus. Jeder Zweig des Workflows wird vom Start bis zum Ende durchlaufen und anhand der Prüfungsvorlagen auf Einhaltung der Regeln untersucht. Für jede Aktivität im Workflow werden dazu die passenden Vorlagen geschaltet. Eine Workflow-Aktivität - wie z.B. in Abbildung 3.1 als Transition *A* modelliert - löst dementsprechend die Schaltung genau der Prüfungsvorlagen aus, die eine passende Petrinetz-Transition *A* enthalten.

Wenn die Analyse das Ende des Workflows erreicht hat, lassen sich Verstöße gegen die durch die Vorlagen formulierten Regeln an den Zuständen der Regel-Petrinetze ablesen. Mit *OK* markierte Zustände signalisieren Regeleinhaltung, mit *ERR* oder anderen Fehlerhinweisen, wie etwa *WrongOrder* für eine falsche Reihenfolge, markierte Zustände weisen auf strukturelle Schwachstellen im Workflow hin.

Für jede Schwachstelle werden die genaue Position im Workflow und die Umstände des möglichen Verstoßes protokolliert. Darüber hinaus kann Reveal aus einigen Regelvorlagen Behebungsvorschläge generieren, die dazu genutzt werden können, die strukturellen Schwachstellen des Workflows zu beheben. Wird etwa in einem Zweig des Workflows eine Aktivität erst *nach* anstatt *vor* einer bestimmten anderen Aktivität ausgeführt, zeigt Reveal diesen Umstand inklusive der Namen der beiden Aktivitäten auf.

3.2 Transformation von Regeln und Workflows in Petrinetze

Hauptvoraussetzung für die Analyse mit Reveal ist die Modellierung von Regeln und Workflows durch Petrinetze. In diesem Abschnitt wird die automatisierte Transformation der XML-basiert vorliegenden Regeln und Workflows in Petrinetze diskutiert. Es ist also vor der Anwendung von Reveal nicht nötig, manuell entsprechende Petrinetze für die Regeln und die Workflows zu erzeugen. Die Umsetzung der Transformations-Werkzeuge wird im Abschnitt 3.5 beschrieben.

3.2.1 Regel-Transformation

Um die in Abschnitt 2.5 beschriebenen Workflow-Eigenschaften als Regel zu instanziieren, müssen die Vorgaben bezüglich der Aktivitäten und Daten eines Workflow-Modells durch eindeutige Schlüsselworte formuliert werden. Größtenteils genügt es, wenn neben den jeweiligen Ressourcen und Orten auch die Modalität sowie die Reihenfolge angegeben werden kann.

Zusätzlich muss für Daten angegeben werden können, ob sie vor bzw. nach der aktuellen Aktivität einer anderen Verwendung zugeführt werden müssen. Damit können Forderungen wie „Filterung vor der Weiterverarbeitung“ und „Löschung nach der Ver-

wendung“ umgesetzt werden. Die Löschung wird damit als Spezialfall einer Aktivität bzw. Verwendung berücksichtigt, so dass eine Regel wie „Löschung vor dem Workflow-Ende“ abgedeckt werden kann.

Die vierte nötige Angabe außer der Modalität, der Reihenfolge und der Datenverwendung betrifft die zeitlichen Fristen. Damit wird formuliert, ab bzw. bis wann eine Aktivität minimal oder maximal vor bzw. nach einer anderen Aktivität ausgeführt werden muss.

Die Tabelle 3.1 zeigt die möglichen Ausprägungen der REVEAL-Regelangaben zu den in Abschnitt 2.5 ermittelten und in Tabelle 2.6 zusammengefassten Regeln.

Tabelle 3.1: Ausprägungen der REVEAL Regelangaben

Angabe	Ausprägungen
Modalität	X (darf nicht, muss)
Ort	X (darf nicht, muss)
Reihenfolge	X (vor, nach) Y
Ressource	X (darf nicht, muss)
Verwendung	X (vorher, nachher) Y
Zeitlimit	X (ab, bis) (minimal, maximal) Frist (vor, nach) Y

Jede Angabe wird anhand eines kurzen Beispiels verdeutlicht.

- A.M (Modalität für Aktivität): Workflow darf nicht „Patientendaten löschen“ enthalten.
- D.O (Ort für Daten): Patientenakte muss lokal gespeichert werden.
- A.R (Reihenfolge für Aktivitäten): Aufklärung muss vor Operation erfolgen.
- D.S (Ressource für Daten): Pfleger darf auf Rezept nicht zugreifen.
- D.V (Verwendung für Daten): Patientendaten dürfen nicht für Werbesendung verwendet werden.
- A.Z (Zeitlimit für Aktivitäten): Auf Patientenanfragen muss innerhalb von 30 Tagen eine Antwort erfolgen.

Um sicherzustellen, dass der modellierte Datenfluss korrekt ist, muss das Prüfverfahren das Workflow-Modell auf parallel ausgeführte Aktivitäten prüfen, die potentiell gleichzeitig auf ein Datenelement zugreifen oder in verschiedenen Pfaden gleichnamige Datenelemente erzeugen, von denen bei der Zusammenführung der Pfade alle bis auf eines verloren gehen. Die Korrektheitseigenschaften beziehen sich nicht auf einzelne Elemente eines Workflows, sondern auf das komplette Workflow-Modell. Sie werden daher nicht als einzelne Angaben formuliert, sondern für den ganzen Workflow geprüft.

Typischerweise sind Korrektheitsforderungen nicht expliziter Teil der Compliance-Regularien. Da bei der Prüfung eines nicht korrekten Workflows in vielen Fällen aber die restlichen Regeln nur teilweise geprüft werden können bzw. irreführende Ergebnisse verursachen, wie das folgende Beispiel zeigt, ist ein korrekter Workflow unabdingbar für die Analyse. Die Korrektheitsprüfung sollte daher Bestandteil jedes Verfahrens sein oder aber als Voraussetzung explizit angegeben werden. REVEAL prüft standardmäßig auf Korrektheit.

Beispiel. Für den Workflow in Abbildung 3.2 sei die Regel aufgestellt worden, dass auf jedem Pfad die Aktivität *DEL* vor Workflow-Ende ausgeführt werden muss. Da der Workflow aber eine ODER-Verzweigung enthält, auf die eine UND-Vereinigung folgt und damit einen Deadlock auslöst, wird der Workflow in zwei von drei Fällen bzw. Pfaden nicht zu Ende ausgeführt.

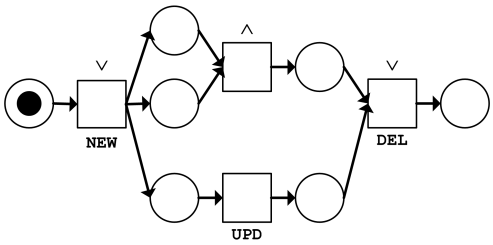


Abbildung 3.2: Beispiel für ein nicht korrektes Workflow-Modell

So enthält zwar jeder Pfad die geforderte *DEL*-Aktivität, tatsächlich wird sie aber nie ausgeführt, wenn der obere oder mittlere Pfad gewählt wird, da dort ein Deadlock die Ausführung aufhält. ■

Einige Regeln müssen mit Ausnahmen versehen werden können. Im Gesundheitsbereich etwa werden häufig Sonderregelungen für den Notfall getroffen, damit für eine Behandlung ausnahmsweise Daten des Patienten verwendet werden können, ohne dass die Einwilligung des - möglicherweise bewusstlosen - Patienten vorliegt. Für solche Ausnahmen können zusätzlich zu den üblichen Angaben Bedingungen formuliert werden, unter denen eine Regel nicht oder in anderer Ausprägung gilt. REVEAL kann, da es als zur Entwurfszeit prüfendes Verfahren nur das Workflow-Modell analysiert, im allgemeinen Fall nicht feststellen, ob eine solche Bedingung tatsächlich erfüllt ist. Daher werden alle möglichen Pfade im Workflow geprüft bzw. für jede Bedingung wird während der Analyse einmal angenommen, dass sie erfüllt ist, und einmal, dass sie nicht erfüllt ist. Im Notfallbeispiel würden demnach sowohl die Pfade bei Normalzustand wie auch die Pfade bei Notfallzustand geprüft.

Beispiel. In einem Krankenhaus gilt die Regel, dass Patienten nur dann operiert werden dürfen, wenn sie über die möglichen Konsequenzen der Operation informiert wurden

und der Operation zugestimmt haben. Die Verwendung von Patientendaten zu Operationszwecken darf dementsprechend nur mit vorliegender Einwilligung der Patienten erfolgen. Im Notfall wird diese Regel aber so abgeändert, dass die Operation ohne Einwilligung erfolgen darf, die Patienten dann aber nach der Operation über die Konsequenzen informiert werden müssen. Der Workflow in Abbildung 3.3 veranschaulicht den Ablauf.

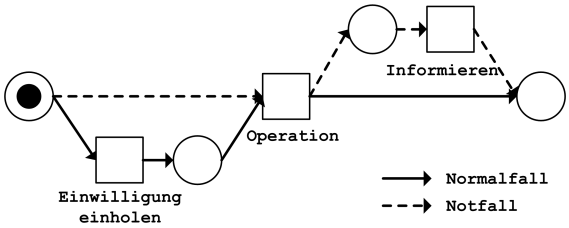


Abbildung 3.3: Beispiel für ein Workflow-Modell mit Ausnahmeregel

Im Normalfall darf die Aktivität *OP* nur ausgeführt werden, wenn vorher die Aktivität *Einwilligung einholen* ausgeführt wurde. Im Notfall wird aber auf diese Vorbedingung verzichtet und stattdessen die Nachbedingung eingeführt, dass die Aktivität *Informieren* auf die Aktivität *OP* folgen muss. ■

3.2.2 Workflow-Transformation

Unter der Annahme, dass die Workflow-Muster von van der Aalst et al. (2003) vollständig sind, kann jeder beliebige Workflow als Petrinetz dargestellt und mit Reveal analysiert werden, da für jedes Muster eine Petrinetz-Modellierung existiert. Die nachfolgende Beschreibung der Petrinetz-Modellierung von Workflows verzichtet daher auf die Diskussion bestimmter Formate wie BPMN oder BPEL. In Abschnitt 3.5 wird näher auf solche Details eingegangen.

Die Hauptelemente eines Workflows sind seine Aktivitäten (welche Schritte werden ausgeführt?) und Daten (was wird bearbeitet?). Hinzu kommen Ressourcen (wer bearbeitet?) und Orte (wo wird ausgeführt bzw. bearbeitet?). Aktivitäten und Daten finden im Petrinetz ihre direkte Entsprechung als Transitionen und Tokens, siehe Abbildung 3.4.

Ressourcen und Orte werden im Petrinetz nicht als eigenständige Elemente modelliert, sondern als Annotationen bzw. Tags an Tokens und Transitionen abgebildet. Der Kontrollfluss eines Workflows wird durch spezielle Tokens repräsentiert. Beim Durchlaufen des Workflows von den Start- zu den Endaktivitäten werden neben den **Daten-Tokens** auch **Kontroll-Tokens** verwendet.

Beispiel. Eine elektronische Patientenakte (EPA) soll von einem Pfleger angelegt, von einem Arzt mit einer Diagnose gefüllt und dann wiederum von einem Pfleger zur Rechnungsstellung verwendet werden. Die Abbildung 3.5a zeigt den entsprechenden

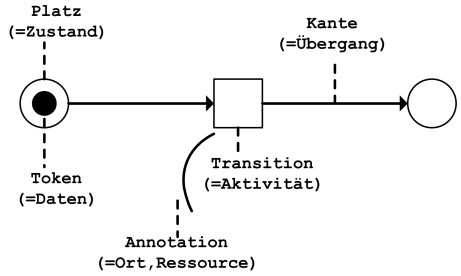


Abbildung 3.4: Petrinetz-Modellierung von Workflow-Elementen

Workflow als Petrinetz. Die drei Aktivitäten *EPA anlegen*, *EPA füllen* und *EPA verwenden* sind jeweils als Transition modelliert. Das im linken Platz befindliche, schwarze Kontroll-Token zeigt an, dass die Workflow-Ausführung noch nicht begonnen hat, der Zustand des Kontrollflusses sich also noch am Anfang des Workflows befindet.

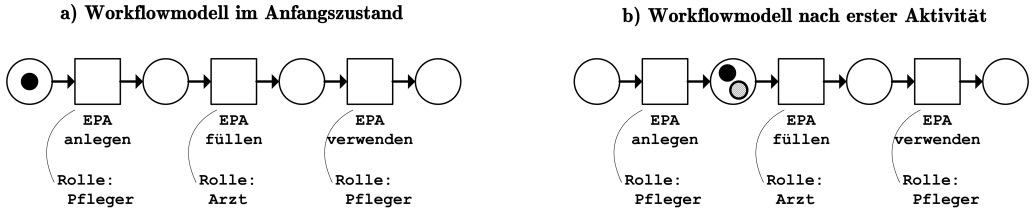


Abbildung 3.5: Beispiel für ein Workflow-Petrinetz

Nach der Ausführung der ersten Aktivität, was dem Feuern der ersten Transition entspricht, befindet sich das Modell im Zustand der Abbildung 3.5b. Das schwarze Kontroll-Token zeigt an, dass die nächste anstehende Aktivität *EPA füllen* ist. Das schraffierte Daten-Token zeigt die von der Aktivität *EPA anlegen* angelegte Patientenakte an. ■

Um das Feuerverhalten eines Petrinetzes deterministisch werden zu lassen, können Regeln für das Feuerverhalten verwendet werden. Diese nachfolgend als **Feuerregeln** (*firing rules*) bezeichneten Regeln legen fest, welche Tokens aus welchem Eingangsplatz eine Transition beim Feuern verbraucht und welche Tokens dabei in welchem Ausgangsplatz erzeugt werden. Das Standard-Feuerverhalten, wie es in Abbildung 3.6 veranschaulicht wird, ist wie folgt definiert: Eine Transition ist aktiv, wenn in jedem ihrer Eingangsplätze mindestens ein Token liegt. Wenn die Transition feuert, verbraucht sie ein Token aus jedem ihrer Eingangsplätze und erzeugt ein Token in jedem ihrer Ausgangsplätze.

Dieses Standardverhalten kann nun in mehrfacher Hinsicht erweitert werden. Für die Modellierung sowohl des Daten- wie auch des Kontrollflusses innerhalb eines Petrinetzes ist die Unterscheidung verschiedener Typen von Tokens sinnvoll. Dementsprechend

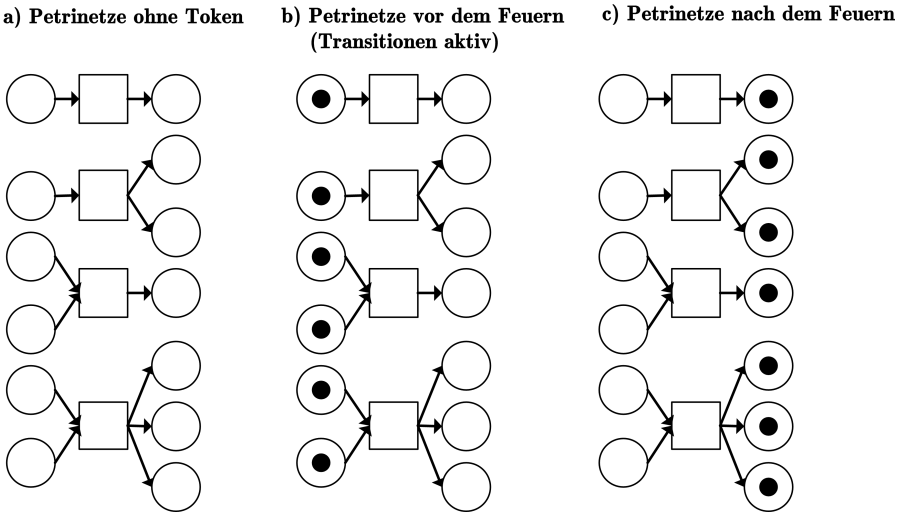


Abbildung 3.6: Standard-Feuerverhalten von Petrinetz-Transitionen

müssen beim Feuern einer Transition ebenfalls verschiedene Token-Typen berücksichtigt werden, so dass nicht zwangsläufig jedes Token zur Aktivierung einer Transition führt und nicht immer nur ein Typ von Tokens erzeugt wird.

Einige Workflow-Konstrukte wie das Muster Multi-Choice (Nr. 6 in Abbildung 2.8) setzen die Ausführung eines Pfads in einem oder mehreren Folgepfaden fort bzw. führen mehrere Pfade in einem zusammen. Wenn nur das Standard-Feuerverhalten zur Verfügung steht, muss jede der möglichen Kombinationen getrennt modelliert werden. Eine Verzweigung, bei der von drei möglichen Pfaden entweder alle oder nur zwei Pfade aktiviert werden, oder sogar nur ein Pfad aktiviert wird, muss dann als acht Einzelfälle modelliert werden, siehe Abbildung 3.7a.

Um nicht jede einzelne der möglichen Kombinationen getrennt modellieren zu müssen, sollte das erweiterte Feuerverhalten die Auswahl bestimmter Folgepfade bzw. Ausgangsplätze erlauben. Komplexe Workflow-Konstrukte lassen sich dann deutlich kompakter darstellen, wie Abbildung 3.7b illustriert.

Reveal erweitert das Feuerverhalten von Transitionen wie folgt.

- Token und Plätze können nach Typen unterschieden werden.
- Transitionen können auch dann aktiv sein, wenn nur n>=1 von m Eingangsplätzen Tokens enthalten.
- Transitionen verbrauchen bei m Eingangsplätzen insgesamt mindestens ein Token, nicht aber mindestens m.
- Transitionen erzeugen bei m Ausgangsplätzen insgesamt mindestens ein Token, nicht aber mindestens m.

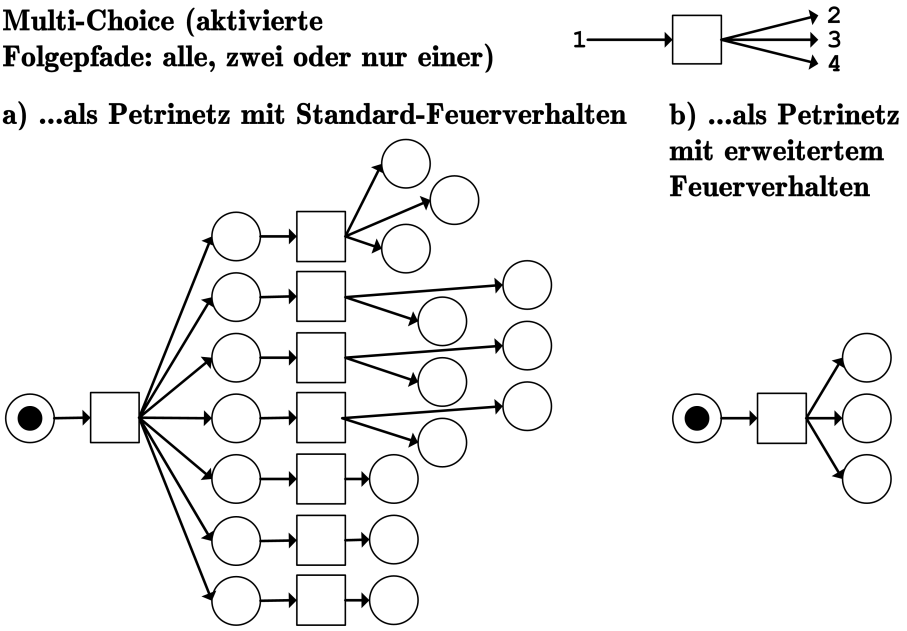


Abbildung 3.7: Petrinetze für das Workflow-Muster *Multi-Choice*

Die konkrete Umsetzung der erweiterten Feuerregeln wird im Rahmen der Implementierung in Abschnitt 3.5 beschrieben.

3.3 Vorlagen zur Formalisierung von Regeln

Die in Tabelle 3.1 zusammengefassten Ausprägungen der Regelangaben werden anhand von einzelnen Petrinetz-Vorlagen für die REVEAL-Prüfung formalisiert. Im Anschluss an die Korrektheitsvorlagen werden die weiteren Vorlagen vorgestellt, und zwar zunächst die Modalität, dann Orte, Reihenfolge, Ressourcen, Verwendung und schließlich Zeitlimits.

3.3.1 Korrektheit

Zusätzlich zu den Regeln bezüglich einzelner Aktivitäten und Daten wird oft gefordert, dass ein Workflow „korrekt“ sein soll. Diese Korrektheit bzw. „soundness“ wird so definiert, dass alle Aktivitäten im Workflow erreichbar sein müssen, das heißt, dass es einen Pfad vom Start des Workflows zu jeder Transition gibt und dass auf keinem dieser Pfade ein Deadlock auftritt (vgl. Reachability- und Liveness-Beschreibung in Murata (1989)). Verstöße gegen die Korrektheitsforderungen werden von Liu und Kumar (2005) als *structural flaws* bezeichnet. In der vorliegenden Arbeit werden diese und weitere Informationsfluss-Schwachstellen zusammengefasst als strukturelle Schwachstellen bezeichnet.

Alle - oder, je nach gewünschter Schärfe der Bedingung, nur einige - Pfade des Workflows müssen vom Start bis zum Ende führen (entsprechende Varianten werden z.B. in Trcka et al. (2008) definiert). Optional muss der Workflow „sauber“ beendet werden, das heißt, beim Erreichen des Workflow-Endes darf kein Token mehr in einem nicht als Ende-Platz gekennzeichneten Platz liegen (z.B. definiert in Fan et al. (2007)). Darüberhinaus ist es für die verlässliche Analyse sinnvoll, noch eine weitere Prüfung durchzuführen, und zwar auf parallele Ausführungspfade, die durch Regeln für Aktivitäten und Daten verknüpft sind. In diesen Fällen kann nämlich keine verlässliche Reihenfolgeprüfung erfolgen, weil die Ausführung von Aktivitäten in parallelen Pfaden zur Entwurfszeit nicht eindeutig zu ordnen ist. In einer Ausführungsumgebung, die echt parallele Ausführung erlaubt, können zwei Aktivitäten gleichzeitig stattfinden. Quasiparallele Ausführungsumgebungen führen zunächst die eine, dann die andere Aktivität aus, wobei die Reihenfolge von nicht im Workflow modellierten Bedingungen (Stichwort *scheduling*) abhängt.

Die zur Prüfung auf diese Korrektheitsanforderungen nötigen Vorlagen werden nachfolgend beschrieben.

Vorlage für Erreichung von Aktivitäten

Eine Aktivität ist dann nicht erreichbar, wenn es auf dem Pfad vom Start zur Aktivität einen Deadlock gibt. Deadlocks entstehen, wenn die durch eine ODER-Verzweigung erzeugten Pfade durch eine UND-Vereinigung zusammengeführt werden. Wird nämlich bei der ODER-Verzweigung einer der Pfade nicht mit einem Token beschickt, wartet die UND-Vereinigung endlos auf das Eintreffen dieses fehlenden Tokens. Die Vorlage in Abbildung 3.8 zeigt, wie sich diese Kombination in einem Petrinetz erkennen lässt.

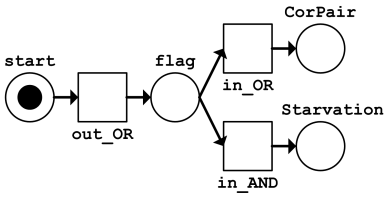


Abbildung 3.8: Verzweigungsvorlage für ODER-Transitionen

Beispiel. Im Beispiel der Abbildung 3.9 wird die Transition *t2* als UND-Vereinigung nicht feuern können, da die ODER-Verzweigung nur genau ein Token entweder in *p2*, *p3* oder *p4* erzeugt. Die Transition *t3* hingegen ist erreichbar, wenn *p4* belegt wird. Die obige Vorlage erlaubt, die kritische Kombination der ODER-Verzweigung in *t1* mit der UND-Verzweigung in *t2* zu entdecken. ■

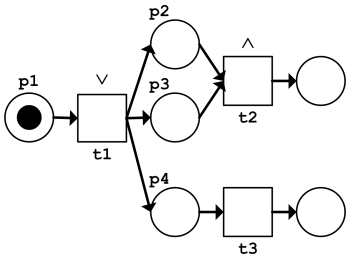


Abbildung 3.9: Beispiel für ODER-Verzweigung mit UND-Vereinigung

Vorlage für Pfade zum Ende des Workflows

Ob es Pfade zum Ende des Workflows gibt und ob alle Pfade zum Ende des Workflows führen, kann dank der Erweiterung um unterscheidbare Tokens mit der Vorlage in Abbildung 3.10 geprüft werden.

Die Transition *split* erzeugt bei jeder verzweigenden Transition im Workflow-Modell so viele Tokens in *ERR* wie Zweige abgehen. Die Transition *join* hingegen unterscheidet zwischen UND- und ODER-Vereinigungen und entfernt bei UND-Vereinigungen nur ein Token aus *ERR*, bevor sie ein neues in *OK* erzeugt. Bei ODER-Vereinigungen werden so viele Tokens aus *ERR* entnommen und in *OK* angelegt, wie Zweige zusammenlaufen.

Verbleiben bei Erreichen des Workflow-Endes noch Tokens in *ERR*, steht jedes Token für einen nicht zum Ende des Workflows führenden Pfad. Ist *ERR* leer, führen alle Pfade des Workflows zum Ende.

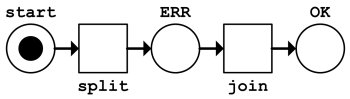


Abbildung 3.10: Muster für Prüfung auf Pfade zum Ende des Workflows

Beispiel. Nicht alle Pfade des Workflow-Modells in Abbildung 3.11 führen zum Ende. Die Transition *t2* kann nie feuern, da entweder nur *p2* oder nur *p3* ein Token enthalten wird (erkennbar an der negativen Nummerierung). Die ODER-Verzweigung bei *t4* kann aber über *t3* bedient werden, und beide Pfade der UND-Verzweigung in *t5* führen über die ODER-Vereinigung in *t8* zum Ende. Die Vorlage enthält also beim Erreichen des Workflow-Endes die Token *-1* und *-2* in *ERR* und zeigt damit an, dass nicht alle Pfade des Workflows abgeschlossen werden können. Die Token *-3, 4, 5, 6, 7* in *OK* hingegen zeigen, dass es Workflow-Pfade gibt, die zum Ende führen. Sowohl die starke wie die schwächeren Varianten der Anforderung nach Erreichbarkeit des Workflow-Endes sind also mit dieser Vorlage prüfbar. ■

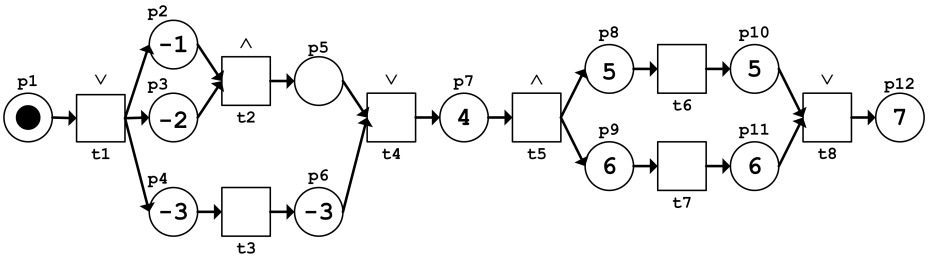


Abbildung 3.11: Beispiel für Pfade zum Ende des Workflows

Vorlage für Ende aller Aktivitäten bei Ende des Workflows

Wird das Workflow-Ende erreicht, sind nicht zwingend alle Aktivitäten beendet. Es können vor allem dann noch Tokens im Workflow verbleiben, wenn auf eine UND-Verzweigung eine ODER-Vereinigung folgt. Da die ODER-Vereinigung die eingehenden Pfade nicht synchronisiert, sondern bereits beim ersten eintreffenden Token feuern kann, ist nicht gewährleistet, dass diese Kombination beim Erreichen des Workflow-Endes vollständig abgearbeitet wurde. Die Abbildung 3.12 enthält eine Vorlage, mit dem das Vorhandensein solcher kritischen Kombinationen in einem Petrinetz erkannt werden kann.

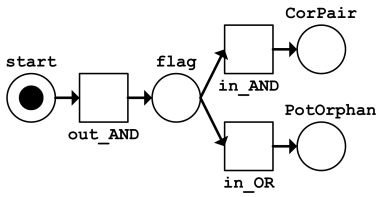


Abbildung 3.12: Verzweigungsvorlage für UND-Transitionen

Beispiel. Im Beispiel der Abbildung 3.13 erzeugt die UND-Verzweigung der Transition *t1* ein Token in *p2* und ein Token in *p3*. Sobald eines der Token für die ODER-Vereinigung der Transition *t4* zur Verfügung steht, kann *t4* feuern und das Workflow-Ende signalisieren. Das jeweils andere Token wird sich aber noch im Workflow befinden, und zwar entweder im oberen Pfad in *p2* oder *p4* oder aber im unteren Pfad in *p3* oder *p5*. Deswegen ist nicht gewährleistet, dass bei Erreichen des Workflow-Endes beide Pfade abgearbeitet wurden. Mit Hilfe der obigen Vorlage kann die Kombination *t1*-*t4* als kritische UND-Verzweigung mit ODER-Vereinigung erkannt werden. ■

Vorlage für parallele Ausführung von Aktivitäten

Mit den bereits vorgestellten Reihenfolge-Vorlagen lässt sich erkennen, ob eine Aktivität A auf einem Ausführungspfad vor einer Aktivität B modelliert wurde. Ein Fall

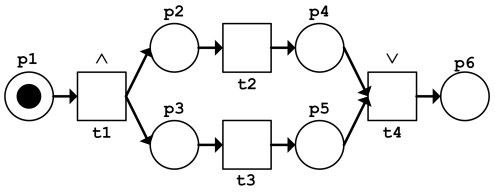


Abbildung 3.13: Beispiel für ODER-Verzweigung mit UND-Vereinigung

ist damit allerdings noch nicht abgedeckt, nämlich der der parallelen Ausführungspfade. Eine UND-Verzweigung erzeugt zwei oder mehr solcher parallelen Pfade. Da zur Entwurfszeit nur das Workflow-Modell vorliegt, die tatsächlichen Ausführungszeitpunkte also unklar sind, kann im allgemeinen Fall bei parallelen Pfaden nicht vorab entschieden werden, in welcher Reihenfolge die Aktivitäten ausgeführt werden. Eine vollständige Analyse muss diesen Umstand erkennen und eine entsprechende Warnung ausgeben, wann immer ein potentieller Konflikt vorliegt. Gibt es eine Regel bezüglich der Ausführungsreihenfolge von zwei Aktivitäten, und liegen diese auf unterschiedlichen, aber parallelen Pfaden, handelt es sich um einen solchen Konflikt. Die Vorlage in Abbildung 3.14 erfasst diesen.

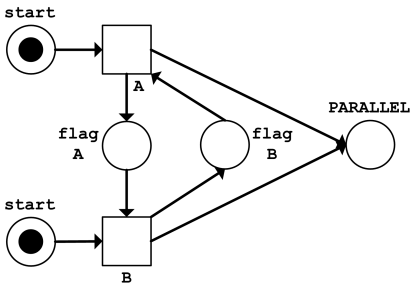


Abbildung 3.14: Vorlage zur Erkennung paralleler, reihenfolgeabhängiger Aktivitäten

Beispiel. Aktivität A soll vor Aktivität B ausgeführt werden. Das Workflow-Modell in Abbildung 3.15 erfüllt diese Regel - manchmal. Abhängig von der Ausführungsumgebung (Stichwort *scheduling*) wird in einigen Fällen aber B vor A ausgeführt, oder, wenn die Ausführungsumgebung eine echt parallele Ausführung ermöglicht, gleichzeitig mit A. Diese Unklarheit ist bei der Analyse zur Entwurfszeit nicht auflösbar. Anhand der Vorlage in Abbildung 3.14 kann aber eine entsprechende Warnung ausgegeben werden. ■

3.3.2 Modalität

Die Modalität beschreibt für ein Workflow-Element (Aktivitäten oder Daten), ob es im Workflow-Modell enthalten sein muss oder nicht enthalten sein darf. Ein dritter

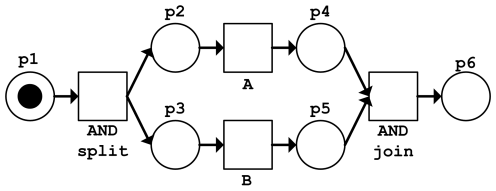


Abbildung 3.15: Beispiel für parallele, reihenfolgeabhängige Aktivitäten

Wert könnte ausdrücken, dass ein Element enthalten sein darf. Da in diesem Fall aber sowohl die Anwesenheit wie auch die Abwesenheit des Elements die entsprechende Regel einhalten würde, wird darauf nicht näher eingegangen.

Modalität „darf nicht“

Für die Prüfung auf das Nicht-Vorhandensein von Elementen ist eine einfache Vorlage wie in Abbildung 3.16 ausreichend. Taucht das Element X im Workflow-Modell auf, zeigt ein Token in *ERR* dies an. Ansonsten zeigt ein Token in *OK*, dass das verbotene Element X im Modell nicht gefunden wurde.

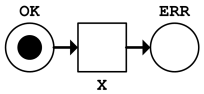


Abbildung 3.16: Vorlage für verbotene Elemente

Modalität „muss“

Elemente, die im Modell enthalten sein **müssen**, können mit der Vorlage in Abbildung 3.17 entdeckt werden. Es funktioniert analog zur obigen Vorlage „darf nicht“, einzig die Benennung der Plätze ist vertauscht. Ein Token in *ERR* zeigt an, dass das geforderte Element nicht im Modell gefunden wurde.

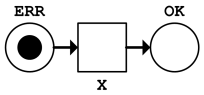


Abbildung 3.17: Vorlage für geforderte Elemente

3.3.3 Orte

Auf die Einhaltung von Ortsvorgaben wird mit der Vorlage in Abbildung 3.18 geprüft. Stimmt die Ortsangabe einer Aktivität oder eines Datenelements mit der in der jeweiligen Regel enthaltenen Ortsvorgabe überein, wird ein Token in *OK* erzeugt. Stimmen

die Angaben nicht überein, erzeugt die Transition ein Token in *ERR* und zeigt damit den Verstoß an.

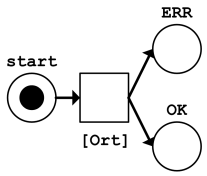


Abbildung 3.18: Vorlage für Ortsvorgaben

Beispiel. Im Beispiel der Abbildung 3.19 darf die Aktivität *A* nur in Freiburg ausgeführt werden. Wurde im Workflow-Modell diese Aktivität an einem anderen Ort modelliert, wird ein Token in *ERR* erzeugt, beispielsweise für Berlin, Hamburg und München. Wenn die Ausführung an allen Orten **außer** Freiburg erlaubt sein soll, braucht nur das Feuerverhalten der Vorlage invertiert zu werden, so dass die Tokens bei Berlin, Hamburg und München in *OK* erzeugt werden. ■

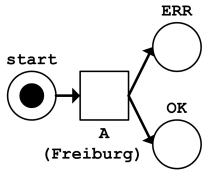


Abbildung 3.19: Beispiel für geforderten Ausführungsort

3.3.4 Reihenfolge

Grundsätzlich gibt es bezüglich der Reihenfolge zweier Aktivitäten X und Y nur zwei Unterscheidungen, nämlich X **vor** Y und X **nach** Y (der Fall der parallelen Ausführung wird weiter unten diskutiert, ist aber auf Workflow-Modellebene nur insofern von Interesse, als dass er die Bestimmung der Ausführungsreihenfolge erschwert). Bezüglich der Reihenfolge von zwei Elementen X und Y sind dennoch drei Regeln zu berücksichtigen. Es zeigt sich, dass die umgangssprachliche Formulierung „X vor Y“ präziser als „Y erfordert X“ angegeben werden kann. Ebenso ist „Y nach X“ präziser als „X löst Y aus“ zu sehen. Schließlich gibt es noch eine dritte Regel: Wenn X und Y im Modell auftreten, muss X vor Y stehen.

Umgangssprachlich haben die Regeln keine hohe Trennschärfe. Streng logisch gesehen beschreiben die drei Regeln aber unterschiedliche Forderungen. Tabelle 3.2 enthält die logischen Wahrheitswerte für alle vier möglicherweise in einem Workflow auftretenden Fälle: nur X tritt auf, nur Y tritt auf, X tritt vor Y auf, X tritt nach Y auf.

In Awad et al. (2009) werden Reihenfolge-Verletzungen in vier Typen eingeteilt.

Tabelle 3.2: Wahrheitstabelle für Reihenfolgeregeln

	X vor Y (Y erfordert X)	Y nach X (X löst Y aus)	Wenn X und Y, dann zuerst X
- X -	✓	¬	✓
- Y -	¬	✓	✓
- X - Y -	✓	✓	✓
- Y - X -	¬	¬	¬

1. Splitting Choice: X wird ausgeführt, aber je nach gewähltem Folgepfad wird Y nicht ausgeführt.
2. Different Branches: X und Y treten nur einzeln in verschiedenen Pfaden auf.
3. Inverse Order: X und Y treten auf dem selben Pfad, aber in der falschen Reihenfolge auf.
4. Lack of Activity: Die zwingend auf X folgende Aktivität Y ist in keinem Pfad enthalten.

Alle vier Typen von Reihenfolge-Verletzungen lassen sich mit den nachfolgend präsentierten Vorlagen entdecken. Regeln wie „darf nicht vorher“ und „muss vorher“ können durch Kombination mit einer Modalitätsangabe ausgedrückt werden.

Reihenfolge: Y erfordert X

Die Regel **Y erfordert X** bzw. umgangssprachlich „X vor Y“ wird durch die Vorlage in Abbildung 3.20 formalisiert. Die Transitionen *X* und *Y* repräsentieren das jeweilige Element X und Y. Tritt Element X im Workflow-Modell auf, feuert Transition *X* und produziert ein Token im Platz *OK/flag X*. Folgt bis zum Workflow-Ende nicht das Element Y, bleibt dieses Token dort liegen und zeigt an, dass die Regel eingehalten wurde (vgl. erste Spalte, erste Zeile der Tabelle 3.2). Tritt das Element Y auf, feuert die Transition *Y*, und zwar in Abhängigkeit des Platzes *OK/flag X*. Liegt dort kein Token, wurde also Element X nicht vorher gefunden, legt Y ein Token im Platz *ERR* ab, wo es einen Regelverstoß anzeigt (vgl. erste Spalte, zweite Zeile der Tabelle 3.2). Liegt ein Token in *OK/flag X*, verbraucht *Y* das Token und produziert ein neues im Platz *OK*, um die Regeleinhaltung anzuzeigen (vgl. erste Spalte, dritte Zeile in Tabelle 3.2).

Die Vorlage in Abbildung 3.20a bildet den Fall der falschen Reihenfolge (vgl. erste Spalte, vierte Zeile in Tabelle 3.2) nicht explizit ab, liefert aber das richtige Prüfungsergebnis. Zur expliziten Abbildung, das heißt zur eindeutigen Anzeige der falschen Reihenfolge, dient die Vorlage in Abbildung 3.20b. Stößt die Analyse auf das Element Y, ohne dass das Element X vorher gefunden wurde, feuert die Transition *Y* ohne ein Token in *OK/flag X* vorzufinden, woraufhin ein Token in *ERR* abgelegt wird. Ein anschließend im Workflow-Modell gefundenes Element X schaltet dann die Transition *X*,

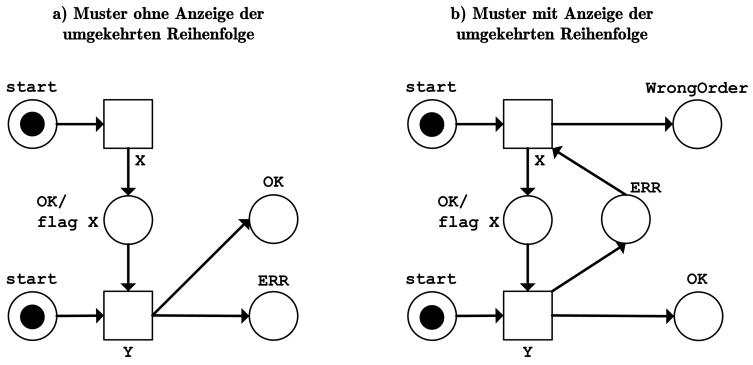


Abbildung 3.20: Vorlage für Reihenfolge „Y erfordert X“

welche wegen des in *ERR* liegenden Tokens ein neues Token in *WrongOrder* erzeugt. Daraus kann dann die falsche Reihenfolge direkt abgelesen werden.

Reihenfolge: X löst Y aus

Die Prüfung auf Einhaltung der Regel **X löst Y aus** bzw. umgangssprachlich „Y nach X“ erfolgt mit der Vorlage in Abbildung 3.21. Die Unterschiede zur Regel „Y erfordert X“ bestehen in der umgekehrten Wertung der beiden Fälle, dass entweder nur das Element X oder nur das Element Y gefunden werden (vgl. Spalte 1 und 2 in den Zeilen 1 und 2 der Tabelle 3.2).

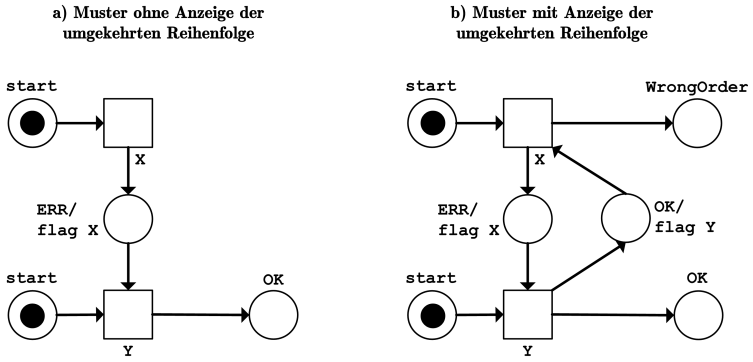


Abbildung 3.21: Vorlage für Reihenfolge „X löst Y aus“

Beim Vorfinden des Elements X produziert die Transition *X* ein Token in *ERR/flag X*. Dort bleibt das Token liegen und signalisiert einen Regelverstoß (vgl. zweite Spalte, erste Zeile in Tabelle 3.2), es sei denn, das Element Y wird anschließend gefunden. *Y* verbraucht das ggf. in *ERR/flag X* liegende Token und produziert ein neues Token in *OK*. Damit zeigt die Vorlage sowohl für den Fall, dass nur Element Y gefunden wird, wie auch für den Fall dass Y nach X gefunden wird, die Regeleinhaltung an (vgl. zweite Spalte, zweite und dritte Zeile in Tabelle 3.2).

Zur Anzeige der falschen Reihenfolge, also des Auftretens von Element Y vor anstatt nach Element X, kommt die Vorlage in Abbildung 3.21b zum Einsatz. Die Funktionsweise ist analog zur bereits beschriebenen Vorlage in Abbildung 3.20. Bezüglich der Vorlage in Abbildung 3.21a sei nur angemerkt, dass auch hier die falsche Reihenfolge bemerkt und durch ein Token in *ERR/flag X* angezeigt wird, jedoch erst nachdem *Y* ein Token in *OK* abgelegt hat. Die Vorlage dient damit eher nur der Veranschaulichung des Analyseansatzes. Für die tatsächliche Analyse empfiehlt sich die Vorlage mit Anzeige der umgekehrten Reihenfolge.

Reihenfolge: Wenn X und Y, dann zuerst X

Zur Formalisierung der dritten Reihenfolgeregel, **Wenn X und Y, dann zuerst X**, dient die Vorlage in Abbildung 3.22. Sofern entweder nur das Element X oder nur das Element Y im Workflow-Modell gefunden werden, feuert nur die entsprechende Transition *X* oder *Y* und produziert ein Token in *OK/flag X* oder *OK/flag Y*. Die Regel wird in beiden Fällen eingehalten (vgl. dritte Spalte, erste und zweite Zeile in Tabelle 3.2). Auch wenn beim Auslösen der Transition *Y* bereits ein Token in *OK/flag X* liegt, wird ein Token in *OK/flag Y* produziert. Zusätzlich wird dann allerdings das Token in *OK/flag X* verbraucht. Der Fall Y nach X wird damit als regelkonform angezeigt (vgl. dritte Spalte, dritte Zeile in Tabelle 3.2).

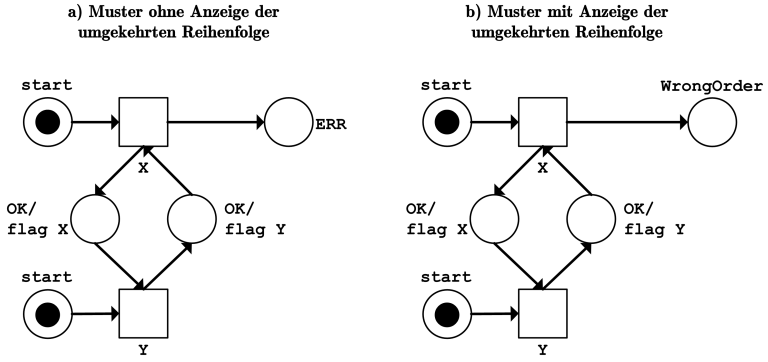


Abbildung 3.22: Vorlage für Reihenfolge „Wenn X und Y, dann zuerst X“

Für den Fall, dass X vor Y auftritt, liegt nach der Analyse ein Token im Platz *ERR* in der Vorlage in Abbildung 3.22a. Tatsächlich ist dieser Fall der falschen Reihenfolge der einzige Fall, in dem *ERR* ein Token erhält (vgl. dritte Spalte, vierte Zeile in Tabelle 3.2). Deswegen muss zur Anzeige der falschen Reihenfolge wie in Abbildung 3.22b nur *ERR* in *WrongOrder* umbenannt werden.

Beispiel. Im Beispiel der Abbildung 3.23 erfüllt nur der obere Pfad die Reihenfolgebedingung „B nach A“. Im unteren Pfad wird *B* ausgeführt, obwohl *A* noch nicht ausgeführt wurde. Die Verletzung tritt sogar zweimal hintereinander auf, nach *p3* und nach *p5*. ■

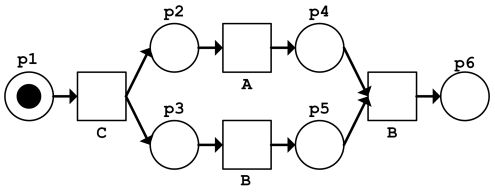


Abbildung 3.23: Beispiel für Reihenfolge „B nach A“

3.3.5 Ressourcen

Die Überprüfung auf Einhaltung von Ressourcenvorgaben erfolgt mit der gleichen Vorlage, wie es auch für Ortsvorgaben verwendet wird. Der Vollständigkeit halber zeigt Abbildung 3.24 die Vorlage in der Ausprägung für Ressourcen.

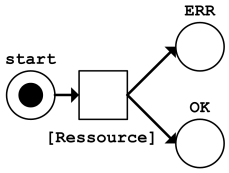


Abbildung 3.24: Vorlage für Ressourcenvorgaben

3.3.6 Verwendung

Generell lassen sich Regeln zur Verwendung von Daten durch bestimmte Aktivitäten anhand der obigen Reihenfolgeregeln abdecken, wobei dann eben nicht beide zu prüfende Elemente vom selben Typ sind, sondern eines ein Datenelement und eines ein Aktivitätenelement ist. Wie nachfolgend gezeigt, genügen jedoch in vielen Fällen einfachere Vorlagen zur Prüfung, wenn es um die direkt anschließende Verwendung/-Löschung oder die Verwendung/Löschung vor dem Workflow-Ende geht.

In vielen Regularien wird die Löschung oder eine spezielle Verwendung von Daten vorgeschrieben, so etwa in HIPAA die Anonymisierung von Gesundheitsdaten vor der Nutzung zu Forschungszwecken oder allgemein die Löschung von Daten entweder direkt nach einer bestimmten anderen Aktivität oder vor dem Ende des Workflows. Hervorzuheben ist, dass Reveal die Löschung von Daten nicht getrennt analysiert, sondern als eine von vielen möglichen Verwendungen behandelt. Wenn im Folgenden von der Verwendung die Rede ist, ist die Datenlöschung (als eine besondere Verwendung) dabei immer inbegriffen.

Datenverwendung durch eine bestimmte Aktivität

Wenn für ein bestimmtes Datenelement *dat* die Verwendung im Workflow vorgeschrieben ist, kann die Vorlage in Abbildung 3.25 zur Prüfung verwendet werden. Nach dem

ersten Auftreten bzw. der Erzeugung des Datenelements legt die erste Transition ein Token in *ERR* ab, wodurch die bisher nicht erfolgte Verwendung signalisiert wird. Beim Auftreten einer Aktivität *A*, die das Datenelement *dat* verwendet, wird das Token aus *ERR* entfernt und ein neues in *OK* erzeugt. Damit bestätigt die Vorlage die Verwendung des Datenelements durch die Aktivität *A*.

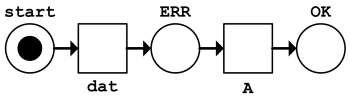


Abbildung 3.25: Vorlage für zwingend zu verarbeitende Datenelemente

Datenverwendung direkt nach einer Aktivität

Falls ein Datenelement sofort nach einer Aktivität verwendet bzw. gelöscht werden muss, kann die Einhaltung im Modell mit der Vorlage in Abbildung 3.26 geprüft werden. Nach der erstmaligen Verwendung des Datenelements *dat* liegt ein Token in *flag*. Die direkt folgende Aktivität im Workflow stößt die Transition *First* an, woraufhin diese das Token aus *flag* verbraucht und ein neues in *OK* oder *ERR* erzeugt: handelt es sich bei der Aktivität um die gewünschte, wie z.B. das Löschen der Daten, in *OK*. Handelt es sich um eine andere, nicht regelkonforme Verwendung, in *ERR*.

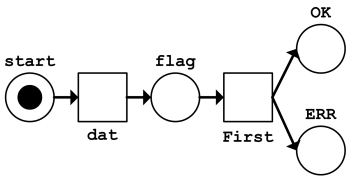


Abbildung 3.26: Vorlage für sofortige Verwendung nach einer Aktivität

Datenverwendung vor Workflowende

Die Vorlage in Abbildung 3.27 kann anzeigen, ob Datenelemente vor dem Workflow-Ende verwendet werden oder nicht. Nach dem ersten Auftreten eines Datenelements im Workflow-Modell liegt ein Token in *flag*. Wird das Datenelement von einer Aktivität verwendet, feuert die Transition *Del*, wodurch das Token aus *flag* entnommen und ein neues in *OK* erzeugt wird. Liegt bei Erreichen des Workflow-Endes noch immer ein Token in *flag*, feuert *End*, so dass *flag* kein Token mehr enthält und stattdessen ein Token in *ERR* abgelegt wird.

Beispiel. Der Workflow in Abbildung 3.28 enthält zwei Pfade. Auf beiden Pfaden wird zunächst eine Patientenakte durch *OPEN* geöffnet, eine Behandlung durch *TREAT*

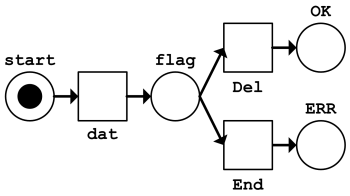


Abbildung 3.27: Vorlage für Verwendung bzw. Löschung vor Workflow-Ende

durchgeführt und die Patientenakte durch *CLOSE* wieder geschlossen. Nur im oberen Pfad werden durch *DEL* angefallene, nur kurzfristig für die Behandlung benötigte Daten gelöscht. Wenn diese Datenlöschung aber durch eine Regel gefordert wird, erfüllt damit nur der obere Pfad die Regel. Nach der Analyse des unteren Pfads würde in der entsprechenden Vorlage ein Token in *ERR* verbleiben und damit die strukturelle Schwachstelle anzeigen.

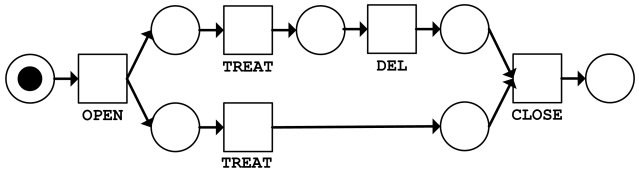


Abbildung 3.28: Beispiel für Datenlöschung

3.3.7 Zeitlimits

Die tatsächlichen Ausführungszeitpunkte von Aktivitäten können zur Entwurfszeit nicht bestimmt werden. Erst zur Laufzeit werden sie bekannt und können dann durch Monitoring überwacht oder im Nachhinein von Log-basierten Ansätzen untersucht werden. Um trotzdem bereits bei der Modellprüfung zur Entwurfszeit eine gewisse Kontrolle zeitlicher Fristen zu gewährleisten, sucht REVEAL nach Kontrollaktivitäten.

Eine solche Kontrollaktivität, in den folgenden Vorlagen mit K bezeichnet, muss entweder vor oder hinter einer Aktivität A stehen, die eine Frist bedingt. Wie die Abbildung 3.29 veranschaulicht, gibt es zusätzlich noch die Unterscheidung minimal oder maximal vorher bzw. nachher.

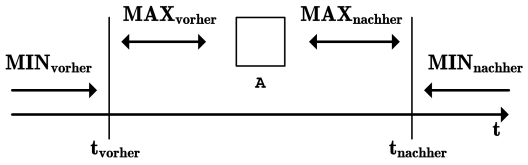


Abbildung 3.29: Minimale/maximale Ausführungszeitpunkte vor/nach einer Aktivität

Wird der Zeitpunkt t_{vorher} als minimaler Vorher-Zeitpunkt angegeben, bedeutet dies, dass eine bestimmte andere Aktivität bis **spätestens** zu diesem Zeitpunkt vor der Aktivität A ausgeführt worden sein muss. Als maximaler Vorher-Zeitpunkt gesehen, dürfte eine bestimmte andere Aktivität **frühestens** zu diesem Zeitpunkt vor X liegen.

Derselben Logik folgend bezeichnet $t_{nachher}$ als maximaler Nachher-Zeitpunkt den Zeitpunkt, bis zu dem eine bestimmte andere Aktivität **spätestens** nach A ausgeführt worden sein muss. Der minimale Nachher-Zeitpunkt legt fest, ab wann eine bestimmte andere Aktivität **frühestens** nach A ausgeführt werden darf.

Beispiel. Die Verabreichung eines Indikationsmittels vor einer Untersuchung muss spätestens bis zu einem gewissen Zeitpunkt vor der Untersuchung erfolgen, damit sich das Mittel ausbreiten kann. Das entspricht einem minimalen Vorher-Zeitpunkt. Gleichzeitig bedingt die Verabreichung aber auch einen maximalen Vorher-Zeitpunkt, ab dem frühestens das Mittel eingesetzt werden darf, denn wenn die Verabreichung zu lange vor der Untersuchung erfolgt, ist die Indikationswirkung unzureichend.

Wenn eine Folgeuntersuchung frühestens zwei Stunden im Anschluß stattfinden darf, ist damit ein minimaler Nachher-Zeitpunkt gegeben. Muss eine Folgeuntersuchung innerhalb von 24 Stunden im Anschluss stattfinden, wird dadurch ein maximaler Nachher-Zeitpunkt definiert. ■

Da die tatsächlichen Ausführungszeitpunkte bei der Modellanalyse zur Entwurfszeit nicht bekannt sind, kann REVEAL nicht zwischen minimalen und maximalen Zeiträumen unterscheiden. Für die Behebung von strukturellen Schwachstellen, die durch fehlende Kontrollaktivitäten deutlich werden, ist die Unterscheidung aber relevant, um eventuell fehlende Aktivitäten an einer zeitlich sinnvollen Stelle einzufügen. Die von der einzufügenden Aktivität benötigte Ausführungsdauer muss dann berücksichtigt werden, damit sie rechtzeitig vor oder nach der Bezugsaktivität A abgeschlossen werden kann. Die Unterscheidung zwischen minimalen und maximalen Zeiträumen ist daher tatsächlich nur für die Behebung von Schwachstellen relevant, so dass REVEAL mit der reinen Unterscheidung von Vorher-/Nachher-Zeitpunkten eine genügende Trennschärfe für die Prüfung bietet.

Zeitpunkte bzw. Kontrollaktivitäten vorher/nachher

Minimale und maximale **Vorher**-Zeitpunkte erfordern eine Kontrollaktivität K, die **vor** der auslösenden Aktivität A im Workflow-Modell steht. Die Vorlage in Abbildung 3.30a erfasst dazu, ob K vor A modelliert wurde (es entspricht der Vorlage für „Y erfordert X“). Wenn ja, bleibt ein Token in *OK* liegen. Wenn nur K gefunden wird, liegt das Token in *OK/flag K*. Tritt A alleine auf, liegt das Token in *ERR*. Werden A und K zwar entdeckt, jedoch in der falschen Reihenfolge, zeigt ein Token in *WrongOrder* dies an.

Minimale und maximale **Nachher**-Zeitpunkte werden durch eine Kontrollaktivität **nach** der auslösenden Aktivität A abgebildet. Entsprechend wird durch die Vorlage in

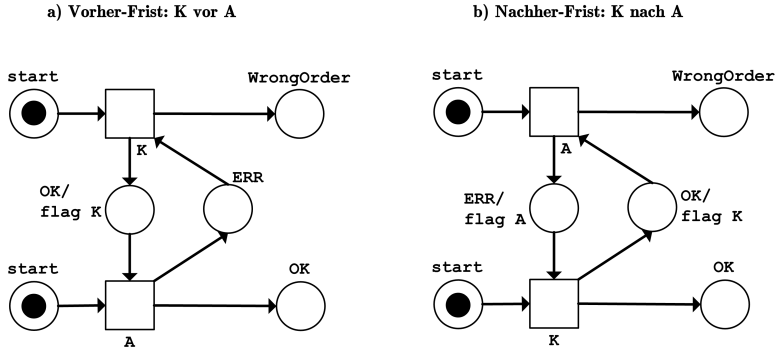


Abbildung 3.30: Vorlage für Zeitpunkte bzw. Kontrollaktivitäten vor/nach einer Aktivität

Abbildung 3.30b geprüft, ob die Kontrollaktivität K nach A im Modell auftritt (die Reihenfolge-Vorlage dazu lautet „X löst Y aus“). Wird die Kontrollaktivität alleine oder nach A gefunden, verbleibt ein Token in *OK/flag K* bzw. in *OK*. Wird A ohne Kontrollaktivität gefunden, zeigt ein Token in *ERR/flag A* dies an. Die falsche Reihenfolge „*A* folgt auf *K*“ wird auch hier durch ein Token in *WrongOrder* signalisiert.

Es sei noch einmal hervorgehoben, dass die letztendlichen Ausführungszeitpunkte bei einer Analyse zur Entwurfszeit unbekannt sind. Anhand des Umwegs über die Prüfung auf das Vorhandensein von Kontrollaktivitäten vor oder nach einer Aktivität, die eine Frist bedingt, kann aber zumindest sichergestellt werden, dass eine Kontrolle der Fristen im Modell vorgesehen ist. Die modellierten Kontrollaktivitäten müssen dann den rechtzeitigen Beginn oder die rechtzeitige Beendigung der fristauslösenden Aktivitäten kontrollieren und bei einem - eben erst zur Laufzeit feststellbaren - Verstoß eine Warnung erzeugen.

Beispiel. Der in Abbildung 3.31a dargestellte Workflow nimmt eine Patientenanfrage zur Datenaktualisierung in *Request* an, führt die Aktualisierung in *Update* durch und informiert den Patienten in *Inform* über die erfolgreiche Aktualisierung. Es gelte nun die Regel, dass der Patient innerhalb von 30 Tagen nach seiner Anfrage informiert werden muss. Eine Prüfung des Workflows zeigt, dass keine Kontrollaktivität die Einhaltung dieser Frist überwacht. Der Workflow in Abbildung 3.31b hingegen enthält die geforderte Kontrollaktivität K und wird deswegen als regelkonform betrachtet. ■

3.4 Analysealgorithmus

Der Prüfalgorithmus führt eine komplette Untersuchung des Workflows durch: Von jedem Startpunkt des Workflow-Modells wird auf jedem Pfad zum Ende des Workflows jede Aktivität betrachtet. Während des Durchlaufs werden die zu der jeweiligen Aktivität gehörenden Regel-Petrinetze geschaltet. Zu jedem Zeitpunkt während

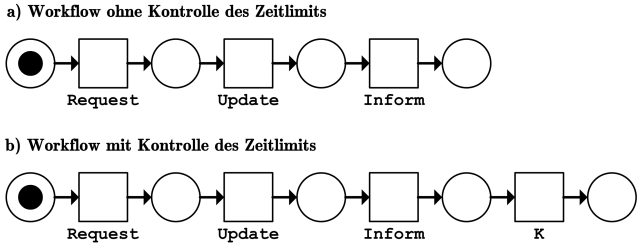


Abbildung 3.31: Beispiel für Kontrollaktivitäten zur Zeitpunktkontrolle

des Durchlaufs lassen sich an den Zuständen der Regel-Petrinetze die bisher gefundenen Schwachstellen ablesen. Die Plätze sind dazu in drei Kategorien eingeteilt: OK signalisiert Regeleinhaltung, ERR signalisiert Regelverstöße, und UNRES signalisiert nicht auflösbare Einhaltungsfragen. Nach dem Durchlauf können an diesen Plätzen die Ergebnisse der Untersuchung abgelesen werden. Zusätzliche Annotationen an den Fehlerzuständen enthalten dabei Details der Schwachstellen, z.B. welche Rolle welche Aktivität auszuführen versucht hat, aber nicht ausführen durfte.

Listing 1 zeigt den Analysealgorithmus als Pseudocode.

```
foreach workflow path in WF from Pstart to Pend do
    // visit all WF paths
    Identify current WF:transition;
    foreach matching Rule:transition do
        // trigger applicable PPNs
        Fire PPN:transition depending on WF:annotations;
        if branching then
            create unique token for each branch
        Move to next WF:transition(s);
    // loop will end when all paths have been visited
foreach error place in PPN do
    // list policy violations
    foreach token in error place do
        Identify WF:path through token ID;
        Use token tags to identify WF:transition responsible for violation;
        Evaluate Rule:annotations and provide details to analyst;
        if WrongOrder errors then
            identify and show both WF:transitions
    // loop will end when all violations have been analyzed
```

Algorithmus 1: REVEAL-Algorithmus

3.4.1 Verknüpfung von Regel- und Workflow-Petrinetzen

Um die Wiederverwendbarkeit der generierten Regelvorlagen zu gewährleisten, werden diese nicht in den Workflow eingefügt, sondern separat gehalten. Dennoch muss natürlich bei der Gegenüberstellung von Workflow und Regeln eine Verknüpfung geschaffen werden. Diese geschieht bei Reveal über die Namen beziehungsweise, sofern Namen mehrfach vorkommen, über die eindeutigen Identifikatoren von Aktivitäten.

Beispiel. Enthält der Workflow eine Aktivität wf:A, werden in den Regel-Petrinetzen alle gleichnamigen Transitionen r:A geschaltet. Sind im Workflow mehrere wf:A-Aktivitäten enthalten, erfolgt die Verknüpfung über die IDs, so dass bei wf:A(3) nur r:A(3) und nicht auch r:A(1) oder r:A(2) geschaltet werden. ■

3.4.2 Suchlauf

In einem Suchlauf verfolgt Reveal jeden Pfad von den Startpunkten des Workflows bis zu den Endpunkten des Workflows. Analog zur Tiefensuche steigt die Suche dabei rekursiv in das Workflow-Petrinetz ab, bevor Verzweigungen in die Breite untersucht werden. Petrinetz-typisch steigt die Anzahl der zu durchsuchenden Pfade exponentiell mit der Anzahl der verzweigenden Plätze und Transitionen. Der Speicherbedarf für den Suchstatus wächst nur auf die Tokens bezogen exponentiell; auf die Regel-Petrinetze bezogen ist der Speicherbedarf konstant, da ein Regel-Petrinetz für beliebig viele Pfade gleichzeitig zur Suche genutzt werden kann. Die Pfadunterscheidung innerhalb der Netze erfolgt anhand von eindeutigen Kennungen. Tokens tragen eine ID, die genau einen Pfad identifiziert (siehe Abschnitt 3.5.3 für Details).

Für jedes Workflow-Element wird geprüft, ob passende Regeln vorliegen und ob die Regeln erfüllt werden. Passend bedeutet dabei, dass im Regel-Petrinetz eine gleichnamige Transition vorhanden ist. Wie die Vorlagen für die einzelnen Regeltypen aussehen, ist aus Abschnitt 3.3 ersichtlich. Bei Aktivitäten können die Modalität, Ortsangaben, Reihenfolgen, Ressourcen und Zeitlimits geprüft werden, bei Daten die Modalität, Ortsangaben, Ressourcen und Verwendungen.

Daraus ergeben sich verschiedene Fälle, die aus der Regelprüfung resultieren können. Grundsätzlich gibt es drei solcher Fälle: die Regel wird eingehalten, die Regel wird nicht eingehalten, oder die Regeleinhaltung ist mit den vorliegenden Informationen nicht zu bestimmen. Darüber hinaus gibt es verschiedene Anhaltspunkte, anhand derer die Einhaltung geprüft wird. Angaben bezüglich der Modalität, Orte, Ressourcen und Verwendung werden als Annotationen (gespeichert als XML-Tags) an den jeweiligen Workflow- bzw. Regelelementen mitgeführt. In der in Abschnitt 3.5 beschriebenen Implementierung des Prototypen wird die Petrinet-Markup-Language PNML zur XML-basierten Speicherung der Netze samt der Zusatzinformationen verwendet. Die geforderte Reihenfolge von Aktivitäten wird im Regel-Petrinetz direkt festgelegt, vgl. Vorlage in Abschnitt 3.3.4. Geforderte Zeitlimits werden direkt als Aktivität im Regel-Petrinetz modelliert und genau wie Reihenfolge-Regeln als Vorlage festgehalten.

Meda et al. (2010) empfehlen, die Ermittlung der Workflow-Pfade und die Analyse in einem Schritt zu vereinen. Im Gegensatz zu dem von ihnen vorgestellten Verfahren „GT4DF“, welches im ersten Schritt eine Instanz jedes Pfades erzeugt und im zweiten Schritt diese Instanzen analysiert, arbeitet REVEAL diese beiden Aufgaben tatsächlich in einem einzigen Durchlauf ab. Dazu wird während der Analyse jeder Pfad im Workflow mit einer eigenen Kennung versehen, so dass nach der Analyse die gefundenen Schwachstellen genau einem Pfad zugeordnet werden können. Weil jede auf einem Pfad gefundene Aktivität bzw. Transition sofort analysiert wird, kann die bisherige Auswertung der Regeleinhaltung jederzeit an den Tokens in den Regelvorlagen abgelesen werden.

Alle Pfade eines Workflows zu analysieren bzw. zu ermitteln ist nur für Workflows ohne Schleifen trivial. Sobald ein Workflow-Modell Schleifen enthält, besteht die Gefahr, dass die Analyse immer wieder dieselbe Schleife durchläuft und daher nie zum Ende gelangt. Das nachfolgende Beispiel verdeutlicht das Problem.

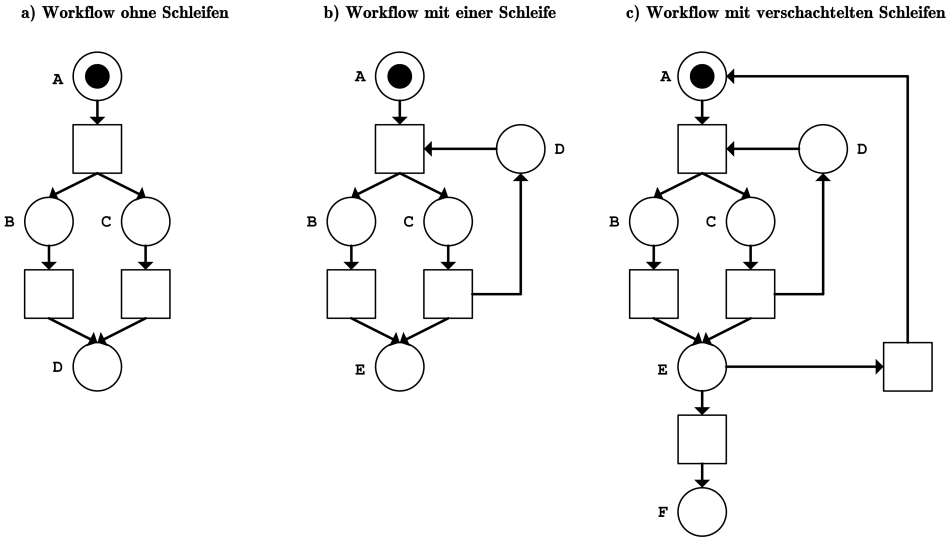


Abbildung 3.32: Beispiele für Workflows ohne und mit Schleifen

Beispiel. Der Workflow in Abbildung 3.32a enthält genau zwei Pfade, die leicht durch eine Tiefensuche identifiziert werden können. Dazu genügt es, bei jedem verzweigenden Element alle abgehenden Pfade weiter zu untersuchen. Im Beispiel besteht ein Pfad aus der Platzfolge A-B-D, der andere aus der Folge A-C-D.

Dieser einfache Ansatz würde die Analyse des Workflow-Modells in Abbildung 3.32b nie beenden, da außer den Folgen A-B-E, A-C-E und A-C-D-B-E auch die endlose Folge A-C-D-C- usw. untersucht werden würde. Es muss also erkannt werden, ob ein Platz bereits besucht wurde. Eine Lösung zur Analyse schleifenenthaltender Workflows besteht darin, jeden Platz nur maximal zweimal zu besuchen und die Folge abzubrechen,

wenn ein Platz darin häufiger angetroffen wird. Die beiden längsten Folgen im Beispiel Abbildung 3.32b lauten bei dieser Analyse A-C-D-C-E und A-C-D-B-E. Längere Folgen sind ausgeschlossen, da sie Plätze mehr als zweimal enthalten würden.

Für Workflows mit verschachtelten Schleifen wie in Abbildung 3.32c gezeigt genügt aber auch dieser Ansatz nicht, da im Vorhinein bekannt sein müsste, wie tief die Verschachtelung ist. Die Folge A-C-D-C-E-A-C-E-F würde, obwohl sie jede Schleife nur einmalig durchläuft, mit der Begrenzung „Plätze dürfen nur zweimalig enthalten sein" nicht analysiert, weil der Platz C dreimalig besucht werden müsste. ■

Im allgemeinen Fall bietet sich für Workflows, die verschachtelte Schleifen enthalten können, die Verwendung von Pfad- bzw. Folge-Prefixen an. Die Idee dabei ist, das Wiederholen einer bestimmten Schleife zu erkennen und nur einmalig zuzulassen, ohne die Entscheidung nur vom mehrfachen Besuch einzelner Plätze abhängig zu machen. Anhand des Pfad-Prefixes, also der Folge bisher besuchter Plätze, lässt sich feststellen, wie häufig eine Schleife (erkennbar als Teilfolge) bereits abgearbeitet wurde. Dieser Analyseansatz lässt sich leicht parameterisieren, um Schleifen nicht nur einmalig, sondern n-malig zuzulassen. Eine Obergrenze muss aber angegeben werden, damit keine Endlosschleifen entstehen.

3.4.3 Lokalisierung von Schwachstellen

Zu jedem Zeitpunkt während des Suchlaufs lassen sich die gefundenen Schwachstellen im Workflow an den Tokens der Regel-Petrinetze ablesen. Allerdings ist dabei zu beachten, dass einige Prüfungsvorlagen aus einem Fehlerzustand wieder in einen Regelkonformität signalisierenden Zustand gelangen können, wie beispielsweise die Vorlage **X löst Y aus**. Nach dem Auftreten einer Aktivität X verbleibt die Vorlage bis zur Entdeckung der Aktivität Y oder bis zum Ende des Workflows in einem Fehlerzustand; wird die Aktivität Y gefunden, ändert sich der Zustand und zeigt die Regeleinhaltung an. Vorlagen, die sich auf mehrere Elemente beziehen, können also erst bei Erreichen des Workflow-Endes abschließend ausgewertet werden.

Strukturelle Schwachstellen werden durch REVEAL eindeutig lokalisiert, indem das auslösende Workflow-Element angegeben wird. Unvollständige Workflow-Modelle, die nicht alle zur Auswertung der vorgegebenen Regeln benötigten Information wie z.B. über die Ressourcen oder Orte im Workflow enthalten, erlauben nur die Warnung vor möglichen Schwachstellen im Modell. Die Entscheidung, ob es sich tatsächlich um eine Schwachstelle handelt, ist dann erst mit zusätzlichen Informationen zu treffen.

Da für jede aufgestellte Regel ein eigenes Regel-Petrinetz erzeugt wird, ist die Zuordnung von Regel-Petrinetzen zu Regeln nach dem Workflow-Suchlauf unkompliziert. Basierend auf der eindeutigen Kennzeichnung von Regeln und Petrinetzen durch einmalige IDs ist die Zuordnung automatisiert möglich.

Dank der Verwendung von eindeutigen Token-IDs, die die direkte Zuordnung zu genau einem Workflow-Pfad liefern, kann ein Regel-Petrinetz für die Analyse mehrerer bzw. aller Workflow-Pfade benutzt werden, wie im nächsten Beispiel gezeigt.

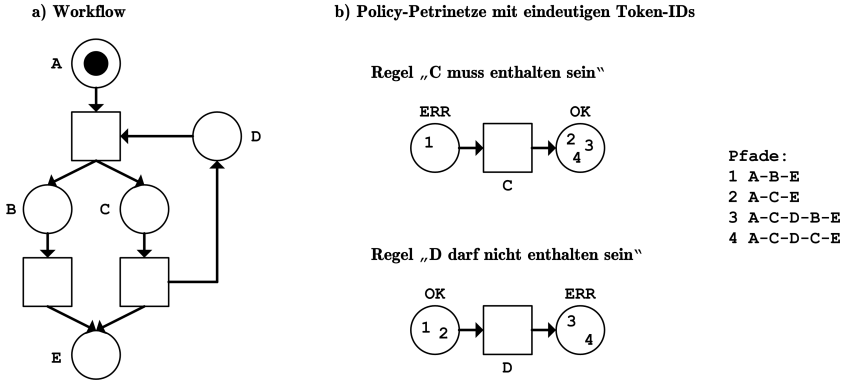


Abbildung 3.33: Beispiel für die Pfadzuordnung eindeutiger Token-IDs

Beispiel. Der Beispiel-Workflow in Abbildung 3.33a enthält vier Pfade (zwei ohne, zwei mit Schleife). Für die aufgestellten Regeln, nämlich „C muss enthalten sein" und „D darf nicht enthalten sein" ergibt sich die in Abbildung 3.33b gezeigte Tokenbelegung der Regel-Petrinetze, wobei jede Zahl für die ID eines Tokens steht. Die Pfadzuordnung kann entweder indirekt über die Token-IDs erfolgen, wie der besseren Lesbarkeit wegen in der Abbildung dargestellt, oder aber direkt über die Token-IDs, die dann aus den Folgen der Pfade bestehen. Das Token 3 beispielsweise hätte die Kennung ACDBE. Da jede Folge einmalig ist, sind auch die so entstehenden Token-IDs eindeutig.

Wie unmittelbar aus den Regel-Petrinetzen ersichtlich, verstößt der erste Pfad A-B-E gegen die Regel „C muss enthalten sein", während die restlichen Pfade sie einhalten. Die Regel „D darf nicht enthalten sein" wird hingegen von den Pfaden A-B-E und A-C-E eingehalten, nicht aber von den beiden durch die Schleife führenden Pfaden. ■

Das Beispiel verdeutlicht, wie leicht die Zuordnung von Regel-Petrinetzen zu Workflow-Pfaden ist. Nach einem Suchlauf erspart eine automatisierte Auswertung der Regel-Petrinetze die manuelle Nachbearbeitung, indem REVEAL anhand von zwei Zuordnungen direkt im Workflow anzeigt, welche Elemente eine Schwachstelle verursachen bzw. potentiell gegen eine Regel verstoßen. Dazu wird erstens für jedes Regel-Petrinetz über die oben erklärte eindeutige Kennzeichnung die entsprechende Regel ermittelt. Dann wird zweitens über die Token-ID der betroffene Pfad identifiziert und die verletzte Regel an das auslösende Element gehängt, wie im folgenden Beispiel beschrieben. Tatsächlich lässt sich diese Auswertung, das heißt die Markierung problematischer Workflow-Elemente mit den durch sie verletzten Regeln, bereits zeitsparend während der Analyse durchführen, sobald ein Regel-Petrinetz einen endgültigen Fehlerzustand erreicht hat.

Beispiel. Nach dem Workflow-Suchlauf zeigt das Regel-Petrinetz in Abbildung 3.34a an, dass die Pfade eins, drei und vier gegen die Regel „D darf nicht enthalten sein" verstoßen. Die den Verstoß auslösende Aktivität D wird über die Zuordnung Netz-zu-Regel

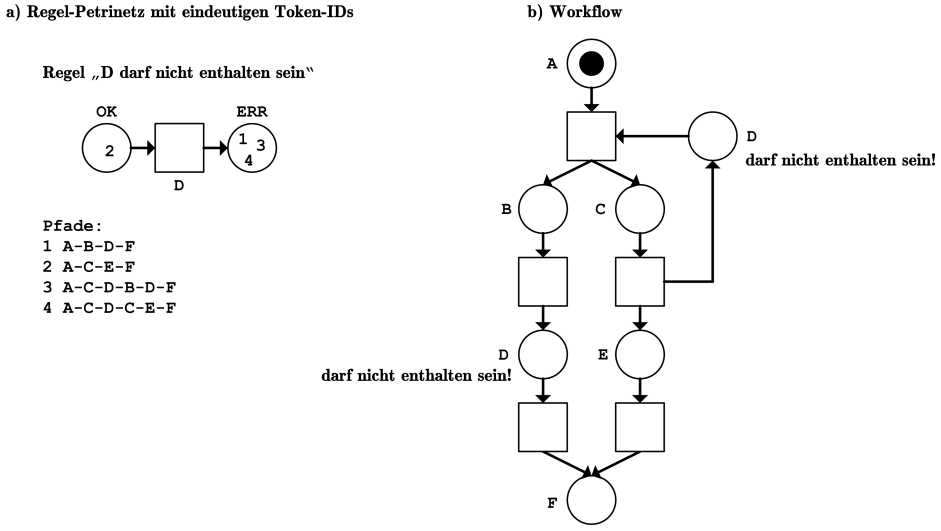


Abbildung 3.34: Beispiel für die Workflow-Zuordnung von Regeln

und Token-zu-Pfad identifiziert. Tokens müssen nicht gesondert einem Netz zugeordnet werden, da sie Teil eines Netzes sind. Im Workflow-Modell wird dann die verbotene Aktivität D automatisiert mit der entsprechenden Regel markiert. Das Ergebnis der Zuordnung ist in Abbildung 3.34b zu sehen. ■

Mehrfach im Workflow auftretende Elemente können durch eindeutige IDs unterschieden werden, so dass exakt nur die Elemente markiert werden, die eine Vorlage als problematisch erfasst hat.

Anhand desselben Prinzips werden nicht nur alle Regelelemente mit ihren Angaben (Aktivitäten und Daten sofern angegeben jeweils mit Modalität, Ort, Reihenfolge, Ressource, Verwendung und Zeitlimit) geprüft und ggf. zur Markierung von Schwachstellen im Workflow-Modell genutzt, sondern auch die Korrektheitseigenschaften. Zur Vermeidung von Deadlocks darf beispielsweise keine UND-Vereinigung auf eine ODER-Verzweigung folgen. Wird eine solche Kombination im Modell gefunden, wird sowohl die auslösende ODER-Verzweigung wie auch die letztlich den Deadlock verursachende UND-Vereinigung im Workflow-Modell markiert.

3.5 Implementierung

REVEAL ist prototypisch als Java-Anwendung implementiert. Als Format für die Ein- und Ausgabedateien wurde die Petrinet-Markup-Language (PNML) gewählt. Diese liegt in der Version 1.0 als ISO-Standard vor und umfasst sowohl High-Level-Petrinetze (inklusive typisierter Plätze) wie auch einfache, sogenannte P/T-Netze.

PNML beschreibt Petrinetze mit Hilfe von XML. Die Petrinetz-Elemente Platz, Transition und Kante finden ihre direkte Entsprechung als <place>, <transition>

und <arc>. Die Listings 1, 2, 3 und 4 zeigen die für die Implementierung verwendeten PNML-Elemente und Attribute.

Listing 1: PNML-Netz

```
<net id='123'>
<!-- places, transitions,
  arcs -->
</net>
```

Listing 2: PNML-Platz

```
<place id='p5' type='type1'>
 <name><text>Place P
 </text></name>
</place>
```

Listing 3: PNML-Transition

```
<transition id='t3'>
 <name><text>Transition T
 </text></name>
</transition>
```

Listing 4: PNML-Arc

```
 <arc id='17' source="p5"
 target="t3"/>
```

Leider unterstützt der PNML-Standard farbige bzw. typisierte Tokens nicht. Plätzen kann zwar ein erlaubter Typ zugewiesen werden, aber nur genau ein solcher Typ. Die Definition verschiedener Tokens in einem Platz ist damit nicht möglich.

Für Erweiterungen sieht der PNML-Standard das XML-Element *tool-specific* vor. Damit lassen sich beliebige XML-Angaben einem PNML-Element zuordnen. Listing 5 zeigt, wie eine solche <tool-specific>-Angabe für eine Transition angegeben wird.

Listing 5: PNML-Transition mit ToolSpecific-Angabe

```
<transition id='3'>
 <name><text>Transition T</name></text>
 <toolspecific tool='xy' version='1'> <location='freiburg'/>
   <inputdata>vorname</inputdata>
   <inputdata>nachname</inputdata>
   <inputdata>plz</inputdata> <inputdata>stadt</inputdata>
   </inputdata>
 </toolspecific>
</transition>
```

Listing 6: PNML-Transition mit FiringRule

```
<transition id='3'> <name><text>Transition T</name></text>
 <toolspecific tool='xy' version='1'>
 <firingRule id='72' priority='1'>
      <consumption>
        <material tokenType="SIG_FLAG" amount="1"/>
      </consumption> <production>
        <material tokenType="SIG_ERR" amount="1"/>
      </production>
 </firingRule> </toolspecific>
</transition>
```

Genau wie die in Listing 5 gezeigten Zusatzangaben, in diesem Fall zum Ort und zu den Eingabedaten einer Transition, lassen sich auch Angaben zum erweiterten Feuerverhalten einer Transition definieren. Der Begriff **Feuerregel** wird im Folgenden für die Lenkung des Feuerverhaltens verwendet; er bezieht sich *nicht* auf die durch die Petrinetze erfassten Regelvorlagen. Listing 6 enthält die Definition einer Feuerregel als *toolspecific*-Angabe.

Feuerregeln bestehen neben einer eindeutigen Kennung und einer Prioritätsangabe aus zwei Teilen. Der erste Teil (*consumption*) beschreibt, welche Tokens in welchen Eingangsplätzen liegen müssen, damit die Transition aktiv ist. Diese Tokens werden entsprechend beim Feuern der Transition verbraucht, das heißt aus den Eingangsplätzen entfernt. Der zweite Teil (*production*) gibt an, welche Tokens in welchen Ausgangsplätzen beim Feuern der Transition erzeugt werden.

Sowohl der verbrauchende wie auch der erzeugende Teil werden anhand der *material*-Angabe definiert. Darin wird festgelegt, welcher Typ von Tokens in welcher Anzahl verbraucht oder erzeugt wird. Zur weiteren Präzisierung kann der Typ der in Frage kommenden Eingangs- und Ausgangsplätze definiert werden (bei den in Abschnitt 3.3 vorgestellten Vorlagen z.B. wird zwischen den Typen OK und ERR unterschieden). Bei Bedarf kann sogar ein einzelner Platz anhand seiner eindeutigen Kennung angegeben werden. Dies ist aber nur in den Fällen nötig, in denen eine Transition mit mehreren Plätzen des gleichen Typs verknüpft ist. Ansonsten genügt die Unterscheidung der Plätze anhand des Platztyps.

3.5.1 Regel-Transformation

Die Implementierung der Regel-Transformation verfolgt dasselbe Prinzip wie die der Workflow-Transformation: eine XML-Eingabedatei wird mittels eines Parsers in ihre Elemente zerlegt, aus denen dann nach und nach ein Petrinetz aufgebaut wird, welches als PNML-Ausgabedatei gespeichert wird. Der besseren Übersichtlichkeit wegen verwendet der Prototyp eine eigene Regelsprache, deren Elemente direkt den in Tabelle 3.1 definierten entsprechen. Das Einlesen anderer Regelsprachen wie etwa XACML oder WS-Policy lässt sich durch Anpassung der Elementnamen ebenfalls umsetzen, erfordert aber wegen der Komplexität der Standards einigen Mehraufwand.

In der Tabelle 3.3 sind links die von Reveal berücksichtigten Regel-Elemente aufgelistet, rechts die jeweilige Übersetzung in PNML-Vorlagen, wie sie im Abschnitt 3.3 vorgestellt wurden.

Listing 7: Beispielregeln vor Transformation (XML)

```
1 <activity name="Wasser versalzen">
2   <modality>must not</modality> </activity>
3 <activity name="Wasser kochen">
4   <before>Wasser in Teekanne giessen</before> </activity>
```

Tabelle 3.3: Element-Übersetzung Regel in PNML

Regel-Element	**Übersetzung in PNML-Vorlagen**
Aktivitäten: activity	Name der Aktivität
Daten: input, output	Eingabe und Ausgabe der Aktivität
Modalität: must not, must	verbotene und geforderte Elemente
Orte: location	Ortsangaben
Reihenfolge: before after ifbothafirst	„y erfordert x“ „x löst y aus“ „wenn x und y, dann x zuerst“
Ressourcen: role service	(nicht) berechtigte Rolle (nicht) berechtigter Dienst
Verwendung: before after	„y erfordert x“ „x löst y aus“
Zeitpunkte: before limittype limit after limittype limit	 „y erfordert x“, min/max, Zeitraum „x löst y aus“, min/max, Zeitraum

Listing 8: Beispielregeln nach Transformation (PNML)

```
1  <pnml><net id="beispiel"> <place id="p0" type="OKplace">
2    <name><text>ok</text></name>
3    <graphics> <position x="100" y="0"/>
4     <dimension x="20" y="20" /> </graphics>
5    <toolspecific tool="plc2pnml" version="alpha">
6     <ruleName>must not Wasser versalzen</ruleName>
7    </toolspecific> </place> [...]
8  <transition id="t0">
9    <name><text>Wasser kochen</text></name>
10   <graphics> <position x="300" y="50" />
11    <dimension x="20" y="20" /> </graphics>
12   <toolspecific tool="plc2pnml" version="alpha">
13    <firingRule id="fr0" priority="1">
14     <production> <material tokenType="SIG_ERR" amount="1"
15       placeType="ERRplace" /> </production> </firingRule>
16   </toolspecific> </transition> [...]
17 <arc id="a0" source="p0" target="t0"/> [...] </net></pnml>
```

Beispiel. Die in Listing 7 gezeigten Regeln verbieten die Aktivität *Wasser versalzen* und fordern, dass die Aktivität *Wasser kochen* vor der Aktivität *Wasser in Teekanne giessen* stattfinden muss. Entsprechend enthält das Listing 8 das Regel-Petrinetz in PNML. Aus Platzgründen enthält das Listing nicht alle Plätze, Transitionen und Kanten. ■

3.5.2 Workflow-Transformation

Für die Modellierung von Workflows steht eine Vielzahl graphischer und logischer Notationen zur Verfügung. Wie im Abschnitt 3.2 diskutiert, lassen sich alle Workflows in der für die Analyse mit Reveal nötigen Petrinetz-Form darstellen. Für die prototypische Implementierung wurde BPEL (Organization for the Advancement of Structured Information Standards (OASIS), 2007) als Ausgangsnotation gewählt. Workflows, die in anderen Notationen wie der ebenfalls weit verbreiteten BPMN oder XPDL vorliegen, können mit einer angepassten Transformationsroutine ebenfalls analysiert werden. Typischerweise genügt dazu das Austauschen der BPEL-Schlüsselwörter im Parser gegen die entsprechenden Schlüsselwörter der gewählten anderen Notation.

Die Übersetzung von BPEL-Workflows in Workflow-Petrinetze im PNML-Format erfolgt mittels eines Parsers, der die XML-Elemente wie beispielsweise *invoke* für Dienstaufrufe einliest, die dann in die entsprechenden PNML-Elemente Plätze, Transitionen und Kanten überführt werden. In der Tabelle 3.4 sind links die von Reveal berücksichtigten BPEL-Elemente aufgelistet, rechts die jeweilige Übersetzung in PNML-Elemente.

Um eine präzisere Analyse des Datenflusses zu ermöglichen, werden außerdem die zu den BPEL-Aktivitäten gehörenden WSDL-Angaben eingelesen und als Tags der jeweiligen PNML-Transitionen gespeichert. Für Aktivitäten, die Nachrichten senden oder empfangen, kann Reveal dann zusätzlich zu den Namen der Nachrichten auch die Namen der in den Nachrichten enthaltenen Variablen verwenden. Die eingelesenen WSDL-Angaben sind mit ihren PNML-Übersetzungen in der Tabelle 3.5 zusammengefasst.

Beispiel. Patientendaten sollen für eine wissenschaftliche Studie verwendet werden. HIPAA §164.514b2i schreibt vor, dass die Postleitzahlen aller Patienten aus einem PLZ-Gebiet mit weniger als 20.000 Einwohnern auf drei Stellen gekürzt werden. Ohne die aus den WSDL-Angaben gezogenen Informationen über die Variablennamen kann für die Nachricht „Patientenakte“ nicht genau analysiert werden, ob das Element *PLZ* enthalten ist und anonymisiert wurde. Da aber nicht nur der BPEL-Nachrichtenname, sondern auch die WSDL-Angaben über die Nachrichtenvariablen in PNML übersetzt und von Reveal zur Analyse herangezogen werden, ist die Untersuchung auf WSDL-Variablen- und nicht nur auf WSDL-Nachrichtenebene möglich. ■

Zur Veranschaulichung der Workflow-Transformation dienen die Abbildung 3.35 sowie die Listings 9 und 10.

Listing 9: Beispiel-Workflow vor Transformation (BPEL)

```
1 <bpel:process> <bpel:sequence name="main">
2   <bpel:receive name="acceptRequest"/>
3   <bpel:flow><empty name="a"/><empty name="b"/></bpel:flow>
4   <bpel:invoke name="sendReply"/>
5 </bpel:sequence> </bpel:process>
```

Tabelle 3.4: Element-Übersetzung BPEL in PNML

BPEL-Element	**Übersetzung in PNML-Elemente**
Struktur:	
process	place (Start), arc, transition (Start), (process-Inhalt) transition (Ende), arc, place (Ende)
Zuweisung:	
assign	arc, transition, arc, place (mit Variablen als Tags)
Grund-aktivitäten:	
empty	arc, transition, arc, place
invoke	arc, transition, arc, place (mit Attributen als Tags)
receive	arc, transition, arc, place (mit Attributen als Tags)
reply	arc, transition, arc, place (mit Attributen als Tags)
wait	arc, transition, arc, place
Strukturierte Aktivitäten:	
flow	arc, transition (parallel verzweigend), (flow-Inhalt), transition (parallel vereinigend), arc place
pick	arc, transition (1-aus-n), (pick-Inhalt), transition (1-aus-n), arc, place
sequence	arc, transition, (sequence-Inhalt), transition, arc, place
switch	arc, transition (auswählend), (switch-Inhalt), transition (auswählend), arc, place
while	arc, transition (Schleifenstart), (while-Inhalt), transition (Schleifenende), arc, place

Listing 10: Beispiel-Workflow nach Transformation (PNML)

```
1  <pnml> <net id="beispiel">
2  <place id="p0" type="input">
3    <name><text>start</text></name>
4    <initialMarking><text>1</text></initialMarking> </place>
5  <place id="p1"/> [...] <place id="p7"/>
6  <place id="p8" type="output">
7    <name><text>end</text></name> </place>
8  <transition id="t0"> <name><text>acceptRequest
9    </text></name> </transition> <transition id="t1"/>
10 <transition id="t2"> <name><text>a</text></name> </transition>
11 <transition id="t3"> <name><text>b</text></name> </transition>
12 <transition id="t4"/>
13 <transition id="t5"> <name><text>sendReply
```

Tabelle 3.5: Element-Übersetzung WSDL in PNML

WSDL-Element	**Übersetzung in PNML-Elemente**
operation:input:message parts	<input> Transitionsattribut
operation:output:message parts	<output> Transitionsattribut

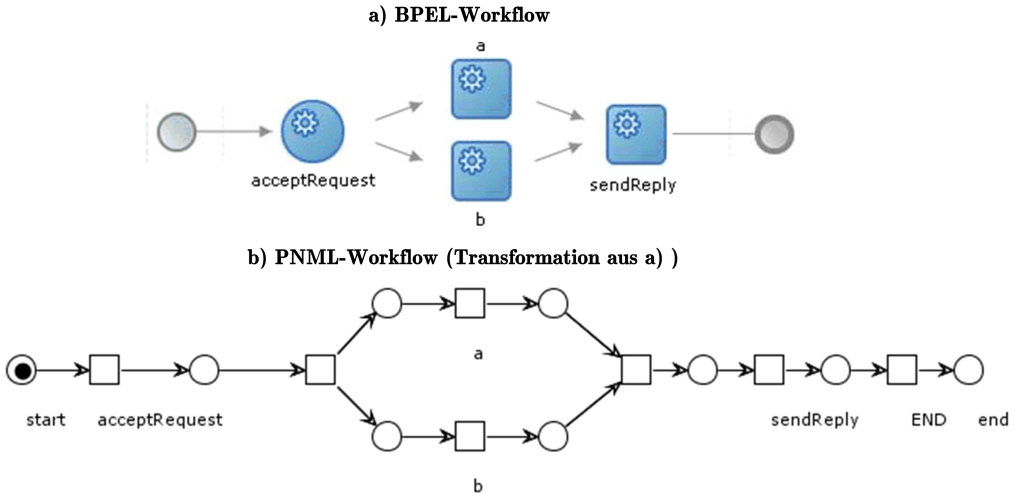


Abbildung 3.35: Beispiel für die Workflow-Transformation

```
14   </text></name> </transition>
15 <transition id="t6"> <name><text>END</text></name> </transition>
16 <arc id="a0" source="p0" target="t0"/> [...]
17 <arc id="a15" source="t6" target="p8"/>
18 </net> </pnml>
```

Beispiel. Der in Abbildung 3.35 gezeigte Workflow besteht aus der Aktivität *acceptRequest*, gefolgt von zwei parallelen Pfaden mit jeweils einer Aktivität, *a* im oberen Pfad, *b* im unteren Pfad, und wird abgeschlossen von der Aktivität *sendReply*. Die graphische Ähnlichkeit der BPEL- und Petrinetz-Darstellung ist deutlich sichtbar. Da in einem Petrinetz Transitionen nicht untereinander durch Kanten verbunden sein dürfen, müssen Hilfsplätze eingeführt werden. Diese werden üblicherweise als Zustandsanzeiger interpretiert: liegt ein Token im Platz hinter der Aktivität *a* vor der Aktivität *sendReply*, ist die aktuelle Workflow-Ausführung bis genau dorthin fortgeschritten.

Im Listing 9 ist der BPEL-Workflow, der um die für dieses Beispiel irrelevanten zusätzlichen BPEL-Angaben gekürzte wurde, in seiner XML-Form angegeben. Entsprechend enthält das Listing 10 den PNML-Workflow ebenfalls als XML. Sich nur in den Platz- bzw. Kanten-IDs unterscheidende Angaben wurden herausgekürzt. ■

3.5.3 Analyse

Aus der Workflow- und Regel-Transformation entstehen PNML-Dateien, die für die Prüfung als Petrinetz-Objekte in Java eingelesen werden. Die gewählten Klassen orientieren sich direkt an den PNML-Elementen.

- Netze: PNML-*nets* werden zu PetriNet-Objekten
- Plätze: PNML-*places* werden zu Place-Objekten
- Transitionen: PNML-*transitions* werden zu Transition-Objekten
- Kanten: PNML-*arcs* werden zu Arc-Objekten
- Tokens: aus dem PNML-*initialMarking* bzw. der von REVEAL verwendeten *token* Angabe werden Token-Objekte

Die Abbildung 3.36 zeigt die von REVEAL definierten Java-Klassen für Petrinetze.

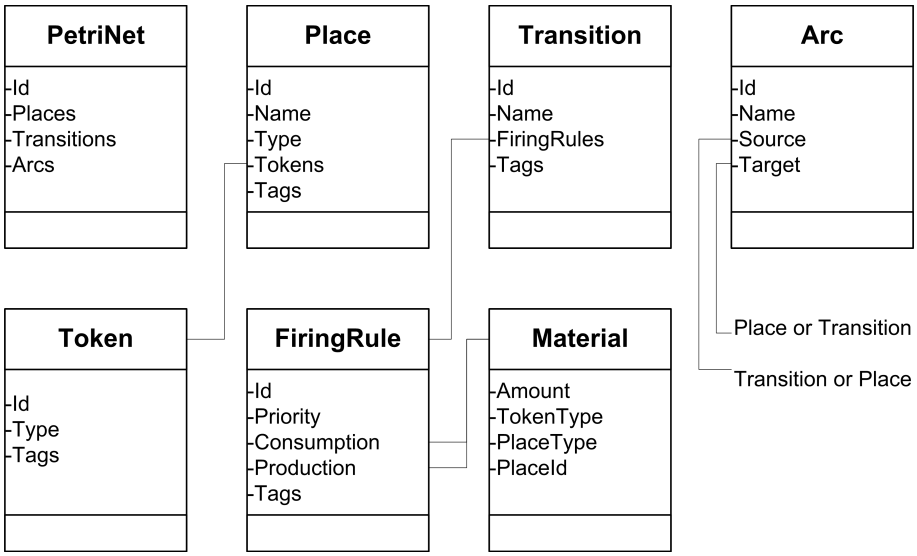


Abbildung 3.36: Klassendiagramme für REVEAL-Petrinetze

Sobald der Workflow und die Regeln als Objekt-Repräsentation vorliegen, kann mit der Analyse der Workflow-Pfade begonnen werden. Zwei Funktionen leiten die Analyse komplett durch alle Workflow-Pfade, inklusive dem frei wählbaren n-maligen Abarbeiten von Schleifen auf den Pfaden (vgl. Abschnitt 3.4.2). Die Funktion *visitPlace* (siehe Listing 2) wird aufgerufen, um die auf eine Transition folgenden Plätze dem aktuell untersuchten Workflow-Pfad hinzuzufügen. Die Funktion *visitTransition* (siehe Listing 3) übernimmt die gleiche Aufgabe für auf einen Platz folgende Transitionen.

```
visitPlace(place, token)
Input : Place to be analyzed, token that is reaching this place
Output : Number of paths; main task is calling 'visitTransition'
Get current path prefix from token
if null==prefix then
    create new, empty prefix;
for loopcounter = 0 to MAXLOOPS do
    find current path prefix in current path
    if not found break
if loopcounter == MAXLOOPS then
    return
// do not analyze loops more often than MAXLOOPS allows
Append current place to path prefix
if no transition follows then
    increase path counter
    return
// count number of workflow paths
if just one transition follows then
    visitTransition(next transition, current token)
else  foreach following transition do
    Create new token with unique Id
    visitTransition(next transition, new token)
```

Algorithmus 2: Funktion **visitPlace** zur Pfadanalyse

```
visitTransition(transition, token)
Input : Transition to be analyzed, token that is reaching this transition
Output : Number of paths; main task is triggering the policy check
triggerPolicy(current transition,current token)
if no place follows then
    increase path counter
    return
// count number of workflow paths
if just one place follows then
    visitPlace(next place, current token)
else
    foreach following place do
        Create new token with unique Id
        visitPlace(next place, new token)
```

Algorithmus 3: Funktion **visitTransition** zur Pfadanalyse

Beide Funktionen arbeiten rekursiv, also dem Prinzip der Tiefensuche nach immer weiter in Pfade absteigend, bevor der nächste Pfad (das heißt, die nächste vom aktuellen Element ausgehende Verzweigung) analysiert wird.

Für jede auf einem Pfad durch das Workflow-Petrinetz gefundene Transition, also für jede Aktivität im Workflow, wird geprüft ob eine gleichnamige Transition in einem Regel-Petrinetz vorhanden ist. Wenn ja, und wenn die Transition aktiv ist, wird das Regel-Petrinetz entsprechend der vorhandenen Tokens und den für die Regel-Transition definierten Feuerregeln geschaltet. Besonders wichtig ist hierbei die Unterscheidung der Tokens anhand ihrer eindeutigen IDs. Um nicht für jeden Pfad des Workflows sämtliche Regelvorlagen neu instanziieren zu müssen, wird pro Vorlage nur eine Instanz gebildet, innerhalb derer für jeden Pfad unterschiedliche Tokens erzeugt werden. Dazu bekommt die Funktion *visitTransition* während der Pfadanalyse immer das jeweils aktuell verwendete Token übergeben und reicht die ID des Tokens an die Prüffunktion *triggerPolicy* weiter. Deren Pseudocode ist im Listing 4 aufgeführt.

```
triggerPolicy(transition, token)
Input : Workflow transition, current token
Output : No direct output; triggers the firing of policy nets, if active
if policy net has a matching transition then
    // token ID serves to determine current path
    if policy transition is active for current path then
        Fire policy transition
```

Algorithmus 4: Funktion **triggerPolicy** zur Schaltung der Regel-Petrinetze

Innerhalb der Regel-Petrinetze gibt es zwei Sorten von Plätzen. Beide zeigen den aktuellen Status der Regelauswertung an, aber nur eine Sorte enthält bereits vor Ende des Workflows bzw. der Pfadanalyse ein endgültiges Ergebnis. Die andere Sorte zeigt ein vorübergehendes, nur potentiell endgültiges Ergebnis an. Vor dem Hintergrund, dass nur ein einzelner, nicht vorausschauender Suchlauf stattfindet, ist das logisch: Regeln, die eine Beziehung mehrerer Elemente beinhalten, wie beispielsweise eine bestimmte Reihenfolge zweier Aktivitäten, können erst nach dem Suchlauf bewertet werden, weil es nach dem Auftreten des ersten Elements unter Umständen bis zum Workflow-Ende unklar bleibt, ob ein weiteres reglementiertes Element bzw. die zweite Aktivität noch auftritt oder nicht.

Plätze mit endgültiger Bewertung sind daran zu erkennen, dass sie keine abgehenden Kanten besitzen; die Unterscheidung ist in Abbildung 3.37 skizziert. Beim Erreichen eines Platzes mit endgültiger Bewertung markiert Reveal die auslösende Workflow-Transition mit der verletzten Regel und den vorhandenen Pfadinformationen wie etwa den zuvor ausgeführten Aktivitäten. Plätze mit einer möglicherweise nur vorübergehenden Bewertung führen ebenfalls zu einer Markierung, wenn die Bewertung eine Schwachstelle anzeigt. Die Markierung wird aber ggf. entfernt oder „entschärft“, wenn die Bewertung im weiteren Laufe der Analyse geändert wird.

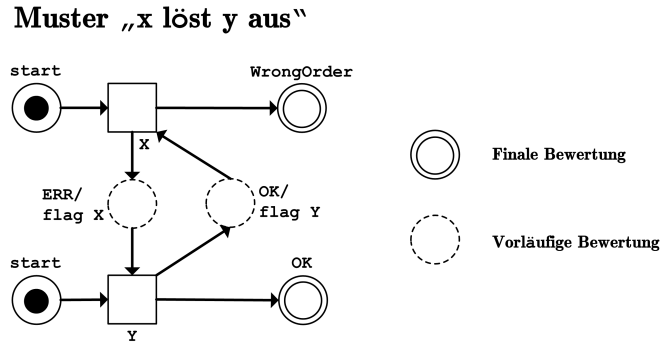


Abbildung 3.37: Plätze mit endgültiger und vorübergehender Bewertung

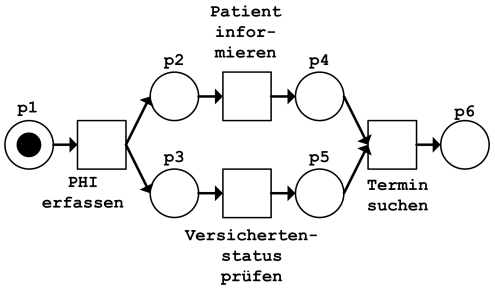


Abbildung 3.38: Beispiel für Plätze mit endgültiger und vorübergehender Bewertung

Beispiel. Der in Abbildung 3.38 dargestellte Workflow der Patienten-Neuaufnahme besteht aus vier Schritten. Zwischen der ersten Aktivität *PHI erfassen* und der letzten Aktivität *Termin suchen* finden die beiden Aktivitäten *Patient informieren* und *Versichertenstatus prüfen* parallel statt. Für die Regel „*PHI erfassen* ist verboten" wird bereits nach der ersten Aktivität bzw. Transition im Workflow-Petrinetz ein endgültiger Bewertungszustand im Regel-Petrinetz erreicht. Gilt stattdessen die Regel „*PHI erfassen* löst *Patient informieren* aus", wird das Regel-Petrinetz zunächst in einen vorübergehenden Bewertungszustand geschaltet, bevor es nach der Aktivität *Patient informieren* in den endgültigen *OK*-Zustand geschaltet wird. Ein Regel-Petrinetz für die Regel „*Patientendaten weitergeben* verboten" würde hingegen während der gesamten Analyse im Ausgangszustand bleiben und entsprechend die Bewertung *OK* signalisieren (vgl. Vorlage „darf nicht" in Abbildung 3.16). ■

4 Evaluation

4.1 Workflow-Prüfung durch Reveal

Die Evaluation der Prüfung hinsichtlich der in Abschnitt 2.5 aufgestellten Anforderungen erfolgt dreigeteilt. Zunächst wird die Korrektheitsprüfung ausgewertet. An zweiter Stelle steht die Datenfluss-Analyse, und abschließend wird die Kontrollfluss-Analyse ausgewertet. Zur Auswertung wird das in Abschnitt 2.3 vorgestellte Fallbeispiel herangezogen. Dessen Workflow aus Abbildung 2.9 ist in Petrinetz-Form in Abbildung 4.1 dargestellt.

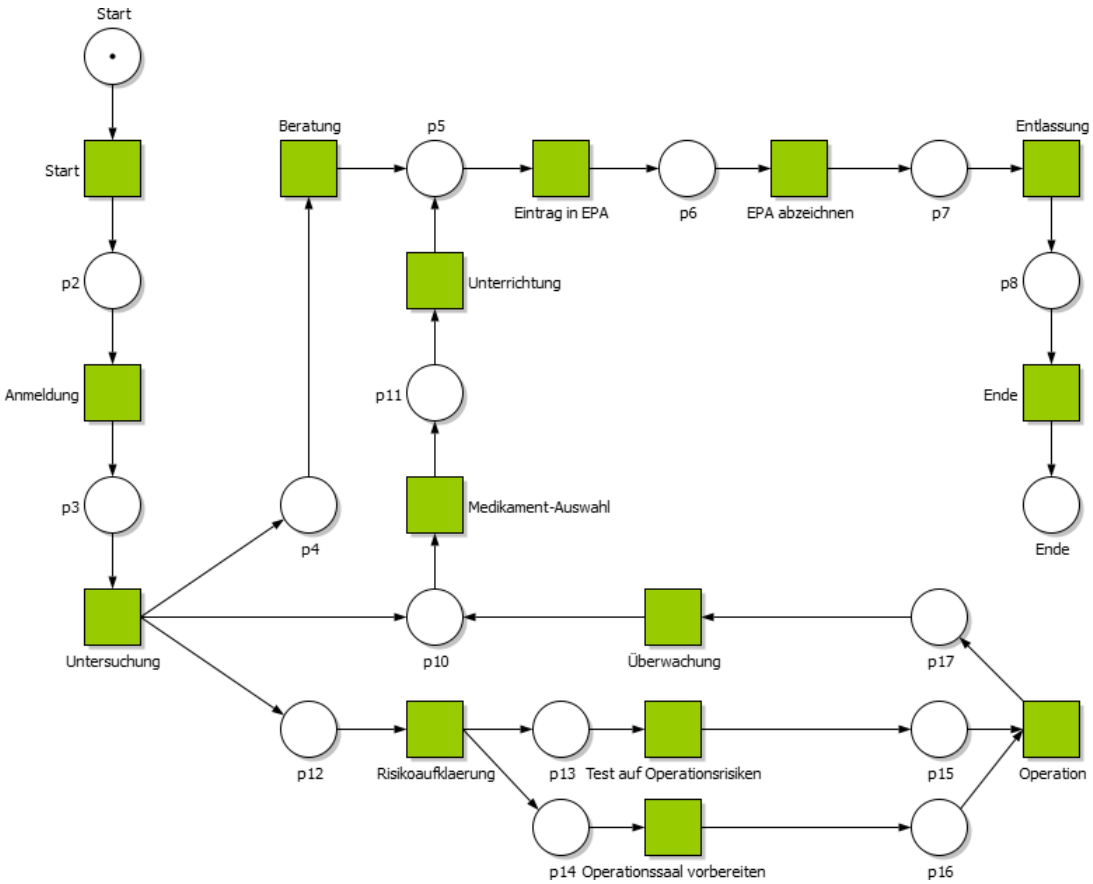


Abbildung 4.1: Fallbeispiel-Workflow als Petrinetz

Der Workflow enthält drei Pfade, auf die in den untenstehenden Abschnitten anhand

ihrer eindeutigen Kennungen Bezug genommen wird. Alle Pfade beginnen gleichermaßen mit den Transitionen *Anmeldung* und *Untersuchung* und enden gleichermaßen mit den Transitionen *Eintrag in EPA, EPA abzeichnen, Entlassung*. Die unterschiedlichen Transitionen dazwischen lauten wie folgt.

- *Beratung*
- *Medikament-Auswahl, Unterrichtung*
- *Risikoaufklärung, Test auf OP-Risiken parallel zu OP-Saal vorbereiten, OP, Überwachung*

Für jedes der neun Regel-Elemente, die die Prüfung erfassen muss, wurde eine entsprechende Regel aufgestellt. Zusätzlich wurden die Vorlagen zur Korrektheitsprüfung verwendet. Listing 11 enthält die in XML formalisierten Regeln, und zwar fünf bezüglich der Workflow-Aktivitäten und vier bezüglich der Workflow-Daten. Natürlichsprachlich lauten die Regeln wie folgt, wobei die Kürzel aus Tabelle 2.6 aufgegriffen werden:

- A.M: Die Aktivität *Anmeldung* muss im Workflow enthalten sein.
- A.O: Die Aktivität *Operation* muss im Operationssaal ausgeführt werden.
- A.R: Die Aktivität *Risikoaufklärung* muss vor der Aktivität *Operation* ausgeführt werden.
- A.S: Die Aktivität *Risikoaufklärung* muss von der Ressource (Rolle) *Arzt* ausgeführt werden.
- A.Z: Die Aktivität *Untersuchung* muss spätestens fünf Stunden nach der Aktivität *Anmeldung* ausgeführt werden.
- D.M: Das Datenelement *Unterschrift* muss im Workflow enthalten sein.
- D.O: Das Datenelement *EPA* darf nur lokal gespeichert werden.
- D.S: Das Datenelement *Unterschrift* darf nur von der Ressource (Rolle) *Arzt* erzeugt werden.
- D.V: Das Datenelement *Gebrauchsangaben* muss - sofern es erzeugt wird - von der Aktivität *Eintrag in EPA* verwendet werden.

Listing 11: Regeln für das Fallbeispiel

```
<allRules>
<!-- ***************************** -->
<correctness/>
<!-- ***************************** -->
<activityRules>
<!-- A.M -->
  <activity name="Untersuchung">
  <modality>must</modality> </activity>
<!-- A.O -->
  <activity name="OP">
  <location>Operationssaal</location> </activity>
<!-- A.R -->
  <activity name="Risikoaufklaerung">
  <before>OP</before> </activity>
<!-- A.S -->
  <activity name="Risikoaufklaerung">
  <role>Doctor</role> </activity>
<!-- A.Z -->
  <activity name="Untersuchung">
    <after limitType="max" limit="5 hours">Anmeldung</after>
  </activity>
</activityRules>
<!-- ***************************** -->
<dataRules>
<!-- D.M -->
  <data name="name"> <modality>must</modality> </data>
<!-- D.O -->
  <data name="EPA"> <location>local</location> </data>
<!-- D.S -->
  <data name="EPA"> <role>doctor</role> </data>
<!-- D.V -->
  <data name="gebrauchsangaben">
  <use>Eintrag in EPA</use> </data>
</dataRules>
</allRules>
```

In der oben gezeigten Form ist der Workflow regelkonform zu den aufgestellten Regeln. Zur besseren Veranschaulichung der Analyse werden im Folgenden bei Bedarf kleine Änderungen am Workflow vorgenommen bzw. Schwachstellen eingesetzt, die im Rahmen des jeweiligen Analyseaspekts diskutiert werden.

4.1.1 Korrektheit

Ein Workflow-Modell wird von Reveal als korrekt betrachtet, wenn es die in Abschnitt 2.5.1 aufgestellten Forderungen nach Erreichbarkeit aller Aktivitäten einerseits und Fertigstellung aller Aktivitäten bei Workflow-Ende andererseits erfüllt. Als Petrinetz-Eigenschaft ausgedrückt bedeutet dies, dass erstens keine UND-Vereinigung direkt auf eine ODER-Verzweigung folgen darf (würde zu einem Deadlock, also Nicht-Erreichung von Aktivitäten führen) und dass zweitens keine ODER-Vereinigung direkt auf eine UND-Verzweigung folgen darf (könnte „liegenbleibende“ Tokens im Petrinetz erzeugen).

Folgerichtig meldet Reveal wie in Abbildung 4.3 (Textmeldung am unteren Bildrand) gezeigt die nicht korrekte Kombination einer ODER-Verzweigung (in *t3*) mit einer UND-Vereinigung (*p5*), nachdem diese zu Testzwecken entsprechend manipuliert wurde (siehe Abbildung 4.2), und zwar auf allen drei Pfaden durch den Workflow. In der manipulierten Form könnte der Workflow nicht zu Ende geführt werden, da es bei Transition *t5* zu einem Deadlock kommen würde: zwei Tokens wären zur Fortführung benötigt, es steht aber aus der ODER-Verzweigung in *t3* nur eines zur Verfügung. Ansonsten, das heißt wenn statt der UND-Vereinigung eine ODER-Vereinigung folgt, ist der Workflow korrekt.

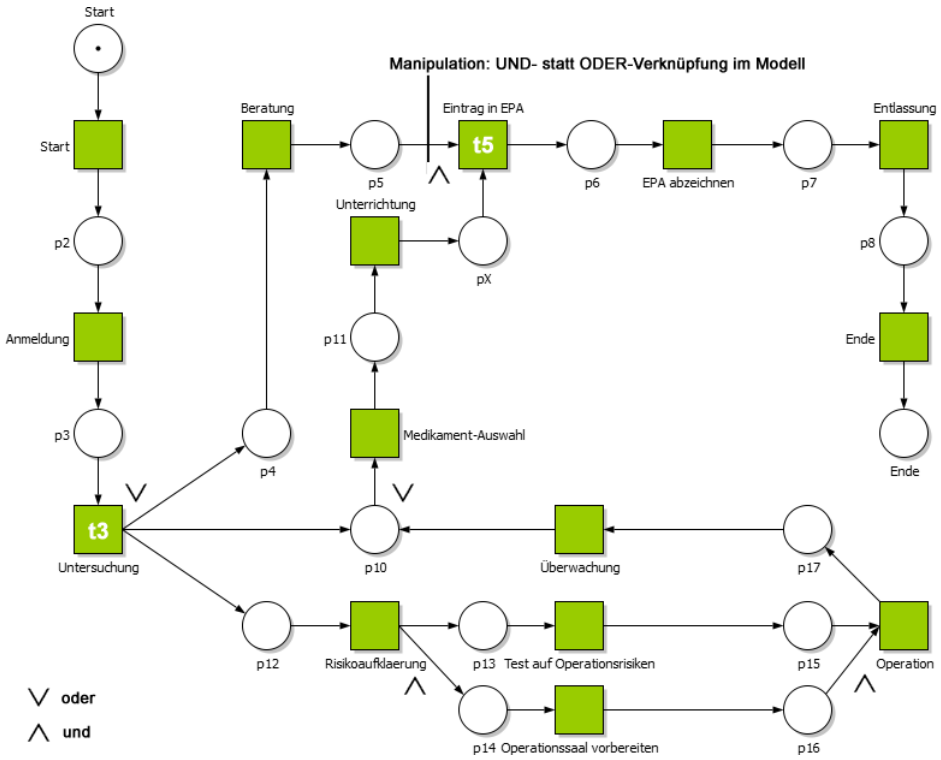


Abbildung 4.2: Inkorrekter Workflow

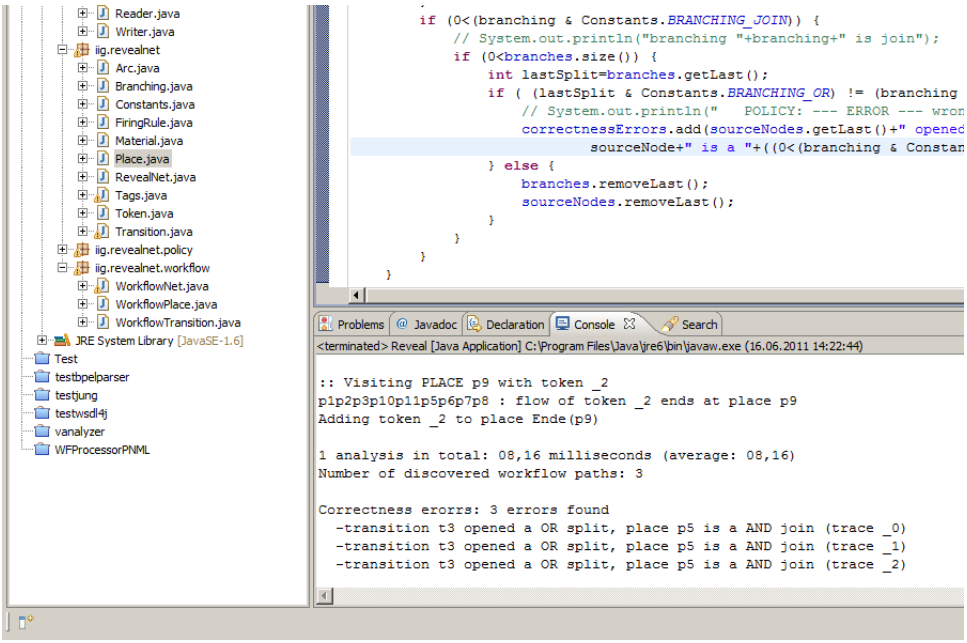


Abbildung 4.3: Korrektheitsprüfung mit REVEAL

Die Vorlagen zur Korrektheitsprüfung (vgl. Abschnitt 3.3.1) sind insofern nur eingeschränkt einsetzbar, als sie bei verschachtelten Verzweigungen falsche Warnungen liefern können. Der Prototyp prüft die Korrektheit eines Workflows daher anhand eines Stacks, auf dem während der Analyse die gefundenen Verzweigungen abgelegt werden. Folgt eine passende Vereinigung, wird die Verzweigung vom Stack entfernt. Ist die folgende Vereinigung von einem anderen Typ (folgt also UND auf ODER oder ODER auf UND), wird eine Warnmeldung erzeugt. Dadurch können Verschachtelungen beliebiger Tiefe auf korrekte Kombinationen von Verzweigungen und Vereinigungen geprüft werden. Mit einer Petrinetz-Vorlage ist die Prüfung nur für eine festgelegte Verschachtelungstiefe möglich, da Petrinetze ohne entsprechende Erweiterungen die Reihenfolge der auftretenden Verzweigungen nicht beliebig häufig „speichern“ können (vgl. Petrinetze und Zählen in Murata (1989)).

REVEAL zeigt für nicht korrekte Kombinationen sowohl das öffnende wie auch das schließende Element an, also die auslösenden Plätze oder Transitionen. Anhand der gezeigten ID (siehe unten in Abbildung 4.3) kann das jeweilige Korrektheits-Problem vom Workflow-Designer sofort identifiziert werden.

4.1.2 Datenfluss

Die vier aufgestellten Datenfluss-Regeln D.M, D.O, D.S und D.V sind in der Abbildung 4.4 in den Workflow des Fallbeispiels eingetragen.

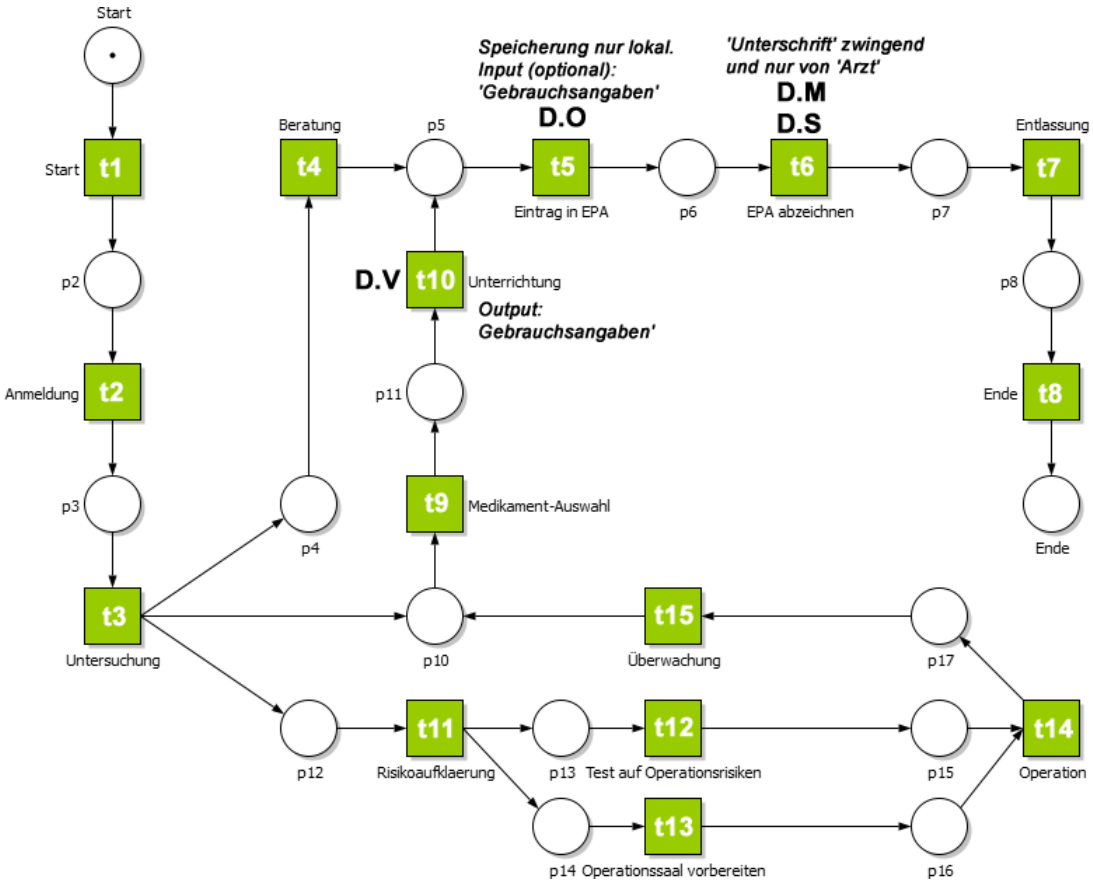


Abbildung 4.4: Fallbeispiel-Workflow mit Datenfluss-Vorgaben

Modalität

In Abbildung 4.5 ist das Ergebnis der Datenflussprüfung auf Modalität zu sehen. Die aufgestellte Regel lautet, dass das Datenelement *Unterschrift* in jedem Workflow-Pfad enthalten sein muss. Wie die Tokens in den mit *OK* markierten Plätzen der Regel-Petrinetze zeigen, erfüllt jeder Pfad des Workflows diese Vorgabe. Zum Vergleich zeigt Abbildung 4.6 das Prüfungsergebnis, nachdem *Unterschrift* aus dem Workflow entfernt wurde und somit keiner der Pfade die entsprechende Regel einhält.

Ort

Zusätzlich zur Modalität eines Datenelements kann auch eine Ortsangabe erfolgen. Die elektronische Patientenakte (EPA) darf im Fallbeispiel nur lokal abgelegt werden. Wie die Abbildung 4.7 zeigt, erfasst REVEAL die nötigen Modell-Elemente und bestätigt die Regeleinhaltung. Das Listing 12 enthält den Ausschnitt des Workflow-Petrinetzes, in dem die Transition *t5: Eintrag in EPA* definiert wird. In der viert- und drittletzten Zeile wird festgehalten, dass die Transition bzw. Aktivität die EPA verwendet und

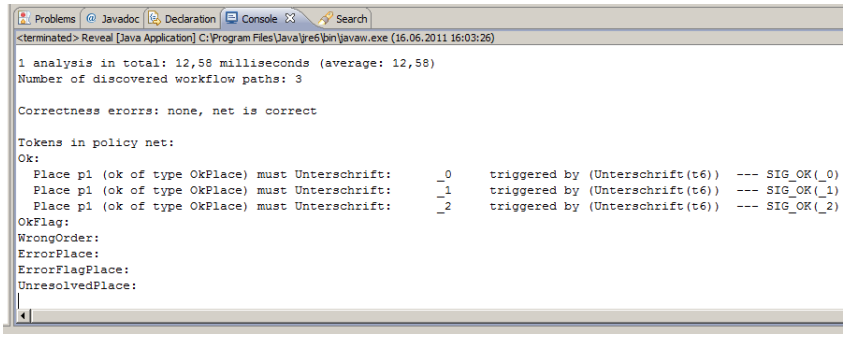


Abbildung 4.5: Datenflussprüfung: Modalitätsregel eingehalten

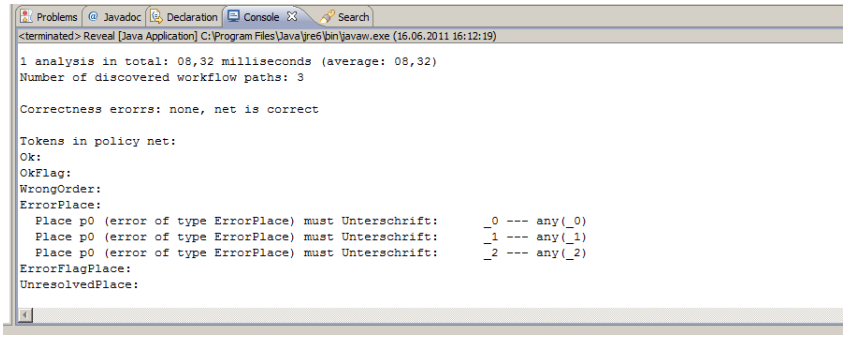


Abbildung 4.6: Datenflussprüfung: Modalitätsregel verletzt

lokal ablegt. Anstelle der Angabe *lokal* könnte hier beispielsweise auch der Name eines Servers eingetragen werden.

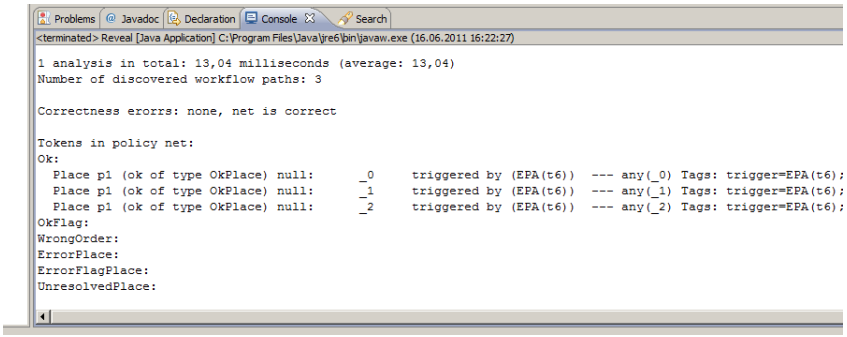


Abbildung 4.7: Datenflussprüfung: Ortsregel eingehalten

Listing 12: Transition *Eintrag in EPA* als PNML

```
1  <transition id="t5">
2    <name>
3      <text>Eintrag in EPA</text>
4    </name>
5    <graphics>
6      <position x="370" y="120"/>
7      <dimension x="40" y="40"/>
8    </graphics>
9    <toolspecific tool="plc2pnml" version="alpha">
10     <data>EPA</data>
11     <location>lokal</location>
12   </toolspecific>
13 </transition>
```

In den Fällen, in denen das zu prüfende Element modelliert wurde, zeigt Reveal die (Nicht-) Einhaltung an. Fehlt das Element, also beispielsweise die Angabe *location*, wird eine Warnung ausgegeben, da die Einhaltung dann nicht automatisiert bewertbar ist. In einem solchen Fall kann allerdings eine Standardbewertung eingestellt werden, so dass bei Abwesenheit eines Elements die jeweilige Regel immer als (nicht) eingehalten bewertet wird. Der Prototyp gibt in diesem Fall eine Meldung wie die folgende aus (diese Meldung wurde ausgegeben, nachdem das XML-Element <location> aus der PNML-Datei des Workflows gelöscht wurde).

```
SIG_ERR(_0) Tags: trigger=EPA(t5);Clash encountered=EPA (t5);
CLASH=Policy tag -location- not found in the triggering workflow element.
```

Ressource

Entsprechend der Ortsprüfung läuft auch die Ressourcenprüfung ab. Der Screenshot in Abbildung 4.7 gibt die Meldung wieder, die Reveal nach der Prüfung des Workflows erzeugt, wenn die *Unterschrift* nicht wie verlangt vom Arzt, sondern von einer Krankenschwester erzeugt wird. Keiner der drei Pfade durch den Workflow hält diese Regel ein, und anhand der Angaben zur auslösenden Transition *t6* sowie zum auslösenden Datenelement *unterschrift* und zum Workflow-Pfad *_0, _1* oder *_2* kann das Problem vom Workflow-Designer bzw. -Prüfer direkt nachvollzogen werden.

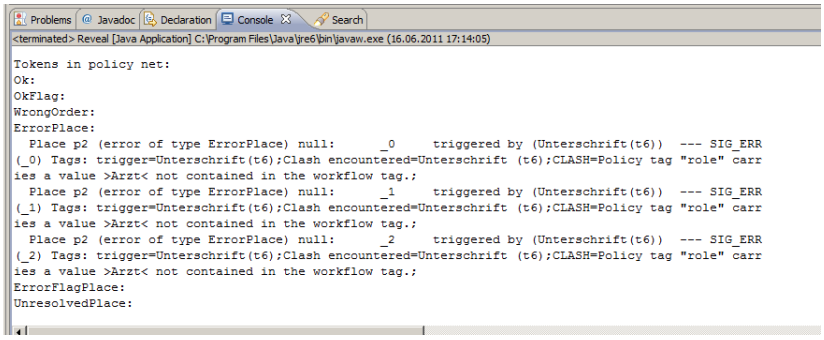


Abbildung 4.8: Datenflussprüfung: Ressourcenregel verletzt

Verwendung

Im Fallbeispiel gilt die Regel, dass die bei der Aktivität *Unterrichtung* entstehenden *Gebrauchsangaben* von der Aktivität *Eintrag in EPA* verwendet werden müssen. Allgemein muss bei einer solchen Verwendungsregel zwischen drei Fällen unterschieden werden. Im ersten Fall wird das Datenelement gar nicht erzeugt, muss bzw. kann also auch nicht verwendet werden. Im zweiten wird das Datenelement erzeugt und verwendet. Im dritten und einzigen nicht-regelkonformen Fall wird das Datenelement zwar erzeugt, aber nicht verwendet. Entsprechend fällt die Prüfung des Workflows im Fallbeispiel aus, deren Ergebnis in Abbildung 4.9 festgehalten ist. Auf zwei Pfaden tritt zunächst die Aktivität *Unterrichtung* (Transition t10) auf und erzeugt das Datenelement *Gebrauchsangaben*, bevor es von der Aktivität *Eintrag in EPA* (Transition t5) verwendet wird. Auf einem Pfad wird das Datenelement gar nicht erst erzeugt, so dass auch hier die Regel eingehalten wird.

4.1.3 Kontrollfluss

Die Kontrollflussprüfung wurde anhand des in Abbildung 4.10 gezeigten Workflows evaluiert. Die aufgestellten Regeln A.M, A.O, A.R, A.S und A.Z sind neben den jeweiligen Aktivitäten eingetragen.

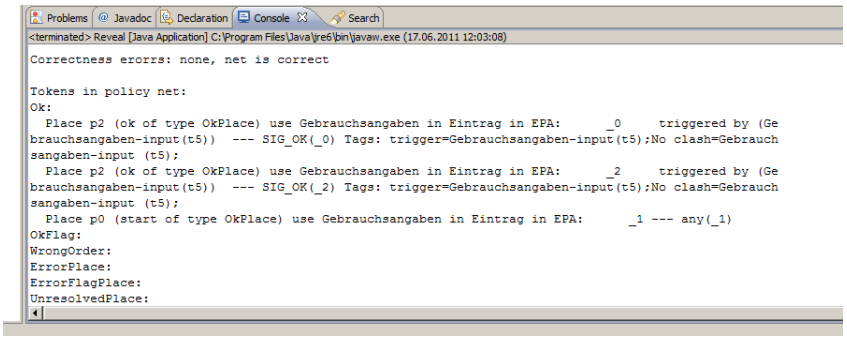


Abbildung 4.9: Datenflussprüfung: Verwendungsregel eingehalten

Modalität

In Abbildung 4.11 ist das Ergebnis der Kontrollflussprüfung auf Modalität zu sehen. Die aufgestellte Regel lautet, dass die Aktivität *Anmeldung* in jedem Workflow-Pfad enthalten sein muss. Der Workflow hält die Regel ein, wie das Token im Platz *OK* des Regel-Petrinetzes zeigt. Drei weitere Tokens weisen jedoch scheinbar auf eine Schwachstelle hin. Tatsächlich handelt es sich dabei aber nicht um eine Schwachstelle, sondern nur um eine ergänzende Warnung (daher der Hinweis „ignore“ in der Meldung am unteren Rand der Abbildung). Reveal analysiert alle Pfade des Workflows und bewertet dabei zunächst nicht nur die vollständigen Pfade vom Beginn bis zum Ende des Workflows, sondern auch neue Pfade die nach einer Verzweigung beginnen. Die drei Warnmeldungen rühren daher, dass auf keinem Pfad von der ODER-Verzweigung *t3* bis zum Ende des Workflows die Aktivität *Anmeldung* bzw. Transition *t1* gefunden wird. Beim automatisierten Zusammenführen der Prüfungsergebnisse stellt Reveal fest, dass die drei Pfade auf den Anfangspfad folgen, welcher bereits die Regel erfüllt da er *Anmeldung* bzw. *t1* enthält. Die drei Meldungen dienen daher nur der Vollständigkeit und können ignoriert bzw. auf Wunsch mittels eines Schalters im Prototypen ausgeblendet werden.

Entfernt man die Aktivität *Anmeldung* aus dem Workflow, wie in Abbildung 4.12 geschehen, wird die ursprüngliche Regel auf keinem Pfad mehr eingehalten. Negiert man nun die Regel, so dass die Aktivität *Anmeldung* nicht mehr enthalten sein darf, halten alle Pfade des Workflows diese negierte Regel ein, wie das Prüfungsergebnis in Abbildung 4.13 darstellt.

Ort

Die Aktivität *Operation* (Transition *t12*) darf nur im Operationssaal stattfinden. Im Gegensatz zur Vorlage für Modalitätsangaben muss hierbei nicht nur geprüft werden, ob eine bestimmte Transition im Petrinetz auf allen Pfaden steht, sondern auch, ob die modellierte Ortsangabe den gewünschten Wert trägt. Abbildung 4.14 enthält die Ausgabe des Analyse-Prototypen für den Fall, dass die Ortsangabe nicht wie gewünscht

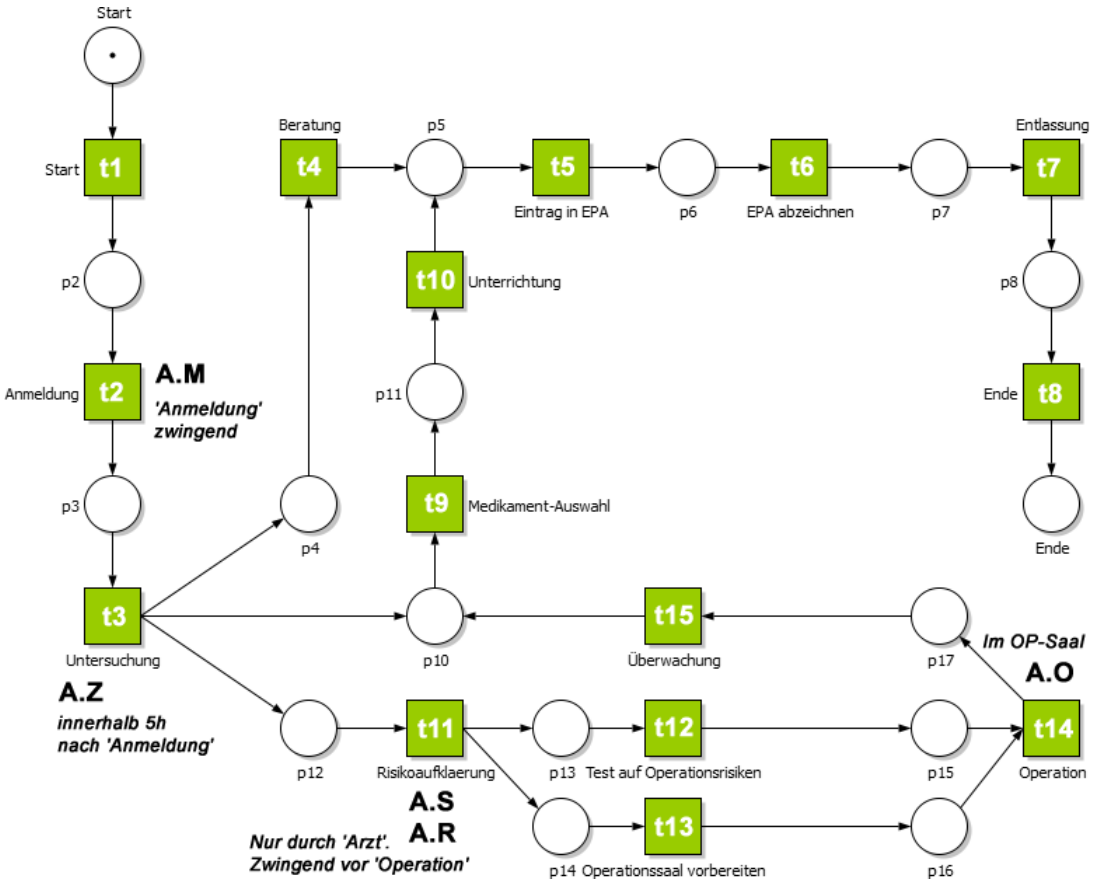


Abbildung 4.10: Fallbeispiel-Workflow mit Datenfluss-Vorgaben

Operationssaal lautet.

Um bei einer Negativ-Auswahl von Orten nicht jeden einzelnen Ort auflisten zu müssen, erlaubt der Prototyp die Angabe eines führenden Ausrufezeichens. Die Angabe *!Operationssaal* wird dann als „alle Orte ausser dem Operationssaal“ interpretiert.

Reihenfolge

Im Fallbeispiel muss die *Risikoaufklärung* vor der *Operation* erfolgen. Hierbei ist zu beachten, dass diese Vorgabe für die Reihenfolge keine Modalitätsvorgabe beinhaltet, das heißt, weder die Risikoaufklärung noch die Operation **müssen** im Workflow enthalten. Wenn aber die Operation enthalten ist, muss zuvor die Risikoaufklärung erfolgen. Wenn nur die Risikoaufklärung enthalten ist, ist dies regelkonform. Das Prüfungsergebnis in Abbildung 4.15 gilt für den regelkonformen Workflow des Fallbeispiels. Auf zwei von drei Pfaden ist weder die Aktivität bzw. Transition *Risikoaufklärung* noch *Operation* vorhanden, was die beiden Tokens im Startplatz *p0* des Regel-Petrinetzes anzeigen (siehe die unteren beiden der drei Tokens in der Abbildung). Auf dem dritten Pfad stehen beide Transitionen in der richtigen Reihenfolge (siehe oberes der drei

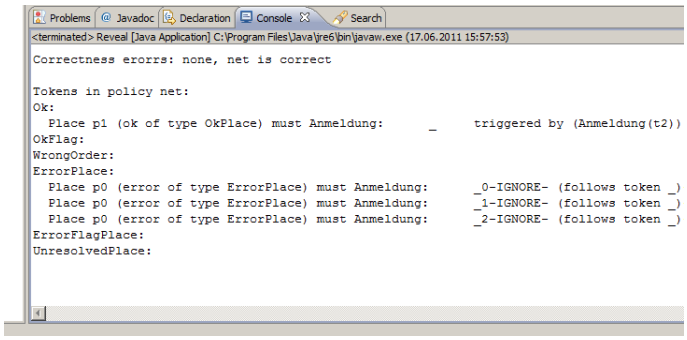


Abbildung 4.11: Kontrollflussprüfung: Modalitätsregel eingehalten

Tokens in der Abbildung).

Ressource

Die *Risikoaufklärung* im Behandlungs-Workflow darf nur durch einen *Arzt* erfolgen. Diese Rollenzuweisung muss entsprechend im Workflow modelliert sein. Falls im Workflow für die Aktivität keine spezielle Rolle modelliert bzw. annotiert ist, gibt REVEAL eine Fehlermeldung entsprechend der in Abbildung 4.16 aus. Neben den Ergebnis-Plätzen *OK* und *ERROR* gibt es dazu noch den Typ *UNRESOLVED* bzw. Tokens mit der Kennung *SIG_ UNRES* (wie am unteren Rand der Abbildung zu sehen). Die Einhaltung der Ressourcenregel ist in diesem Fall unklar und kann erst ausgewertet werden, wenn im Workflow-Modell die Aktivität mit einer Rollenzuweisung versehen wurde.

Sobald die Rollenzuweisung im Workflow gefunden wird, kann die angegebene Rolle mit der geforderten verglichen und die Regeleinhaltung somit ausgewertet werden. In Abbildung 4.17 ist die Auswertung für den Fall zu sehen, dass die geforderte Rolle *Arzt* im Workflow für die Aktivität *Risikoaufklärung* modelliert wurde. Die XML-basierte PNML-Codierung der entsprechenden Transition *Risikoaufklärung* (*t11*) ist im Listing 13 angegeben.

Listing 13: Transition *Risikoaufklärung* als PNML

```
1  <transition id="t11">
2    <name>
3      <text>Risikoaufklaerung</text>
4    </name>
5    <graphics>
6      <position x="280" y="520"/>
7      <dimension x="40" y="40"/>
8    </graphics>
9    <toolspecific tool="plc2pnml" version="alpha">
10     <branchingType split="and"/>
11     <role>Arzt</role>
```

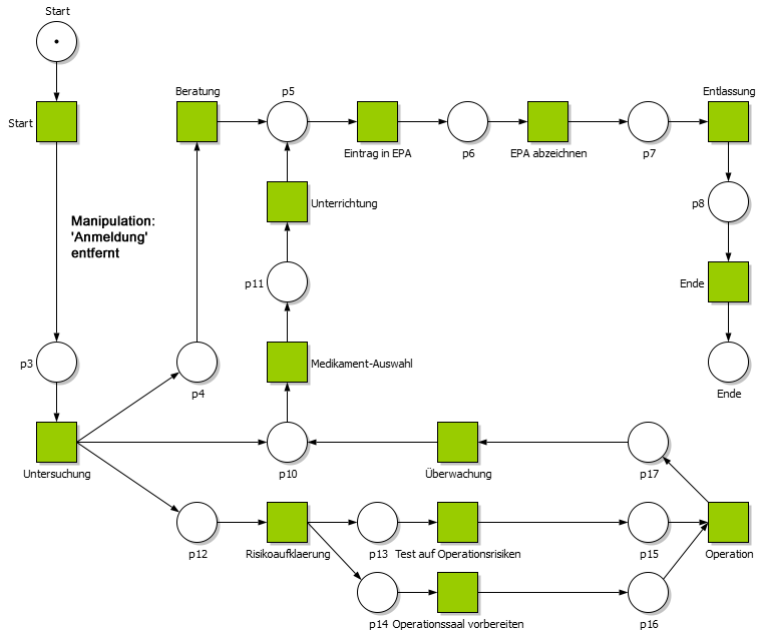


Abbildung 4.12: Fallbeispiel-Workflow ohne Aktivität *Anmeldung*

```
12      </toolspecific>
13  </transition>
```

Zeitlimit

Für die Regel, dass die *Untersuchung* innerhalb von fünf Stunden nach der *Anmeldung* durchgeführt werden muss, ist das Prüfungsergebnis in Abbildung 4.18 wiedergegeben. Gleichzeitig wurde in diesem Testfall auch die umgekehrte Reihenfolge, nämlich dass die *Anmeldung* fünf Stunden nach der *Untersuchung* stattfinden muss abgefragt. Dementsprechend liegen abschließend zwei Ergebnis-Tokens in den Ergebnis-Plätzen (siehe in der Abbildung unten). Ein Token zeigt an, dass die ursprüngliche Regel *Untersuchung* nach *Anmeldung* eingehalten wird. Das andere Token zeigt an, dass die umgekehrte Reihenfolge nicht eingehalten wird und die Transitionen *Anmeldung* und *Untersuchung* dafür in der falschen Reihenfolge stehen (signalisiert durch ein SIG_WO-Token im *WrongOrderPlace*).

Bei der Prüfung auf die Einhaltung von Zeitlimits ist zu beachten, dass die statische Analyse bzw. die Analyse zur Entwurfszeit nur anzeigen kann, ob die beiden von der Regel erfassten Aktivitäten in der richtigen Reihenfolge modelliert wurden. Die tatsächliche Ausführungszeit kann erst zur Laufzeit ermittelt und beispielsweise von einem Laufzeit-Monitor überwacht werden.

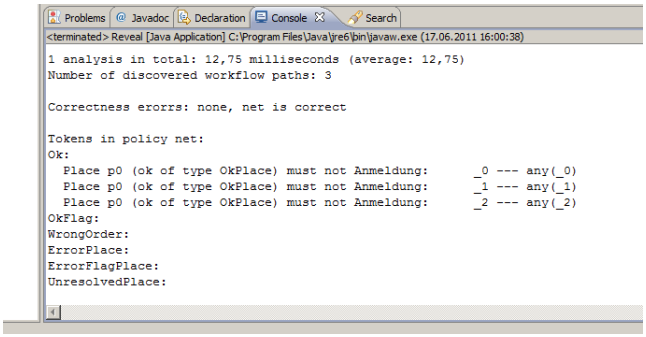


Abbildung 4.13: Kontrollflussprüfung: negierte Modalitätsregel eingehalten

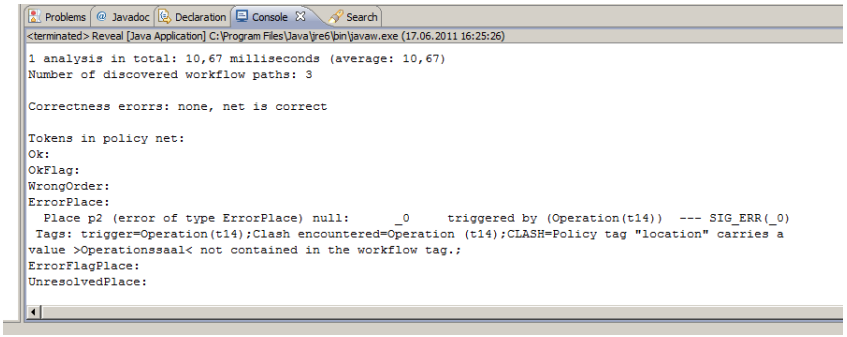


Abbildung 4.14: Kontrollflussprüfung: Ortsregel verletzt

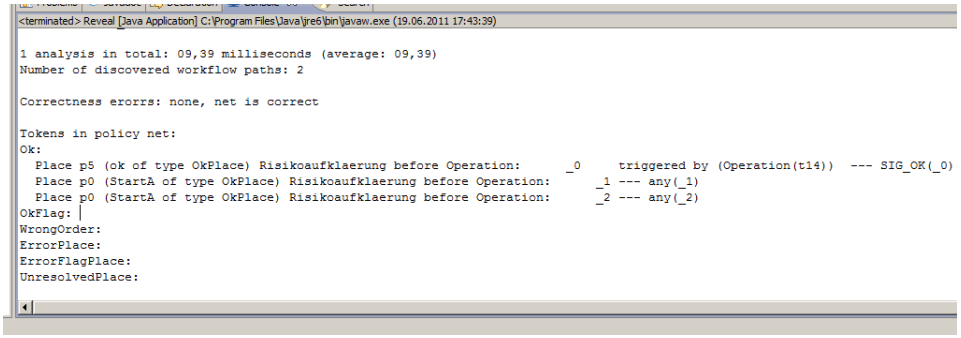


Abbildung 4.15: Kontrollflussprüfung: Reihenfolgeregel eingehalten

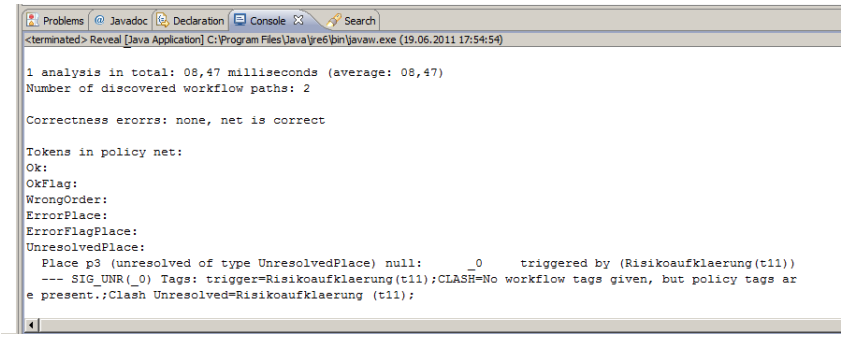


Abbildung 4.16: Kontrollflussprüfung: Einhaltung der Ressourcenregel unklar

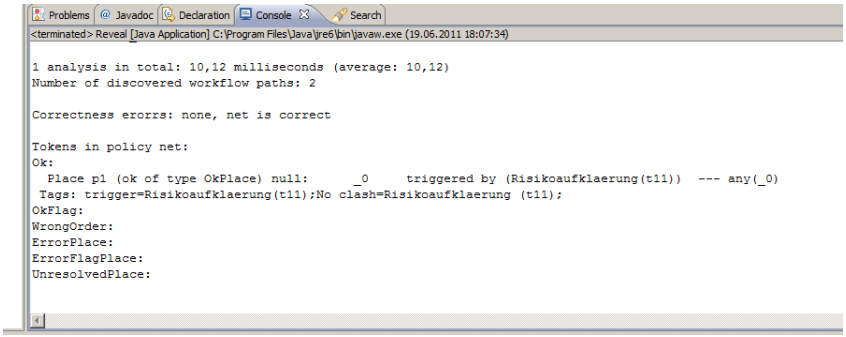


Abbildung 4.17: Kontrollflussprüfung: Ressourcenregel eingehalten

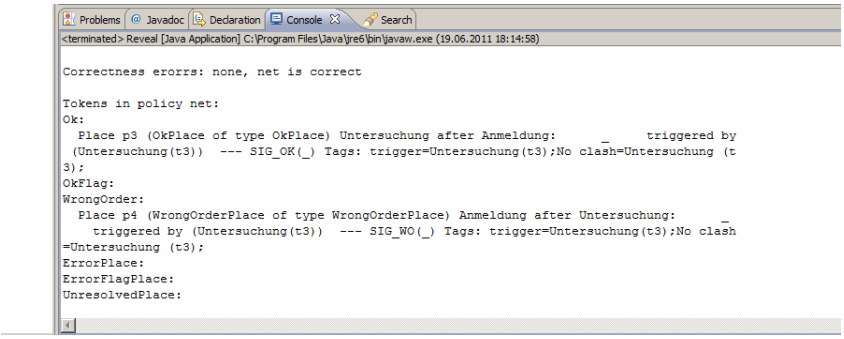


Abbildung 4.18: Kontrollflussprüfung: Zeitlimitregel eingehalten

4.1.4 Widerspruchsfreiheit zwischen Daten- und Kontrollfluss

Die Kombination aus Daten- und Kontrollfluss muss selbst dann noch geprüft werden, wenn der Datenfluss für sich genommen keine Schwachstellen enthält und der Kontrollfluss für sich genommen als korrekt befunden wurde. Zur Demonstration wurde aus dem Workflow des Fallbeispiels die Aktivität *Eintrag in EPA* entfernt. Der resultierende Workflow ist in Abbildung 4.19 dargestellt und ist aus Kontrollfluss-Sicht korrekt. Der Datenfluss hält auf einem von drei Pfaden die Regel ein, dass die *Gebrauchsangaben* aus der *Unterrichtung* durch *Eintrag in EPA* verwendet werden müssen, und zwar auf dem Pfad über die Aktivität *Beratung*, weil dort die *Gebrauchsangaben* gar nicht erzeugt werden. Auf allen drei Pfaden ist aber ein Widerspruch in der Datenfluss-Kontrollfluss-Kombination festzustellen, und zwar dass die Aktivität *EPA abzeichnen* die elektronische Patientenakte benötigt, diese aber wegen der fehlenden Aktivität *Eintrag in EPA* nicht zur Verfügung steht. Dementsprechend wird bei dem in Abbildung 4.20 nicht nur auf zwei von drei Pfaden die Verwendung der *Gebrauchsangaben* moniert, sondern auch auf allen drei Pfaden das fehlende Datenelement *EPA* hervorgehoben (siehe Meldung „missing data elements“).

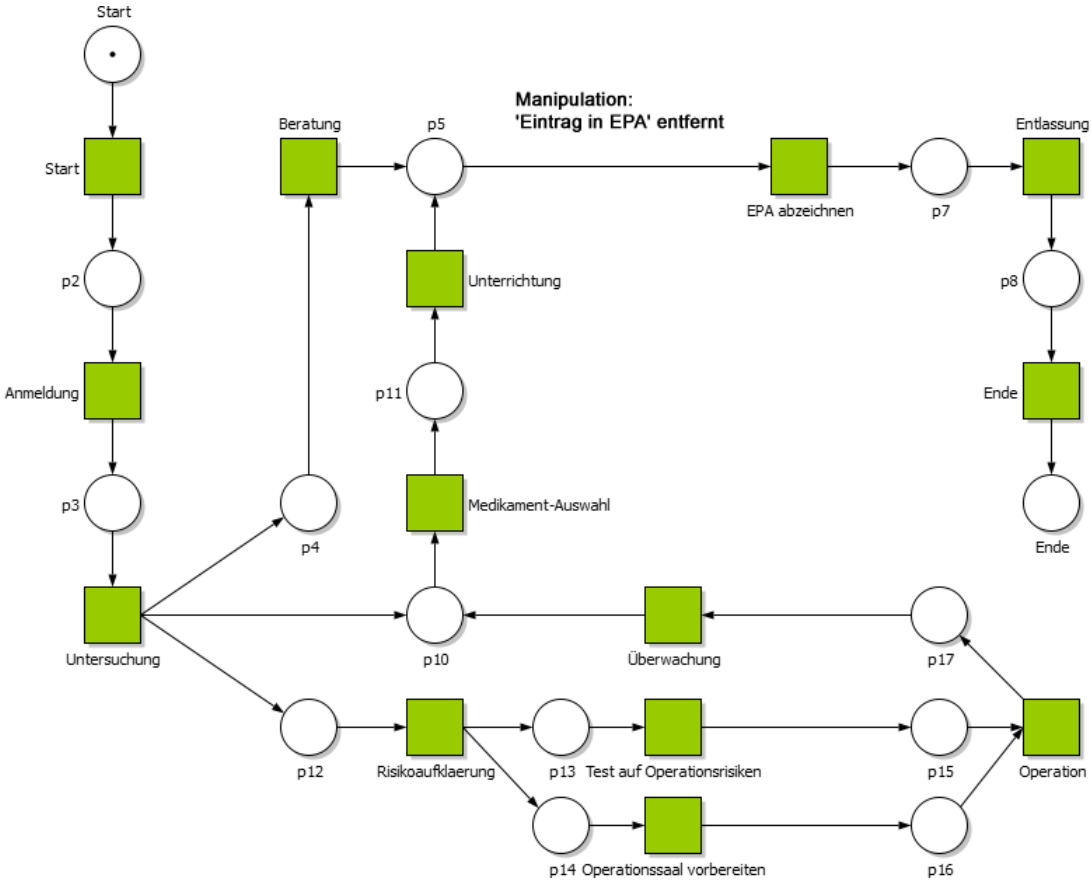


Abbildung 4.19: Behandlungs-Workflow ohne Aktivität *Eintrag in EPA*

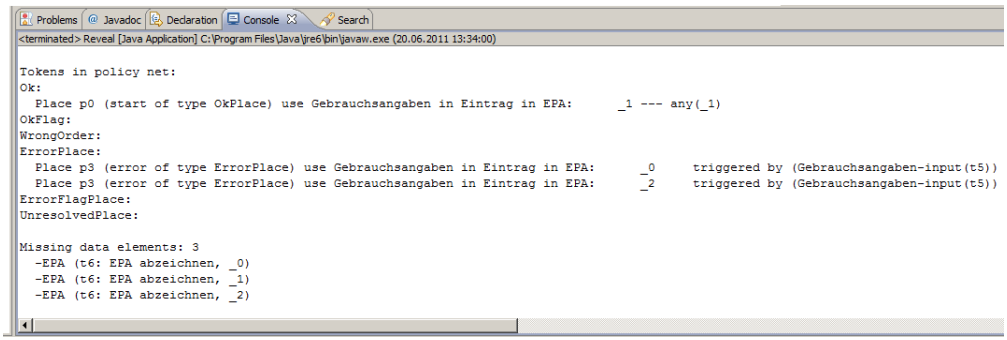


Abbildung 4.20: Widerspruchsfreiheit: Fehlende Datenelemente bzw. -verwendung

Für die Prüfung des vollständigen Workflows, das heißt inklusive der ursprünglichen Aktivität *Eintrag in EPA*, ist das Ergebnis in Abbildung 4.21 zu sehen. Sowohl die Einhaltung der Verwendungsregel für die *Gebrauchsangaben* wie auch die Widerspruchsfreiheit des Daten- und Kontrollflusses werden hier bestätigt.

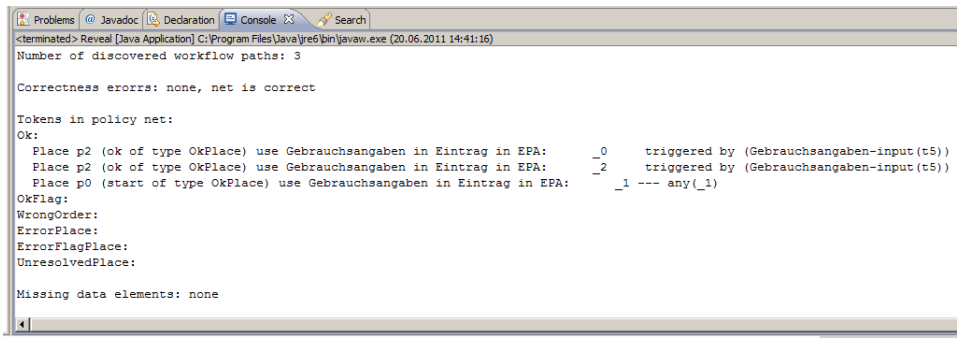


Abbildung 4.21: Widerspruchsfreiheit: keine Schwachstellen gefunden

4.1.5 Einhaltung von HIPAA-Regeln

Wie unter anderem von Breaux (2009) gezeigt, lassen sich alle HIPAA-Regeln anhand der in Abschnitt 2.1.3 vorgestellten Regelelemente ausdrücken. Auszugsweise wird nachfolgend gezeigt, wie genau die Übertragung von HIPAA-Regeln in die Regelelemente aussieht. Die dazu ausgewählten HIPAA-Regeln lauten wie folgt.

- §164.502b „minimum necessary“ (need to know)
- §164.504e2iiI „delete PHI after use“
- §164.506a1 „obtain consent“
- §164.506a3Cii „document if consent not obtained“

- §164.524b2i „upon request, react within 30 days“
- §164.528 „provide PHI listing of last 6 years“

Laut der „Minimum necessary“-Regel dürfen nur die minimal für eine Aktivität benötigten Daten ausgetauscht werden. Für die Prüfung mit Reveal wird dazu die Modalität und ggf. die Verwendung von Datenelementen angegeben. Im Zusammenhang mit dem „default deny“-Grundsatz von HIPAA kann zusätzlich über die Angabe der Ressource (Rolle/Dienst) eine Zugriffsberechtigung geprüft werden.

Für dritte Parteien gilt bei HIPAA „delete PHI after use“, Gesundheitsdaten müssen also nach der Verarbeitung gelöscht werden. Bei der Prüfung wird dies durch die Modalität und Reihenfolge von Aktivitäten und die Verwendungsregeln für Daten modelliert.

Grundsätzlich muss laut HIPAA vor der Verwendung von Daten die Einwilligung der Betroffenen eingeholt werden („obtain consent“). Auch das kann durch die Modalität und Reihenfolge von Aktivitäten ausgedrückt werden.

Für die Regel „document if consent not obtained“, also die Verpflichtung zu dokumentieren, warum im Ausnahmefall eine Einwilligung nicht vor der Verwendung eingeholt wurde, dient die Kombination aus Modalität und Reihenfolge bei Aktivitäten und der Modalität von Daten („Dokumentation muss vorliegen“).

Auf Anfragen von Patienten nach den über sie gespeicherten Gesundheitsdaten muss HIPAA §164.524b2i entsprechend innerhalb von 30 Tagen reagiert werden („upon request, react within 30 days“). Mit einem Zeitlimit zwischen Eingang der Anfrage und Reaktion auf die Anfrage kann die Einhaltung dieser Regel von Reveal im Workflow-Modell geprüft werden (die tatsächliche Einhaltung zur Laufzeit kann wiederum nicht im Modell, sondern erst von einem Monitor geprüft werden).

Bei einer Gesundheitsdatenauskunft für Patienten müssen die letzten sechs Jahre berücksichtigt werden („provide PHI listing of last 6 years“). Entsprechend dürfen die Daten nicht vor Ablauf dieser Frist gelöscht werden, was für die Prüfung in Form von Zeitlimits formuliert wird.

Die Tabelle 4.1 zeigt zusammenfassend, anhand welcher Regelelemente die HIPAA-Regeln erfasst werden. Dabei werden die in Tabelle 2.6 eingeführten Kürzel verwendet.

Tabelle 4.1: Zuordnung von HIPAA-Regeln zu Regelelementen

§164.502b „minimum necessary“	D.M, D.V, D.S
§164.504e2iiI „delete PHI after use“	A.M, A.R, D.V
§164.506a1 „obtain consent“	A.M, A.R
§164.506a3Cii „document if consent not obtained“	A.M, A.R, D.M
§164.524b2i „upon request, react within 30 days“	A.Z
§164.528 „provide PHI listing of last 6 years“	D.Z

Im Gegensatz zu anderen Regularien, wie etwa dem Bundesdatenschutzgesetz, enthalten HIPAA-Regeln keine expliziten Angaben zu den Orten der Ausführung von

Aktivitäten. Dementsprechend finden in der Tabelle 4.1 Ortsangaben (O) keine Verwendung. Wie in den Abschnitten 4.1.2 und 4.1.3 gezeigt, beinhaltet die Prüfung mit Reveal sämtliche in der Anforderungstabelle 2.6 aufgeführten Elemente, also auch Ortsangaben. Die HIPAA-Regeln beanspruchen demnach nur eine Untermenge der von Reveal erfassten Elemente.

4.1.6 Geschwindigkeit der Prüfung

Für die Einbindung in Workflow-Editoren, also Software zum Entwurf von Workflows, sollte die Prüfung auf Regeleinhaltung nur wenige Millisekunden benötigen, um den Entwurfsvorgang möglichst wenig aufzuhalten. Davon abgesehen stellt natürlich auch die Prüfung innerhalb weniger Sekunden eine starke Reduktion der Prüfungszeit dar, wenn sie mit der minuten-, stunden- oder womöglich tagelangen manuellen Prüfung verglichen wird, die bei entsprechend großen und komplexen Workflows und Regelsätzen nötig sein könnte. Die Tabelle 4.2 enthält die Angaben über die vom Reveal-Protoyp für die Prüfung von Workflows unterschiedlicher Größe und Komplexität benötigte Zeit.

Tabelle 4.2: Analysedauer verschiedener Testfälle (in Millisekunden)

Transitionen	10	25	50	75	100	150	200	300	400	500
6% Splits	1	4	22	14	101	24	585	983	650	224732
13% Splits	1	4	10	3213	95	131	434	1072	4158	31830
25% Splits	3	8	13	193	368	2867	2988	-	-	-
50% Splits	6	6	16661	10840	1294	15423	-	-	-	-

Bei den Testfällen ohne Angabe einer Analysedauer war die Prüfung nach fünf Minuten noch nicht abgeschlossen, was als *time out* gewertet wurde. Bei allen Testläufen wurde die Analysedauer über zehn Tests aufsummiert und gemittelt. Die Schwankungen sind einerseits auf die Java-Laufzeitumgebung zurückzuführen. Andererseits wurden die Workflows zufällig erzeugt und enthielten dementsprechend die Verzweigungen und Schleifen an unterschiedlichen Stellen. Neben dem in Tabelle 4.2 pro Zeile angegebenen prozentualen Anteil von Verzweigungen spielt vor allem die Anzahl der im Workflow enthaltenen, potentiell verschachtelten Schleifen eine wichtige Rolle. Dieser Anteil lag bei den obigen Testfällen bei sechs Prozent. Besonders Schleifen, die kurz vor Ende des Workflows noch einmal fast zum Anfang des Workflows zurückführen und damit weitere Schleifen innerhalb des Workflows umschließen, führen wegen der vielen zusätzlich zu untersuchenden Pfade zu einer stark verlängerten Analysezeit. Die Ergebnisse in der dritten und vierten Zeile resultierten aus der Untersuchung von Workflows mit 25 bzw. 50 Prozent Verzweigungen, das heißt, jede vierte bzw. jede zweite Aktivität des Workflows verzweigte in mindestens zwei Ausführungspfade. In der Workflow-Sammlung von Ben-Eliahu (2008) beispielsweise treten Verzweigungen deutlich seltener auf.

Bei den oben untersuchten Testfällen wurden jeweils 15 Regeln eingesetzt. Die Analysedauer steigt linear mit der Anzahl der formulierten Regeln, wie Tabelle 4.3 zeigt; es wurden jeweils 50 Transitionen untersucht, wobei sechs Prozent Verzweigungen und sechs Prozent Schleifen enthalten waren. Durchschnittlich stand die Analysedauer in einem Verhältnis von etwa Eins zu Eins zur Anzahl Regeln. Die Schwankungen in der gemessenen Dauer sind auf die Java-Laufzeitumgebung zurückzuführen. Trotz zwischenzeitlicher Speicherbereinigung wurden für identische Testfälle bei mehrfacher Ausführung immer leicht unterschiedliche Ausführungszeiten gemessen, selbst wenn über 100 oder mehr Testfälle gemittelt wurde.

Tabelle 4.3: Analysedauer bei steigender Anzahl Regeln (in Millisekunden)

Regeln	5	10	15	20	25	50	100
Dauer	6	9	12	14	16	83	132

Die in Workflow-Sammlungen wie von (Ben-Eliahu, 2008) zu beobachtende typische Workflow-Größe von weniger als 50 Aktivitäten lässt sich diesen Testergebnissen zufolge innerhalb weniger Millisekunden prüfen. Mit einer wachsenden Anzahl von Schleifen, allgemein aber bereits mit einer wachsenden Anzahl Verzweigungen wächst der Bedarf an Speicher und Analysezeit exponentiell. Bis zu einer Größe von etwa 400 Aktivitäten bzw. etwa 50 Verzweigungen und 25 (teilweise verschachtelten) Schleifen analysiert der Prototyp Workflows in einer Geschwindigkeit, die für die Analyse „auf Knopfdruck“ in einem Workflow-Editor geeignet ist. Während sicherlich noch einige Geschwindigkeitsoptimierungen möglich sind, bleibt die Schranke durch die bereits diskutierte Zustandsraumexplosion bestehen. Workflows mit mehr als 500 Aktivitäten bzw. deutlich mehr als 50 Verzweigungen und 25 (teilweise verschachtelten) Schleifen können mit dem hier verfolgten Ansatz nicht in weniger als einer Sekunde analysiert werden.

Der Speicherbedarf bei der Analyse stieg von unter 1 MB bei kleinen Workflows auf bis zu 11 MB bei den großen Workflows.

4.2 Bewertung des Reveal-Verfahrens

Aus den Ergebnissen der Prüfung des Fallbeispiel-Workflows lässt sich das neue Verfahren wie in Tabelle 4.4 gezeigt einordnen. Für die Korrektheitsprüfung sowie entweder die Datenfluss- oder die Kontrollflussprüfung existierten bereits Verfahren, die aber weder *Orte* noch *Ressourcen* erfassten. Damit erreichte keines der bisherigen Verfahren den für Compliance-Prüfungen nötigen Umfang der Informationsflussanalyse (siehe Abschnitt 2.5). Reveal schließt diese Lücke und bietet darüber hinaus die Möglichkeit, den Daten- und Kontrollfluss *gleichzeitig* zu analysieren.

Außerdem stellt Reveal ein Verfahren dar, welches mit Petrinetzen nur einen einzelnen, für Workflow-Designer wegen der Ähnlichkeit zu den üblichen graphischen Modellen gut zugänglichen Formalismus nutzt. Seinem vorlagenbasierten Ansatz zufolge

Tabelle 4.4: Einordnung des Reveal-Verfahrens

Eigenschaft / **Verfahren**	**K**	**DF**				**KF**					**W**
		M	O	S	V	M	O	R	S	Z	
Reveal	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

K=Korrektheit,DF=Daten-,KF=Kontrollfluss,W=Widerspruchsfreiheit,M=Modalität,O=Orte,R=Reihenfolge,S=Ressourcen, V=Verwendung,Z=Zeitlimits

geben die Ergebnisse des Tests am Prototypen die Hoffnung, dass Reveal leicht für eventuelle, zukünftig zusätzlich identifizierte Analyseziele eingesetzt werden kann. Dazu ist im Idealfall nicht mehr nötig, als die zusätzlich zu prüfenden Regeln in eine Vorlage bzw. ein Petrinetz zu transformieren. Anhand solcher zusätzlichen Vorlagen kann Reveal mit dem bestehenden Prüfalgorithmus einen Workflow auf die Einhaltung *neuer* Regeln prüfen. Die Regeln aus den zum Zeitpunkt der Untersuchung aktuellen Compliance-Regularien, wie sie im zweiten Kapitel diskutiert wurden, werden von den in dieser Arbeit vorgestellten Vorlagen bereits erfasst.

5 Zusammenfassung und Ausblick

Den Fortschritten bei der Automatisierung der Änderung und Ausführung von Geschäftsprozessen steht die bisher weitgehend manuelle, aber gesetzlich erzwungene Prüfung bremsend gegenüber. Die vorliegende Arbeit ist als Beitrag zur automatisierten Compliance-Prüfung von Geschäftsprozessen zu sehen, bei dem im Gegensatz zu den bisher existierenden Ansätzen der für Compliance-Prüfungen nötige Umfang an erfassbaren Regeln erreicht wird.

Workflows, der Definition von Hansen und Neumann (2005) folgend als vollständig oder teilweise automatisierte Geschäftsprozesse gesehen, können nun nicht nur wie bisher schnell geändert und ausgeführt werden, sondern mit dem vorgestellten Ansatz auch schnell geprüft werden. Setzt man die Grenze für die „schnelle“ Prüfung auf fünf Minuten, liegt sie beim entwickelten Prototypen bei einer Workflow-Größe von etwa 500 Aktivitäten. Workflows mit bis zu etwa 100 Aktivitäten können hingegen innerhalb weniger Millisekunden analysiert werden. Die typische Workflow-Größe in Sammlungen wie der von Ben-Eliahu (2008) beträgt weniger als 25 Aktivitäten.

Der vorgestellte Ansatz namens Reveal bietet die Möglichkeit, Workflows automatisiert auf Regeleinhaltung zu prüfen und dabei nicht nur alle Compliance-Regelelemente zu erfassen, sondern auch gleichzeitig den Datenfluss *und* den Kontrollfluss zu berücksichtigen. Workflows und Regeln werden dazu in Petrinetze transformiert, so dass die Einhaltung der Compliance-Regeln durch eine vorlagenbasierte Analyse geprüft werden kann. Entsprechen Workflow-Elemente nicht den Regeln bzw. ermöglichen sie einen Regelverstoß, wird die Position der jeweiligen Schwachstellen im Workflow lokalisiert.

Die Prüfung sowohl des Daten- als auch des Kontrollflusses ist nötig, um die im zweiten Kapitel klassifizierten Compliance-Regeln abdecken zu können. Bisherige Ansätze legen jedoch, wie im selben Kapitel diskutiert, einerseits den Fokus auf einen der beiden Informationsflüsse, andererseits erfassen sie nur einen Teil der Compliance-Regeln. Dass die gleichzeitige Prüfung all dieser Regeln möglich ist, zeigt das im dritten Kapitel vorgestellte Prüfverfahren, dessen Eignung zur automatisierten Prüfung im vierten Kapitel anhand eines Fallbeispiels gezeigt wurde.

Reveal beinhaltet Regelvorlagen für alle im zweiten Kapitel identifizierten, zur Compliance-Prüfung nötigen Regelelemente, nämlich Modalität, Ort, Reihenfolge, Ressource (Rolle/Dienst), Verwendung, Zeitlimit. Neben dem für Zugriffskontrolle notwendigen Element der *Ressource* wird mit dem *Ort* ein Element erfasst, welches von bisherigen Analyseansätzen ignoriert wurde, aber im Rahmen des „Cloud Computing“ bereits stark an Bedeutung gewonnen hat und mit fortschreitender internationaler Auslagerung von (Teilen von) Geschäftsprozessen wohl weiter an Bedeutung gewinnen wird.

Über die Prüfung von Workflows zur Entwurfszeit hinaus lässt sich Reveal für weitere Zwecke einsetzen. Dazu zählt die Workflow-Reparatur bzw. die Behebung struktureller Schwachstellen, die Unterstützung dynamischer Analysen, also auch Laufzeit-Prüfungen, und die Unterstützung von Audits nach der Workflow-Ausführung.

Reveal und Schwachstellenbehebung
Bei der Workflow-Prüfung gefundene strukturelle Schwachstellen sollten vor der Ausführung des Workflows behoben werden, um Compliance-Verstöße zu vermeiden. Anstelle einer manuellen Reparatur kommt auch eine automatisierte Reparatur in Frage. Höhn (2010) beschreibt ein Verfahren, welches einen Workflow entsprechend anzugebender Vorlagen automatisiert umstellt und beispielsweise fehlende Aktivitäten einfügt.

Da Reveal zu jeder Schwachstelle die genauen Umstände protokolliert und in automatisiert verwertbarer Form zur Verfügung stellt, ist eine Kombination der Verfahren möglich. Nach der einmaligen Sequenz *Prüfung, dann Reparatur* ist auch eine mehrfache Wiederholung denkbar. So kann sichergestellt werden, dass eventuell durch die Reparatur hervorgerufene Schwachstellen gefunden werden. Bei einem wiederholten Aufruf müsste aber dem Umstand Rechnung getragen werden, dass Zyklen denkbar sind, in denen zwei Schwachstellen abwechselnd gefunden werden und die Reparatur die jeweils andere auslöst.

Reveal und Monitoring
Wie im Abschnitt 1.3.2 gezeigt, kann die Compliance-Prüfung nicht nur auf verschiedenen Ebenen, sondern auch zu verschiedenen Zeitpunkten stattfinden. Der für die vorliegende Arbeit gewählte Zeitpunkt der Entwurfszeit erlaubt statische Analysen, bei denen nur das Workflow-Modell vorliegt. Eine dynamische Analyse durch einen Monitor, das heißt während der Ausführung des Workflows, ist immer dann sinnvoll, wenn die statische Analyse aufgrund fehlender Informationen keine eindeutigen Ergebnisse liefern kann. Dies ist beispielsweise häufig bei Kontrollen nach dem „Vier-Augen-Prinzip“ der Fall, weil dafür die genaue Rollen- bzw. Benutzerbelegung im Moment der Ausführung bekannt sein muss. *Liveness*-Eigenschaften lassen sich oft nur so oder nach der Ausführung analysieren.

Häufig verwenden Monitore die Ergebnisse der statischen Analyse, um gezielt an potentiellen Schwachpunkten die Ausführung zu überwachen. Der Vorteil der Kombination liegt auf der Hand: eindeutige Schwachstellen können dank der statischen Analyse vor der Ausführung behoben werden, und die Anzahl der von der dynamischen Analyse während der Laufzeit zu überwachenden Stellen wird durch die statische Analyse reduziert. Wie erfolgreich dies auf Quellcode-Ebene sein kann, zeigt Livshits (2006).

Reveal führt die statische Analyse auf Workflow-Ebene durch und identifiziert dabei all jene potentiellen Schwachstellen, die sich wegen fehlender Informationen nicht eindeutig bewerten lassen (Typ *UNRES* in Abschnitt 3.4). Damit ist die Datengrundlage für dynamische Verfahren gelegt, die diese Stellen gezielt während der Workflow-Ausführung überwachen können.

Reveal und Workflow-Rekonstruktion
Die zeitlich letzte Prüfmöglichkeit besteht in der Auswertung von Logs, also Aufzeichnungen von Workflow-Ausführungen. Aus den einzelnen Logeinträgen, die typischerweise aus den Ausführungszeitpunkten und weiteren Ausführungsdaten einzelner Aktivitäten bestehen, muss dabei der tatsächliche Ablauf des Workflows rekonstruiert werden. Abweichungen vom geplanten Ablauf, wie ihn das Workflow-Modell darstellt, weisen auf mögliche Regelverstöße hin.

Während der statischen Analyse generiert Reveal als Nebenprodukt alle Pfade durch einen Workflow. Diese bei der Rekonstruktion als *Traces* bezeichneten Pfade können anschließend zum Vergleich mit dem erwarteten Ablauf herangezogen werden. Außerdem lässt sich Reveal nicht nur auf das ursprüngliche Workflow-Modell, sondern auch auf ein rekonstruiertes Workflow-Modell anwenden. Accorsi und Wonnemann (2010) stellen ein Verfahren zur Rekonstruktion und zum automatisierten Audit vor. Dieses und weitere Verfahren können die vorher lokalisierten strukturellen Schwachstellen für ein gezieltes Audit verwenden.

Über den primären Anwendungsfall der Prüfung von Workflows zur Entwurfszeit hinaus unterstützt Reveal also auch die Schwachstellenbehebung sowie Laufzeit- und Logzeit-Analysen. Bezogen auf die aktuelle Sicherheitslücke (siehe Meldung vom 23. Juni 2011, Heise Online (2011)) bei der Verwendung der elektronischen Gesundheitskarte (EGK) beispielsweise ließe sich damit untersuchen, welche Patientendaten in EGK-basierten Geschäftsprozessen gefährdet sind und welche Aktivitäten im Prozess *nach* der gefährdeten PIN-Eingabe stehen und somit potentiell unsicher sind.

Literaturverzeichnis

[104th Congress of the USA 1996] 104th Congress of the USA: *Health Insurance Portability and Accountability Act (HIPAA)*. 1996

[106th Congress of the USA 1999] 106th Congress of the USA: *Gramm-Leach-Bliley Act (GLB)*. 1999

[107th Congress of the USA 2002] 107th Congress of the USA: *Sarbanes-Oxley Act of 2002*. 2002. – URL `http://www.sec.gov/about/laws/soa2002.pdf`

[Abbott et al. 1976] Abbott, R. ; Chin, J. ; Donnelley, J. ; Konigsford, W. ; Tokubo, S. ; Webb, D.: Security Analysis and Enhancements of Computer Operating Systems (RISOS Study) / ICET, Nat. Bureau of Standards, Washington, DC 20234. 1976. – Forschungsbericht. NBSIR 76-1041

[Accorsi 2008] Accorsi, Rafael: Automated Privacy Audits to Complement the Notion of Control for Identity Management. In: *Policies and Research in Identity Management* Bd. 261. Springer-Verlag, 2008, S. 39–48

[Accorsi et al. 2011] Accorsi, Rafael ; Lowis, Lutz ; Sato, Yoshinori: Automatisierte Compliance-Zertifizierung Cloud-basierter Geschäftsprozesse. In: *Wirtschaftsinformatik* 53 (2011), Nr. 3, S. 139–149

[Accorsi et al. 2008] Accorsi, Rafael ; Sato, Yoshinori ; Kai, Satoshi: Compliance Monitor for Early Warning Risk Determination. In: *Wirtschaftsinformatik* 50 (2008), October, Nr. 5

[Accorsi und Wonnemann 2010] Accorsi, Rafael ; Wonnemann, Claus: Auditing Workflow Executions against Dataflow Policies. In: *Lecture Notes in Business Information Processing* Bd. 47, Springer, 2010, S. 207–217

[Accorsi und Wonnemann 2011] Accorsi, Rafael ; Wonnemann, Claus: *InDico: Information Flow Analysis of Business Processes for Confidentiality Requirements*. In: *Proceedings of the ERCIM Workshop on Security and Trust Management*. Athens, Greece : Springer, 2011 (Lecture Notes in Computer Science)

[Adam et al. 1998] Adam, Nabil R. ; Atluri, Vijayalakshmi ; Huang, Wei-Kuang: Modeling and Analysis of Workflows Using Petri Nets. In: *J. Intell. Inf. Syst.* 10 (1998), Nr. 2, S. 131–158. – ISSN 0925-9902

[Alvarez und Petrovic 2003] Alvarez, Gonzalo ; Petrovic, Slobodan: A new taxonomy of Web attacks suitable for efficient encoding. In: *Computers & Security* 22 (2003), Nr. 5, S. 435 – 449. – ISSN 0167-4048

[Askarov et al. 2008] Askarov, Aslan ; Hunt, Sebastian ; Sabelfeld, Andrei ; Sands, David: Termination-Insensitive Noninterference Leaks More Than Just a Bit. In: *ESORICS '08: Proceedings of the 13th European Symposium on Research in Computer Security*. Berlin, Heidelberg : Springer-Verlag, 2008, S. 333–348. – ISBN 978-3-540-88312-8

[Aslam 1995] Aslam, Taimur: *A Taxonomy of Security Faults in the Unix Operating System*, Purdue University, Dissertation, August 1995

[Avizienis et al. 2004] Avizienis, Algirdas ; Laprie, Jean-Claude ; Randell, Brian ; Landwehr, Carl: Basic Concepts and Taxonomy of Dependable and Secure Computing. In: *IEEE Trans. on Dependable and Secure Computing* 1 (2004), Jan.-March, Nr. 1, S. 11–33. – ISSN 1545-5971

[Awad et al. 2009] Awad, Ahmed ; Smirnov, Sergey ; Weske, Mathias: Resolution of Compliance Violation in Business Process Models: A Planning-Based Approach. In: *On the Move to Meaningful Internet Systems: OTM 2009* Bd. 5870. Springer, 2009, S. 6–23. – URL `http://dx.doi.org/10.1007/978-3-642-05148-7_4`

[Bace et al. 2006] Bace, John ; Rozwell, Carol ; Feiman, Joseph ; Kirwin, Bill: *Understanding the Cost of Compliance*. 2006

[Ben-Eliahu 2008] Ben-Eliahu, Ziv: *Business Processes of the world unite*. `http://www.cs.bgu.ac.il/~bpmn/BPDs`. 2008

[Berghel 2005] Berghel, Hal: The two sides of ROI: return on investment vs. risk of incarceration. In: *Commun. ACM* 48 (2005), Nr. 4, S. 15–20. – ISSN 0001-0782

[Bessey et al. 2010] Bessey, Al ; Block, Ken ; Chelf, Ben ; Chou, Andy ; Fulton, Bryan ; Hallem, Seth ; Henri-Gros, Charles ; Kamsky, Asya ; McPeak, Scott ; Engler, Dawson: A few billion lines of code later: using static analysis to find bugs in the real world. In: *Communications of the ACM* 53 (2010), Nr. 2, S. 66–75

[Bisbey II und Hollingworth 1978] Bisbey II, R. ; Hollingworth, D.: Protection Analysis: Final Report. / University of Southern California Information Sciences Institute. 1978. – Forschungsbericht. ISI/SR-78-13

[Bishop 2004] Bishop, Matt: *Introduction to Computer Security*. Addison-Wesley, Pearson Education, 2004

[Boehm 1981] Boehm, Barry W.: *Software Engineering Economics*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 1981. – ISBN 0138221227

[Breaux 2009] Breaux, Travis D.: *Legal Requirements Acquisition for the Specification of Legally Compliant Information Systems*. Raleigh, North Carolina, USA, North Carolina State University, Dissertation, April 2009

[Brinkley und Schell 1995] Brinkley, Donald L. ; Schell, Roger R.: What Is There to Worry About? An Introduction to The Computer Security Problem. In: *Information Security: An Integrated Collection of Essays*, IEEE Computer Society Press, 1995, S. 11–39

[Brumley und Boneh 2003] Brumley, David ; Boneh, Dan: Remote timing attacks are practical. In: *SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium*. Berkeley, CA, USA : USENIX Association, 2003, S. 1–1

[Bundesamt für Sicherheit in der Informationstechnik (BSI) 2008] Bundesamt für Sicherheit in der Informationstechnik (BSI): *IT-Grundschutz*. http://www.bsi.bund.de. 2008

[Bundesministerium für Justiz (BMJ) 2009] Bundesministerium für Justiz (BMJ): *Bundesdatenschutzgesetz (BDSG)*. 2009

[Cabanillas et al. 2010] Cabanillas, Cristina ; Resinas, Manuel ; Ruiz-Cortés, Antonio: Hints on how to face business process compliance. In: Resinas, Manuel (Hrsg.) ; Ruiz-Cortés, Antonio (Hrsg.) ; Pastor, Joan A. (Hrsg.) ; Sancho, María R. (Hrsg.): *Proceedings of the JISBD* Bd. 4, 2010, S. 26–32

[Christey 2006] Christey, Steve: *Preliminary List of Vulnerability Examples for Researchers (PLOVER)*. http://cwe.mitre.org/about/sources.html. 2006

[Cohen 1997] Cohen, Fred: Information System Attacks: A Preliminary Classification Scheme. In: *Computers & Security* 16 (1997), Nr. 1, S. 29–46

[Curtis et al. 1992] Curtis, Bill ; Kellner, Marc I. ; Over, Jim: Process modeling. In: *Commun. ACM* 35 (1992), Nr. 9, S. 75–90. – ISSN 0001-0782

[Dadam et al. 2009] Dadam, Peter ; Reichert, Manfred ; Rinderle-Ma, Stefanie ; Goeser, Kevin ; Kreher, Ulrich ; Jurisch, Martin: Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: 'Correctness by Construction' und flexible, robuste Ausführung von Unternehmensprozessen / Universität Ulm. 2009. – Forschungsbericht. DBIS-Report

[Denning und Denning 1977] Denning, Dorothy E. ; Denning, Peter J.: Certification of programs for secure information flow. In: *Commun. ACM* 20 (1977), Nr. 7, S. 504–513. – ISSN 0001-0782

[Department of Defense 1980] Department of Defense: *Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. November 1980. – MIL-STD-1629A

[Du und Mathur 1997] Du, Wenliang ; Mathur, Aditya P.: Categorization of Software Errors that led to Security Breaches. In: *In 21st National Information Systems Security Conference*, 1997, S. 392–407

[Eid und Caldwell 2006] Eid, Tom ; Caldwell, French: Finance and Audit GRC Software Market is Expanding / Gartner Research. 2006. – Forschungsbericht. http://www.gartner.com/it/content/498300/498334/risk_research.pdf

[Emerson 2008] Emerson, E. A.: The Beginning of Model Checking: A Personal Perspective. In: Grumberg, Orna (Hrsg.) ; Veith, Helmut (Hrsg.): *25 Years of Model Checking*. Berlin, Heidelberg : Springer-Verlag, 2008, S. 27–45. – URL http://dx.doi.org/10.1007/978-3-540-69850-0_2. – ISBN 978-3-540-69849-4

[Europäisches Parlament 1995] Europäisches Parlament: *Richtlinie 95/46/EG zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr*. 1995

[Fan et al. 2007] Fan, Shaokun ; Dou, Wanchun ; Chen, Jinjun: Dual Workflow Nets: Mixed Control/Data-Flow Representation for Workflow Modeling and Verification. In: *Advances in Web and Network Technologies, and Information Management* Bd. 4537. Springer, 2007, S. 433–444

[Financial Executives International (FEI) 2004] Financial Executives International (FEI): *FEI Survey*. www.financialexecutives.org. 2004

[Financial Executives International (FEI) 2007] Financial Executives International (FEI): *FEI Survey*. www.financialexecutives.org. 2007

[Georgakopoulos et al. 1995] Georgakopoulos, Diimitrios ; Hornick, Mark ; Sheth, Amit: An overview of workflow management: From process modeling to workflow automation infrastructure. In: *Distributed and Parallel Databases* 3 (1995), S. 119–153. – URL http://dx.doi.org/10.1007/BF01277643. – 10.1007/BF01277643. – ISSN 0926-8782

[Georgieva 2010] Georgieva, Konstantina: Conducting FMEA over the software development process. In: *SIGSOFT Softw. Eng. Notes* 35 (2010), May, S. 1–5. – URL http://doi.acm.org/10.1145/1764810.1764819. – ISSN 0163-5948

[Giblin et al. 2005] Giblin, Christopher ; Liu, Alice Y. ; Müller, Samuel ; Pfitzmann, Birgit ; Zhou, Xin: Regulations Expressed As Logical Models (REALM). In: *Proceeding of the 2005 conference on Legal Knowledge and Information Systems*. Amsterdam, The Netherlands, The Netherlands : IOS Press, 2005, S. 37–48. – ISBN 1-58603-576-2

[Giblin et al. 2006] Giblin, Christopher J. ; Mueller, Samuel ; Pfitzmann, Birgit: *From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation.* 2006. – IBM Research Report 99682

[Hammer 2009] Hammer, Christian: *Information Flow Control for Java - A Comprehensive Approach basedon Path Conditions in Dependence Graphs*, Universität Karlsruhe (TH), Fak. f. Informatik, Dissertation, July 2009. – URL `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000012049`. – ISBN 978-3-86644-398-3

[Hammer und Snelting 2009] Hammer, Christian ; Snelting, Gregor: Flow-Sensitive, Context-Sensitive, and Object-sensitive Information Flow Control Based on Program Dependence Graphs. In: *International Journal of Information Security* 8 (2009), December, Nr. 6, S. 399–422. – Supersedes ISSSE and ISoLA 2006

[Hansen und Neumann 2005] Hansen, Hans R. ; Neumann, Gustaf ; Bea (Hrsg.) ; Friedl (Hrsg.) ; Schweitzer (Hrsg.): *Wirtschaftsinformatik 1: Grundlagen und Anwendungen.* Lucius und Lucius, 2005

[Hartman 2007] Hartman, Thomas E.: *The Cost of Being Public in the Era of Sarbanes-Oxley.* www.foley.com. 2007. – Foley & Lardner LLP Study

[Heise Online 2011] Heise Online: *Elektronische Gesundheitskarte: Gematik legt Bericht zur Sicherheitslücke vor.* http://www.heise.de/newsticker/meldung/Elektronische-Gesundheitskarte-Gematik-legt-Bericht-zur-Sicherheitsluecke-vor-1266836.html. Juni 2011

[Hogan 1988] Hogan, Carole B.: Protection imperfect: the security of some computing environments. In: *SIGOPS Oper. Syst. Rev.* 22 (1988), Nr. 3, S. 7–27. – ISSN 0163-5980

[Howard und Longstaff 1998] Howard, John D. ; Longstaff, Thomas A.: A Common Language for Computer Security Incidents / Sandia National Laboratories. 1998. – Forschungsbericht. SAND98-8667

[Howard et al. 2005] Howard, Michael ; LeBlanc, David ; Viega, John: *19 Deadly Sins of Software Security.* McGraw-Hill Osborne Media, 2005. – ISBN 0072260858

[Howard et al. 2009] Howard, Michael ; LeBlanc, David ; Viega, John: *24 Deadly Sins Of Software Security: Programming Flaws And How To Fix Them.* McGraw-Hill Osborne Media, 2009. – ISBN 9780071626750

[Huynh und Miller 2010] Huynh, Toan ; Miller, James: An empirical investigation into open source web applications implementation vulnerabilities. In: *Empirical Software Engineering* 15 (2010), October, Nr. 5, S. 556–576. – ISSN 1382-3256

[Höhn 2010] Höhn, Sebastian: *Prozess-Rewriting zur Durchsetzung von Compliance in flexiblen Geschäftsprozessen*, Albert-Ludwigs-Universität Freiburg im Breisgau, Dissertation, 2010

[International Organization for Standardization 2005] International Organization for Standardization: *ISO/IEC 17799:2005: Information technology, Security techniques, Code of practice for information security management.* 2005

[IT Governance Institute 2007] IT Governance Institute: *Cobit 4.1.* `http://www.isaca.org/Knowledge-Center/cobit/Pages/Downloads.aspx`. 2007

[Jensen 1996] Jensen, Kurt: *Coloured Petri nets: basic concepts, analysis methods and practical use.* Bd. 1. London, UK : Springer-Verlag, 1996. – ISBN 3-540-60943-1

[Jiwnani und Zelkowitz 2002] Jiwnani, K. ; Zelkowitz, M.: Maintaining Software with a Security Perspective. In: *Proc. ICSM Conference.* Los Alamitos, CA, USA : IEEE Computer Society, 2002, S. 194–204. – ISBN 0-7695-1819-2

[Katt et al. 2009] Katt, Basel ; Zhang, Xinwen ; Hafner, Michael: Towards a Usage Control Policy Specification with Petri Nets. In: *OTM '09: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems.* Berlin, Heidelberg : Springer-Verlag, 2009, S. 905–912. – ISBN 978-3-642-05150-0

[Krsul 1998] Krsul, Ivan V.: *Software Vulnerability Analysis*, Purdue University, Dissertation, May 1998

[Kähmer 2009] Kähmer, Martin: *ExPDT - Vergleichbarkeit von Richtlinien für Selbstregulierung und Selbstdatenschutz*, Universität Freiburg, IIG Telematik, Dissertation, 2009

[Lampson 1973] Lampson, Butler W.: A note on the confinement problem. In: *Commun. ACM* 16 (1973), Nr. 10, S. 613–615. – ISSN 0001-0782

[Landwehr et al. 1994] Landwehr, C. ; Bull, A. ; McDermott, J. ; Choi, W.: A Taxonomy of Computer Program Security Flaws. In: *Computing Surveys 26* 3 (1994), S. 211–254

[Langweg und Snekkenes 2004] Langweg, Hanno ; Snekkenes, Einar: A Classification of Malicious Software Attacks. In: *Proc. of 23rd IEEE PCC Conference*, 2004, S. 827 – 832

[Lindqvist und Jonsson 1997] Lindqvist, Ulf ; Jonsson, Erland: How to Systematically Classify Computer Security Intrusions. In: *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997, S. 154–163

[Lindstrom 2004] Lindstrom, Pete: Attacking and Defending Web Services / Spire Security. 2004. – Forschungsbericht. http://www.forumsys.com/resources/resources/whitepapers/Attacking_and_Defending_WS.pdf

[Lipner 1975] Lipner, Steven B.: A comment on the confinement problem. In: *SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles*. New York, NY, USA : ACM, 1975, S. 192–196

[Lipton 1975] Lipton, R.: The reachability problem requires exponential space / Yale University. 1975. – Forschungsbericht. Nicht mehr lieferbar.

[Liu und Kumar 2005] Liu, Rong ; Kumar, Akhil: An Analysis and Taxonomy of Unstructured Workflows. In: *Business Process Management* Bd. 3649. Springer, 2005, S. 268–284. – URL http://dx.doi.org/10.1007/11538394_18

[Liu et al. 2007] Liu, Y. ; Müller, S. ; Xu, K.: A static compliance-checking framework for business process models. In: *IBM Syst. J.* 46 (2007), Nr. 2, S. 335–361. – ISSN 0018-8670

[Livshits 2006] Livshits, Benjamin: *Improving Software Security With Precise Static and Runtime Analysis*, Stanford University, Dissertation, Dezember 2006

[Lowis und Accorsi 2009] Lowis, Lutz ; Accorsi, Rafael: On a Classification Approach for SOA Vulnerabilities. In: *Proc. IEEE Workshop on Security Aspects of Process and Services Eng. (SAPSE)*, IEEE Computer Press, 2009, S. 439–444

[Lowis und Accorsi 2011] Lowis, Lutz ; Accorsi, Rafael: Vulnerability Analysis in SOA-Based Business Processes. In: *IEEE Transactions on Services Computing* 4 (2011), Nr. 3, S. 230–242. – ISSN 1939-1374

[Lowis et al. 2008] Lowis, Lutz ; Höhn, Sebastian ; Gilliot, Maike: Vulnerability Effect Propagation in Service-Oriented Architectures. In: *Proceedings of Sicherheit 2008* Bd. 128. Saarbrücken, Germany : Springer, 2008, S. 473–484. – ISBN 978-3-88579-222-2

[Martin et al. 2005] Martin, Michael ; Livshits, Benjamin ; Lam, Monica S.: Finding Application Errors and Security Flaws Using PQL: a Program Query Language. In: *20th ACM SIGPLAN OOPSLA Conference*. New York, NY, USA : ACM, 2005 (OOPSLA '05), S. 365–383. – URL http://doi.acm.org/10.1145/1094811.1094840. – ISBN 1-59593-031-0

[Meda et al. 2010] Meda, Hema S. ; Sen, Anup K. ; Bagchi, Amitava: On Detecting Data Flow Errors in Workflows. In: *J. Data and Information Quality* 2 (2010), Nr. 1, S. 1–31. – ISSN 1936-1955

[MITRE Corporation 2011a] MITRE Corporation: *Common Attack Pattern Enumeration and Classification (CAPEC)*. http://capec.mitre.org/. 2011

[MITRE Corporation 2011b] MITRE Corporation: *Common Weakness Enumeration (CWE)*. http://cwe.mitre.org/. 2011

[MITRE Corporation 2011c] MITRE Corporation: *Open Vulnerability and Assessment Language (OVAL)*. http://oval.mitre.org/. 2011

[Moore et al. 2001] Moore, Andrew P. ; Ellison, Robert J. ; Linger, Richard C.: Attack Modeling for Information Security and Survivability / Software Engineering Institute, Carnegie Mellon. 2001. – Forschungsbericht. CMU/SEI-2001-TN-001

[Murata 1989] Murata, Tadeo: Petri nets: Properties, analysis and applications. In: *Proceedings of the IEEE* 77 (1989), April, Nr. 4, S. 541–580. – URL `http://dx.doi.org/10.1109/5.24143`. – ISSN 00189219

[Müller und Pfitzmann 1999] Müller, Günter (Hrsg.) ; Pfitzmann, Andreas (Hrsg.): *Mehrseitige Sicherheit in der Kommunikationstechnik*. Addison-Wesley, 1999

[Müller und Terzidis 2008] Müller, Günter ; Terzidis, Orestis: IT-Compliance und IT-Governance. In: *WIRTSCHAFTSINFORMATIK* 50 (2008), S. 341–343. – URL `http://dx.doi.org/10.1007/s11576-008-0074-5`. – 10.1007/s11576-008-0074-5. – ISSN 0937-6429

[Neuman Levine 2009] Neuman Levine, Gertrude: Defining defects, errors, and service degradations. In: *SIGSOFT Softw. Eng. Notes* 34 (2009), Nr. 2, S. 1–14. – ISSN 0163-5948

[Neumann und Parker 1989] Neumann, Peter G. ; Parker, Donn B.: A Summary of Computer Misuse Techniques. In: *12th Nat. Computer Security Conf., Baltimore, MD*, October 1989, S. 396–406

[North 2008] North, Jeffrey: The Total Economic Impact (tm) Of Guardium Database Security, Monitoring, and Auditing / Forrester Consulting. 2008. – Forschungsbericht. `www.Guardium.com/ForresterROI`

[OASIS 2005] OASIS: *Extended Access Control Markup Language (XACML)*. `www.oasis-open.org/committees/xacml/`. 2005

[Object Management Group (OMG)] Object Management Group (OMG): *Governance, Risk Management and Compliance Roundtable*. http://www.grcroundtable.org/

[OECD 1980] OECD: *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. Organisation for Economic Co-Operation and Development (OECD), 1980. – ISBN 9789264196391

[OMG 2011] OMG: *Business Process Model and Notation*. January 2011. – http://www.omg.org/spec/BPMN/2.0/PDF

[Organization for the Advancement of Structured Information Standards (OASIS) 2007] Organization for the Advancement of Structured Information Standards (OASIS): *Business Process Execution Language (BPEL)*. 2007. – URL `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`

[Ouyang et al. 2006] Ouyang, Chun ; van der Aalst, Wil M.P. ; Dumas, Marlon ; ter Hofstede, Arthur H.M.: *Translating BPMN to BPEL*. 2006. – URL `http://eprints.qut.edu.au/archive/00003615/`

[OWASP Foundation 2009] OWASP Foundation: *Open Web Application Security Project (OWASP)*. http://www.owasp.org/. 2009

[Padovan et al. 2005] Padovan, Boris ; Sackmann, Stefan ; Eymann, Torsten ; Pippow, Ingo ; Strüker, Jens: *Electronic Commerce Enquête 2005 - 10 Jahre Electronic Commerce*. Forschungsbericht Uni Freiburg, IIG Telematik. 2005

[Park und Sandhu 2004] Park, Jaehong ; Sandhu, Ravi: The UCONABC usage control model. In: *ACM Trans. Inf. Syst. Secur.* 7 (2004), Nr. 1, S. 128–174. – ISSN 1094-9224

[Parrend und Frenot 2007] Parrend, Pierre ; Frenot, Stephane: Java Components Vulnerabilities - An Experimental Classification Targeted at the OSGi Platform / INRIA. 2007. – Forschungsbericht. INRIA/RR–6231

[Payment Card Industry 2009] Payment Card Industry: *PCI DSS*. July 2009

[Peterson 1977] Peterson, James L.: Petri Nets. In: *ACM Comput. Surv.* 9 (1977), Nr. 3, S. 223–252. – ISSN 0360-0300

[Piessens et al. 2001] Piessens, Frank ; De Decker, Bart ; De Win, Bart: Developing Secure Software - A survey and classification of common software vulnerabilities. In: *Proc. of the IICIS Conference*, 2001, S. 27–40

[Ponemon Institute 2010] Ponemon Institute: PCI DSS Trends 2010: QSA Insights Report / Ponemon Institute. 2010. – Forschungsbericht. `iss.thalesgroup.com/en/l/program/PCITrendsReport.aspx`

[Pretschner et al. 2006] Pretschner, Alexander ; Hilty, Manuel ; Basin, David: Distributed usage control. In: *Commun. ACM* 49 (2006), Nr. 9, S. 39–44. – ISSN 0001-0782

[Reichert und Dadam 1998] Reichert, Manfred ; Dadam, Peter: ADEPT flex - Supporting Dynamic Changes of Workflows Without Loosing Control. In: *Journal of Intelligent Information Systems* 10 (1998), S. 93–129

[Rusinkiewicz und Sheth 1995] Rusinkiewicz, Marek ; Sheth, Amit: *Specification and execution of transactional workflows*. S. 592–620. In: *Modern Database Systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995. – ISBN 0-201-59098-0

[Sabelfeld und Myers 2003] Sabelfeld, Andrei ; Myers, Andrew C.: Language-Based Information-Flow Security. In: *IEEE Journal on Selected Areas in Communications* 21 (2003), S. 2003

[Sackmann und Kähmer 2008] Sackmann, Stefan ; Kähmer, Martin: ExPDT: A Policy-based Approach for Automating Compliance. In: *Wirtschaftsinformatik* 50 (2008), S. 366–374

[Sackmann et al. 2008] Sackmann, Stefan ; Kähmer, Martin ; Gilliot, Maike ; Lowis, Lutz: A Classification Model for Automating Compliance. In: *E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services, IEEE Conference and Fifth IEEE Conference* 0 (2008), S. 79–86. – ISSN 1530-1354

[Sackmann et al. 2009] Sackmann, Stefan ; Lowis, Lutz ; Kittel, Kai: Selecting Services in Business Process Execution: a Risk-based Approach. In: *Proceedings of the Wirtschaftsinformatik*, 2009, S. 357–366

[Sadiq et al. 2007] Sadiq, Shazia ; Governatori, Guido ; Namiri, Kioumars: Modeling Control Objectives for Business Process Compliance. In: *Business Process Management* Bd. 4714, Springer Berlin / Heidelberg, 2007, S. 149–164. – ISBN 978-3-540-75182-3

[Sadiq et al. 2004] Sadiq, Shazia ; Orlowska, Maria ; Sadiq, Wasim ; Foulger, Cameron: Data flow and validation in workflow modelling. In: *ADC '04: Proceedings of the 15th Australasian database conf*, Australian Computer Society, Inc., 2004, S. 207–214

[Sadiq und Orlowska 1997] Sadiq, Wasim ; Orlowska, Maria E.: On Correctness Issues in Conceptual Modeling of Workflows. In: *Proceedings of the 5th European Conference on Information Systems (ECIS '97)*, 1997, S. 22

[Sadiq und Orlowska 1999] Sadiq, Wasim ; Orlowska, Maria E.: Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. (1999), S. 195–209. – URL `http://portal.acm.org/citation.cfm?id=646087.679894`. ISBN 3-540-66157-3

[Sadiq und Orlowska 2000] Sadiq, Wasim ; Orlowska, Maria E.: Analyzing process models using graph reduction techniques. In: *Inf. Syst.* 25 (2000), April, S. 117–134. – URL http://portal.acm.org/citation.cfm?id=344358.344369. – ISSN 0306-4379

[SANS 2009] SANS: *SANS Top 20 Security Risks*. http://www.sans.org/top-cyber-security-risks/. 2009

[Schneider 2000] Schneider, Fred B.: Enforceable security policies. In: *ACM Trans. Inf. Syst. Secur.* 3 (2000), February, S. 30–50. – URL http://doi.acm.org/10.1145/353323.353382. – ISSN 1094-9224

[Schneier 1999] Schneier, Bruce: Attack Trees. In: *Dr. Dobb's Journal* 24 (1999), S. 21–29

[Sloman 2000] Sloman, Morris: *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems*. http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml. 2000. – Imperial College of Science, Technology and Medicine, Department of Computing

[Stohr und Zhao 2001] Stohr, Edward A. ; Zhao, J. L.: Workflow Automation: Overview and Research Issues. In: *Information Systems Frontiers* 3 (2001), Nr. 3, S. 281–296. – ISSN 1387-3326

[Sun et al. 2006] Sun, Sherry X. ; Zhao, J. L. ; Nunamaker, Jay F. ; Sheng, Olivia R. L.: Formulating the Data-Flow Perspective for Business Process Management. In: *Information Systems Research* 17 (2006), Nr. 4, S. 374–391

[Sun et al. 2004] Sun, Sherry X. ; Zhao, J. L. ; Sheng, Olivia R. L.: Data Flow Modeling and Verification in Business Process Management. In: *Proceedings of the AMCIS*, 2004, S. 4064–4073

[Sundari et al. 2007] Sundari, M. H. ; Sen, Anup K. ; Bagchi, Amitava: Detecting Data Flow Errors in Workflows: A Systematic Graph Traversal Approach. In: *Proceedings of the Workshop on Information Technology and Systems (WITS)*, 2007, S. 133–138

[Svirskas et al. 2007] Svirskas, Adomas ; Courbis, Carine ; Molva, Refik ; Bedzinskas, Justinas: Compliance Proofs for Collaborative Interactions using Aspect-Oriented Approach. In: *Services, IEEE Congress on* 0 (2007), S. 33–40. ISBN 0-7695-2926-7

[Tilbug University 2008] Tilbug University: *Compliance-driven Models, Languages, and Architectures for Services (COMPAS): D2.1 State of the art in the field of compliance languages*. http://www.compas-ict.eu/results.php. 2008

[Trcka et al. 2008] Trcka, Nikola ; van der Aalst, W.M.P. ; Sidorova, Natalia: Analyzing Control Flow and Data Flow in Workflow Processes in a Unified Way / Eindhoven University of Technology. 2008. – Forschungsbericht. report 08-31

[Trcka et al. 2009] Trcka, Nikola ; van der Aalst, W.M.P. ; Sidorova, Natalia: Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows. In: *CAiSE '09: Proceedings of the 21st International Conference on Advanced Information Systems Engineering* Bd. 5565. Berlin, Heidelberg : Springer-Verlag, 2009, S. 425–439. – ISBN 978-3-642-02143-5

[Tsipenyuk et al. 2005] Tsipenyuk, Katrina ; Chess, Brian ; McGraw, Gary: Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors. In: *IEEE Security and Privacy* 3 (2005), Nr. 6, S. 81–84. – ISSN 1540-7993

[van der Aalst 1998] van der Aalst, W. M. P.: The Application of Petri Nets to Workflow Management. In: *The Journal of Circuits, Systems and Computers* 8 (1998), Nr. 1, S. 21–66. – URL `http://wwwis.win.tue.nl/~{}wsinwa/Publications/p53.PDF`

[van der Aalst 1996] van der Aalst, W.M.P.: Petri-net-based Workflow Management Software. In: Sheth, A. (Hrsg.): *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, 1996, S. 114–118

[van der Aalst et al. 2002] van der Aalst, W.M.P. ; Hirnschall, Alexander ; Verbeek, H. M. W.: An Alternative Way to Analyze Workflow Graphs. In: *Proceedings of the 14th Intl. Conf. on Advanced Information Systems Engineering*. London, UK, UK : Springer-Verlag, 2002 (CAiSE '02), S. 535–552. – ISBN 3-540-43738-X

[van der Aalst et al. 2003] van der Aalst, W.M.P. ; ter Hofstede, A.H.M. ; Kiepuszewski, B. ; Barros, A.P.: Workflow Patterns. In: *Distributed and Parallel Databases* 14 (2003), S. 5–51. – 10.1023/A:1022883727209. – ISSN 0926-8782

[van der Aalst und van Hee 2002] van der Aalst, W.M.P. ; van Hee, Kees: *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002. – ISBN 978-0-262-01189-1

[Vanden Berghe et al. 2005] Vanden Berghe, Chris ; Riordan, James ; Piessens, Frank: A Vulnerability Taxonomy Methodology applied to Web Services. In: *Proceedings of the 10th Nordic Workshop on Secure IT Systems (NordSec)*, 2005, S. 49–62

[Vanhatalo et al. 2007] Vanhatalo, Jussi ; Völzer, Hagen ; Leymann, Frank: Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In: *Service-Oriented Computing, ICSOC* Bd. 4749. Springer, 2007, S. 43–55. – URL `http://dx.doi.org/10.1007/978-3-540-74974-5_4`

[Vesely et al. 1981] Vesely, William E. ; Goldberg, Francine F. ; Haasl, David F. ; Roberts, Norman H.: *Fault Tree Handbook*. 1981. – NUREG-0492

[W3C 2007] W3C: *WS Policy*. http://www.w3.org/TR/ws-policy/. 2007. – v1.5

[Web Appl. Sec. Consortium 2004] Web Appl. Sec. Consortium: *Web Application Security Consortium Threat Classification*. http://www.webappsec.org/ projects/-threat/. 2004

[Weber et al. 2005] Weber, Sam ; Karger, Paul A. ; Paradkar, Amit: A Software Flaw Taxonomy: Aiming Tools At Security. In: *Software Engineering for Secure Systems*. St. Louis, Missouri, USA : ACM, 2005, S. 1–7

[Workflow Management Coalition 1999] Workflow Management Coalition: WFMC Terminology and Glossary / WFMC. 1999. – Forschungsbericht. http://www.wfmc.org/reference-model.html

[Xamit 2009] Xamit: *Datenschutzbarometer 2009 der Xamit Bewertungsgesellschaft mbH*. `http://www.xamit-leistungen.de/studienundtests/index.php`. 2009

[Xu et al. 2006] Xu, Ke ; Liu, Ying ; Wu, Cheng: Guided Reasoning of Complex E-Business Process with Business Bug Patterns. In: *ICEBE '06: Proceedings of the IEEE International Conference on e-Business Engineering*. Washington, DC, USA : IEEE Computer Society, 2006, S. 195–202. – ISBN 0-7695-2645-4

Abbildungsverzeichnis

1.1 Prüfungszeitpunkte 9
1.2 Prüfungsebenen 11
1.3 Arten von Informationflüssen (nach Sabelfeld und Myers (2003)) . . . 12

2.1 Die Entwicklung des Sarbanes-Oxley-Acts (aus Hartman (2007)) 20
2.2 Compliance-Ontologie von Breaux (2009) 32
2.3 Beispiel einer EPK (vereinfacht aus (Hansen und Neumann, 2005, Kap. 2.2.5, S. 237)) 34
2.4 Workflow-Beispiel in Aristaflow (aus (Dadam et al., 2009)) 35
2.5 Beispiel eines Workflows in BPEL 36
2.6 Beispiel eines Workflows in BPMN 36
2.7 Ein einfaches Petrinetz vor und nach dem Feuern 37
2.8 Workflow-Patterns von van der Aalst et al. (2003) 39
2.9 Fallbeispiel: Workflow für Krankenhaus-Behandlung 42
2.10 Vier Kontrollflussmuster strukturierter Workflows (Liu und Kumar, 2005) 46
2.11 Beispiel: Widerspruchsfreie Flüsse ergeben widersprüchliche Kombination (nach Fan, Dou und Chen Fan et al. (2007)) 51
2.12 Beispiel für nicht korrekten Workflow 55
2.13 Elemente für die Prüfung des Daten- und Kontrollflusses 57
2.14 Beispiel-Workflow mit Aktivitäten, ohne Daten-Angaben 60
2.15 Beispiel-Workflow mit Daten-Angaben 60
2.16 Beispiel-Workflow mit Ressourcen-Angaben 61
2.17 Beispiel-Workflow mit Zeitlimits 61
2.18 Beispiel für Widerspruch zwischen DF und KF 62
2.19 *Compliance-checking Framework* von Liu, Müller und Xu, Abbildung aus (Liu et al., 2007, S. 350) 65

3.1 Ablaufübersicht des REVEAL-Verfahrens 67
3.2 Beispiel für ein nicht korrektes Workflow-Modell 70
3.3 Beispiel für ein Workflow-Modell mit Ausnahmeregel 71
3.4 Petrinetz-Modellierung von Workflow-Elementen 72
3.5 Beispiel für ein Workflow-Petrinetz 72
3.6 Standard-Feuerverhalten von Petrinetz-Transitionen 73
3.7 Petrinetze für das Workflow-Muster *Multi-Choice* 74
3.8 Verzweigungsvorlage für ODER-Transitionen 75

3.9 Beispiel für ODER-Verzweigung mit UND-Vereinigung 76
3.10 Muster für Prüfung auf Pfade zum Ende des Workflows 76
3.11 Beispiel für Pfade zum Ende des Workflows 77
3.12 Verzweigungsvorlage für UND-Transitionen 77
3.13 Beispiel für ODER-Verzweigung mit UND-Vereinigung 78
3.14 Vorlage zur Erkennung paralleler, reihenfolgeabhängiger Aktivitäten . . 78
3.15 Beispiel für parallele, reihenfolgeabhängige Aktivitäten 79
3.16 Vorlage für verbotene Elemente 79
3.17 Vorlage für geforderte Elemente 79
3.18 Vorlage für Ortsvorgaben 80
3.19 Beispiel für geforderten Ausführungsort 80
3.20 Vorlage für Reihenfolge „Y erfordert X“ 82
3.21 Vorlage für Reihenfolge „X löst Y aus“ 82
3.22 Vorlage für Reihenfolge „Wenn X und Y, dann zuerst X“ 83
3.23 Beispiel für Reihenfolge „B nach A“ 84
3.24 Vorlage für Ressourcenvorgaben 84
3.25 Vorlage für zwingend zu verarbeitende Datenelemente 85
3.26 Vorlage für sofortige Verwendung nach einer Aktivität 85
3.27 Vorlage für Verwendung bzw. Löschung vor Workflow-Ende 86
3.28 Beispiel für Datenlöschung 86
3.29 Minimale/maximale Ausführungszeitpunkte vor/nach einer Aktivität . 86
3.30 Vorlage für Zeitpunkte bzw. Kontrollaktivitäten vor/nach einer Aktivität 88
3.31 Beispiel für Kontrollaktivitäten zur Zeitpunktkontrolle 89
3.32 Beispiele für Workflows ohne und mit Schleifen 91
3.33 Beispiel für die Pfadzuordnung eindeutiger Token-IDs 93
3.34 Beispiel für die Workflow-Zuordnung von Regeln 94
3.35 Beispiel für die Workflow-Transformation 100
3.36 Klassendiagramme für Reveal-Petrinetze 101
3.37 Plätze mit endgültiger und vorübergehender Bewertung 104
3.38 Beispiel für Plätze mit endgültiger und vorübergehender Bewertung . . 104

4.1 Fallbeispiel-Workflow als Petrinetz 105
4.2 Inkorrekter Workflow 108
4.3 Korrektheitsprüfung mit Reveal 109
4.4 Fallbeispiel-Workflow mit Datenfluss-Vorgaben 110
4.5 Datenflussprüfung: Modalitätsregel eingehalten 111
4.6 Datenflussprüfung: Modalitätsregel verletzt 111
4.7 Datenflussprüfung: Ortsregel eingehalten 112
4.8 Datenflussprüfung: Ressourcenregel verletzt 113
4.9 Datenflussprüfung: Verwendungsregel eingehalten 114
4.10 Fallbeispiel-Workflow mit Datenfluss-Vorgaben 115
4.11 Kontrollflussprüfung: Modalitätsregel eingehalten 116
4.12 Fallbeispiel-Workflow ohne Aktivität *Anmeldung* 117

4.13 Kontrollflussprüfung: negierte Modalitätsregel eingehalten 118
4.14 Kontrollflussprüfung: Ortsregel verletzt 118
4.15 Kontrollflussprüfung: Reihenfolgeregel eingehalten 118
4.16 Kontrollflussprüfung: Einhaltung der Ressourcenregel unklar 119
4.17 Kontrollflussprüfung: Ressourcenregel eingehalten 119
4.18 Kontrollflussprüfung: Zeitlimitregel eingehalten 119
4.19 Behandlungs-Workflow ohne Aktivität *Eintrag in EPA* 120
4.20 Widerspruchsfreiheit: Fehlende Datenelemente bzw. -verwendung . . . 121
4.21 Widerspruchsfreiheit: keine Schwachstellen gefunden 121

Tabellenverzeichnis

2.1 Übersicht über Schwachstellenklassifikationen 45
2.2 Korrektheitseigenschaften aus Trcka et al. (2008) 52
2.3 Datenfluss-*Antipatterns* aus Trcka et al. (2008) 52
2.4 Kontroll- und Datenflussanforderungen aus Trcka et al. (2008) 56
2.5 Zuordnung der Datenfluss-Probleme aus Sadiq et al. (2004) 59
2.6 Zu prüfende Workflow-Eigenschaften 63
2.7 Einordnung bestehender Workflow-Analyseverfahren 64

3.1 Ausprägungen der Reveal Regelangaben 69
3.2 Wahrheitstabelle für Reihenfolgeregeln 81
3.3 Element-Übersetzung Regel in PNML 97
3.4 Element-Übersetzung BPEL in PNML 99
3.5 Element-Übersetzung WSDL in PNML 100

4.1 Zuordnung von HIPAA-Regeln zu Regelelementen 122
4.2 Analysedauer verschiedener Testfälle (in Millisekunden) 123
4.3 Analysedauer bei steigender Anzahl Regeln (in Millisekunden) 124
4.4 Einordnung des Reveal-Verfahrens 125