Integration von Markov Modellen in Fehlerbäume

Alexander Prohaska

Lehrstuhl Software Engineering: Dependability
Technische Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
prohaska@cs.uni-kl.de

Abstract: Sicherheitskritische Systeme bestehen heutzutage aus einer Vielzahl von Komponenten, die meist von unterschiedlichen Herstellern mit verschiedenen Sicherheitsanalysetechniken untersucht werden. Diese Sicherheitsanalysen müssen später vom Systemintegrator zu einer heterogenen Systemanalyse zusammengefügt werden. Im Allgemeinen wird dabei vorausgesetzt, dass ein Markov-Modell nicht mehr als ein Ausgangssignal liefert, um eine modulare Integration und damit auch die Wiederverwendung der Sicherheitsanalysen der Komponenten zu ermöglichen. Diese Einschränkung reduziert jedoch die Auswahl der modularen Kompositionsszenarien. Die hier vorgestellte Arbeit ermöglicht es, mehrere Ausgangssignale eines Markov-Modells in einem Fehlerbaum zu verwenden und stellt einen Mechanismus zur Verfügung, der logische Modellierungsfehler erkennt und diese bei der qualitativen und quantitativen Analyse berücksichtigt.

1 Einleitung

Bei sicherheitskritischen Systemen spielt die verteilte Entwicklung eine immer größere Rolle, weil die verschiedenen Systemkomponenten von unterschiedlichen Herstellern produziert werden. Dabei handelt es sich oft um standardisierte Komponenten, die in den unterschiedlichsten Systemen eingesetzt werden können. Zum einen ist es notwendig, eine Komponente in verschiedenen Systemen verwenden zu können, ohne jedes Mal erneut eine Sicherheitsanalyse anfertigen zu müssen. Zum anderen sollen gleichwertige Komponenten untereinander austauschbar sein, ohne die Sicherheit des Systems erneut analysieren zu müssen. Das trägt dazu bei, die Kosten bei der Entwicklung von neuen Produkten durch die Wiederverwendung bereits durchgeführter Analysen zu senken. Auch bei Produktlinien ist es entscheidend, den Analyseaufwand für die einzelnen Ausführungen zu minimieren.

Voraussetzung für die Wiederverwendung von Komponenten ist eine einfache Integration der Komponenten und den dazugehörigen Analysemodellen und Ergebnissen. Dabei ist es essentiell, dass die Anforderungen der Komponenten mit denen des restlichen Systems übereinstimmen. Damit eine Integration gelingen kann ist es notwendig, entsprechende Schnittstellen für die Komponenten zu definieren. Das ist

nicht nur beim Austausch von Komponenten in einem fertig analysierten System zweckdienlich. Denn bei der modularen Komposition über Schnittstellen kann der Fehlerbaum des Gesamtsystems bereits konstruiert werden, bevor die verwendeten Komponenten ausgewählt bzw. die Ergebnisse der Analysen vorliegen. Durch die Schnittstellenabstraktion kann zudem die zu verwendende Sicherheitsanalysetechnik unbestimmt bleiben und ermöglicht somit eine maximale Wiederverwendbarkeit in anderen Systemen.

Damit manche Techniken zu Eingang-Ausgang-Schnittstellen passen, müssen jedoch Restriktionen bestimmt werden, die oft auch die Modularität einschränken. Zum Beispiel wird bei der Markov-Analyse die Anzahl der ausgehenden Signale auf genau eines beschränkt, um die Unabhängigkeit der Ereignisse in der Fehlerbaumanalyse des Gesamtsystems zu gewährleisten. Diese Einschränkung führt dazu, dass der Systemdesigner für die Module auf der untersten Ebene der Fehlerbaumhierarchie entscheiden muss, ob Markov als Analysetechnik zugelassen werden sollen oder nicht. Sollen mehrere Ausgangssignale einer Komponente im Fehlerbaum verwendet werden, dann ist bei diesem Ansatz die Verwendung einer Markov-Analyse für diese Komponente nicht möglich. Steht jedoch fest, dass genau ein Ausgangssignal ausreichend für die Integration der Sicherheitsanalysen ist, dann kann die Beschränkung im Modell umgesetzt und somit die Verwendung von Markov als Analysetechnik erlaubt werden.

Könnte man auf die Beschränkung der Anzahl der Ausgangssignale von Markov-Modellen verzichten, dann käme das dem modularen Modellierungsansatz und der Wiederverwendbarkeit von Komponenten zu Gute. Die hier vorgestellte Arbeit setzt sich deshalb mit diesem Problem auseinander und zeigt, wie mehrere voneinander abhängige Signale aus einem Markov-Modell in einem Fehlerbaum verwendet werden können. Dabei werden logische Modellierungsfehler erkannt und in den Berechnungen der Wahrscheinlichkeiten berücksichtigt.

2 Hintergrund

Die Fehlerbaumanalyse ist wohl eine der bekanntesten und weitverbreitetsten Analysetechniken in Bezug auf die Sicherheit von Systemen [Ve81]. Standards wie z.B. ISO 26262 verlangen für sicherheitskritische Systeme eine deduktive Sicherheitsanalyse und erwähnen dabei explizit die Fehlerbaumanalyse. Im Laufe der Jahrzehnte wurde eine Vielzahl an Varianten und Weiterentwicklungen vorgestellt. Component Fault Trees (CFT) [Ka05] bieten beispielsweise eine Modularisierung über Schnittstellen und ermöglichen dadurch eine komponenten-orientierte Entwicklung und Komposition von Fehlerbäumen. Andere Ansätze erhöhen die Ausdruckskraft von Fehlerbäumen durch das Hinzufügen spezieller Gates, um z.B. zeitliche Abhängigkeiten modellieren zu können. Beispiele für solche Ansätze sind Dynamic Fault Trees (DFT) [DBB92] und Pandora [Wa09].

Im Vergleich zu Fehlerbäumen sind Markov-Modelle [IEC04] zustandsbasiert und verfügen über eine höhere Ausdruckskraft bei der Modellierung von Systemen. Sie

besitzen eine Vielzahl nützlicher mathematischer Eigenschaften [BP96]. Im Gegenzug leiden Markov-Modelle aber unter der Komplexität der quantitativen Analyse und mangelhafter Kompositionsfähigkeit, weil keine Eingangssignale im Markov-Modell verwendet werden können. Deshalb werden meist nur Komponenten, wie z.B. Sensoren, oder kleine Systeme mit Markov modelliert.

Die Integration von Markov-Modellen und Fehlerbäumen ist ein für die Industrie relevantes Thema und wurde daher schon mehrfach in der Literatur behandelt. So werden z.B. die Basic Events in Boolean logic Driven Markov Processes (BDMP) grundsätzlich als unabhängige Markov-Ketten mit zwei Zuständen modelliert [BB03]. Die Verbindung mit Fehlerbäumen erfolgt üblicherweise in Form einzelner Basic Events, die mit Markov modelliert und in den Fehlerbaum aufgenommen werden (siehe z.B. [Zi11]). Diese Form der Integration ist in vielen Softwaretools verfügbar, z.B. der Reliability Workbench von Isograph[1], der KB3 Workbench[2], Item Toolkit[3] und ESSaRel[4]. Alle Tools haben gemein, dass aus einem Markov-Modell nur die Zustandswahrscheinlichkeit von einem Zustand bzw. einer Gruppe von Zuständen im Fehlerbaum verwendet werden kann. Diese Einschränkung verhindert, dass voneinander abhängige Ereignisse im Fehlerbaum verwendet werden.

2.1 Unabhängigkeit von Ereignissen

Eine Möglichkeit zur Berechnung der Wahrscheinlichkeit des Top-Level Events ist es, die Wahrscheinlichkeit für jedes Gatter im Fehlerbaum anhand der Eingänge zu bestimmen. Dazu wird bei den Gattern direkt oberhalb der Basic Events begonnen und stufenweise vorgegangen, bis schließlich das Top-Level Event erreicht wird. Ein Nachteil dieser Methode ist, dass für die einzelnen Gatter nur die Wahrscheinlichkeiten bekannt sind. Information über deren Eingänge und eventuelle Abhängigkeiten zwischen diesen, oder innerhalb von Teilbäumen, gehen verloren. Das ist ein Problem, denn in der Fehlerbaumanalyse wird grundsätzlich davon ausgegangen, dass alle Basic Events voneinander unabhängig sind. Ist dies nicht der Fall, dann kann es zu Fehlern in der qualitativen und quantitativen Analyse kommen. Markov-Modelle sind zustandsbasiert und befinden sich daher zu jedem Zeitpunkt in genau einem Zustand. Ein einzelner Zustand oder eine Gruppe von Zuständen kann mit einem Ausgangssignal verbunden werden. Dieses Ausgangssignal kann nun z.B. als Basic Event oder Eingang für andere Systemkomponenten verwendet werden. Dazu wird berechnet wie wahrscheinlich es ist, dass das Markov-Modell nach einer bestimmten Zeit (z.B. nach Ablauf der Missionszeit) in einem der entsprechenden Zustände ist.

Das Problem bei der Verwendung mehrerer Signale aus dem gleichen Markov-Modell ist, dass die verknüpften Wahrscheinlichkeiten nicht unabhängig sind. Abbildung 1 veranschaulicht dieses Problem anhand eines Widerspruchs. Werden die beiden Ausgangssignale des Markov-Modells im Fehlerbaum an einem UND-Gatter

[1] http://www.isograph.com/
[2] http://researchers.edf.com/software/kb3-44337.html
[3] http://www.itemsoft.com/item_toolkit.html
[4] http://www.essarel.de/

zusammengeführt, dann kann dieses Gatter niemals erfüllt sein. Obwohl rechnerisch aufgrund der zugewiesenen Wahrscheinlichkeiten ein Wert ermittelt werden kann, ist es unmöglich, dass sich das Markov-Modell zur gleichen Zeit in zwei unterschiedlichen Zuständen befindet.

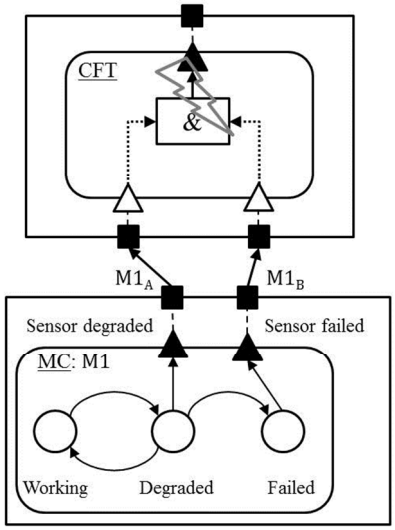

Abbildung 1: Logischer Modellierungsfehler am UND-Gatter

Im Gegensatz dazu stellt die Verwendung mehrerer Signale aus einem Markov-Modell bei einem ODER-Gattern keinen Modellierungsfehler dar, abgesehen von der Nichtbeachtung der Unabhängigkeit von Basic Events in der Fehlerbaumanalyse. Die Abhängigkeit führt zu einer falschen Berechnung der Eintrittswahrscheinlichkeit des ODER-Gatters. Bei zwei Eingängen $M1_A$ und $M1_B$ gilt die Formel $P(ODER) = P(M1_A) + P(M1_B) - P(M1_A) * P(M1_B)$. Für zwei sich gegenseitig ausschließende Ereignisse ist die korrekte Formel jedoch $P(ODER) = P(M1_A) + P(M1_B)$.

Auch wenn die Differenz der beiden Formeln für kleine Werte von $P(M1_A)$ und $P(M1_B)$ gering ist [An01], entspricht dies doch einer optimistischen Approximation. Um eine Unterschätzung der Fehlerwahrscheinlichkeit zu verhindern, dürfen Schnittmengenprodukte abhängiger Ereignisse deshalb nicht von deren Summe abgezogen werden.

Damit die hier aufgezeigten Probleme erkannt werden können ist es deshalb notwendig, die Quellen von Wahrscheinlichkeiten im Fehlerbaum zu propagieren. Es ist nicht ausreichend, die Überprüfung der Abhängigkeit von Basic Events auf die untersten Gatter des Fehlerbaums zu beschränken. Im Folgenden wird eine Methode vorgestellt, die Informationen über die Herkunft von Basic Events im Fehlerbaum propagiert und dadurch Modellierungsfehler an UND-Gattern erkennen bzw. Berechnungsfehler an ODER-Gattern vermeiden kann.

3 Methode

Jedem im System verwendeten Markov-Modell muss ein eindeutiger Name zugewiesen werden, z.B. $M1$ dem ersten, $M2$ dem zweiten usw. Diese Namen werden verwendet, um den Einfluss voneinander abhängiger Ausgangssignale (kurz Signale) von Markov-Modellen im Fehlerbaum verfolgen zu können. Diese Signale werden als Basic Events behandelt und erhalten ebenfalls eindeutige Namen, z.B. $M1_A$, $M1_B$ usw. Zur Vereinfachung werden nur ODER- und UND-Gatter betrachtet. Alle anderen nicht-dynamischen Gatter können durch diese beiden Gatter dargestellt werden. Um die Veranschaulichung des Algorithmus zu vereinfachen wird ferner angenommen, dass Signale aus Markov-Modellen nicht mehrfach als Basic Event verwendet werden. Mit einer zusätzlichen Unterscheidung der Signale in den Optionstabellen ist es jedoch möglich, diese Voraussetzung zu verwerfen.

3.1 Optionstabellen

Basic Events und Gattern werden Optionstabellen zugeordnet, die aus Schlüssel-Wert Paaren bestehen. In ihnen werden konditionelle Ausfallwahrscheinlichkeiten gespeichert. Jede Optionstabelle (kurz Tabelle) verfügt über einen Wert für den Standardschlüssel nil, der die Signale von Markov-Modellen nicht einschränkt. Abhängig von der Verwendung von Signalen im entsprechenden Teil des Fehlerbaums kann eine Tabelle noch weitere Schlüssel enthalten. Bei der Berechnung der zugehörigen Werte wird den Signalen aus den im Schlüssel enthaltenen Markov-Modellen eine Null zugewiesen. Anhand der berechneten Tabelleneinträge kann z.B. bei einem UND-Gatter festgestellt werden, ob ein Eingang ohne den Einfluss eines bestimmten Markov-Modells überhaupt auftreten kann. Das Erstellen der Tabellen wird bei den Basic Events begonnen und bis zum Top Event weitergeführt. Für die Bearbeitung eines Gatters müssen die Tabellen aller Eingänge des Gatters vorliegen. In Bezug auf Basic Events und Gatter bedeutet die Aussage „von Markov-Modell M abhängig“, dass es in der jeweiligen Tabelle einen Eintrag für M gibt.

3.2 Basic Events

Ein Basic Event E, das kein Signal aus einem Markov-Modell ist, bekommt als einzigen Eintrag in die Optionstabelle das Paar $(nil, P(E))$. Basic Events, die aus einem Markov-Modell stammen, bekommen zusätzlich einen Tabelleneintrag mit dem Namen des Markov-Modells und einer Null, denn das Basic Event kann nicht eintreten, wenn der Einfluss des Markov-Modells ignoriert wird.

3.3 ODER-Gatter

Zur Berechnung eines ODER-Gatters werden die Tabellen aller Eingänge benötigt. Zunächst wird die Tabelle des ODER-Gatters mit dem Standardwert initialisiert (Phase 1). Danach werden alle Markov-Modelle ermittelt, von denen das Gatter abhängig ist (Phase 2). Aus der Liste der Markov-Modelle werden schließlich die restlichen Schlüssel

generiert und die zugehörigen Wahrscheinlichkeiten berechnet (Phase 3). Die drei Phasen werden nun im Einzelnen erläutert.

Phase 1: Initiierung der Optionstabelle

Die Tabelle eines ODER-Gatters $ODER$ enthält immer den Eintrag $(nil, P(ODER))$, der die normal berechnete Wahrscheinlichkeit darstellt. Das heißt, dass alle Signale der Markov-Modelle uneingeschränkt auftreten können und somit in die Berechnung der Wahrscheinlichkeit einfließen. Zur Berechnung werden die Einträge mit dem Schlüssel nil der jeweiligen Tabellen genommen. Falls zwei Eingänge E_i und E_j vom selben Markov-Modell abhängig sind, dann gilt $P(E_i, E_j) = 0$, denn beide Ereignisse schließen sich gegenseitig aus. Für ein ODER-Gatter mit n Eingängen lautet die Formel [LB11]:

$$P(ODER) = \sum_{i=1}^{n} P(E_i) - \sum_{1 \le i < j \le n} P(E_i, E_j) + \cdots + (-1)^{k-1} \sum_{1 \le i1 < \cdots < ik \le n} P(E_{i1}, \ldots, E_{ik}) + \cdots + (-1)^{n-1} P(E_1, \ldots, E_n) \quad (1)$$

Abbildung 2 zeigt links als Beispiel ein ODER-Gatter mit drei Eingängen. Setzt man die Wahrscheinlichkeiten (Variable r) der Eingänge in (1) ein, dann erhält man $P(ODER) = P(A) + P(M1_A) + P(M1_B) - P(A, M1_A) - P(A, M1_B) - P(M1_A, M1_B) + P(A, M1_A, M1_B) = 0{,}039502$. Berücksichtigt man jedoch die Abhängigkeit der beiden Basic Events $M1_A$und $M1_B$, dann hat der Schlüssel nil den Wert $P(ODER) = P(A) + P(M1_A) + P(M1_B) - P(A, M1_A) - P(A, M1_B) = 0{,}0397$.

Phase 2: Ermittlung der relevanten Markov-Modelle

Für die weiteren Einträge in der Tabelle ist es notwendig zu wissen, welche Markov-Modelle bei der Berechnung des Gatters beachtet werden müssen. Dazu wird aus den Tabellen aller Eingänge eine Liste von eindeutigen Schlüsseln generiert, die genau einen Namen eines Markov-Modells enthalten. Falls es keine Signale aus Markov-Modellen unterhalb des Gatters gibt, dann bleibt die Namensliste leer und es werden keine weiteren Einträge zur Tabelle hinzugefügt.

Phase 3: Füllen der Optionstabelle

Aus der Liste der relevanten Markov-Modelle werden nun die weiteren Schlüssel für die Einträge der Tabelle generiert. Es ist auch notwendig Kombinationen von Markov-Modellen zu betrachten, da es vorkommen kann, dass bei der Berechnung des nächst höheren Gatters gleich mehrere Markov-Modelle ignoriert werden müssen. Dazu wird jedem Namen aus der Liste eine binäre Variable zugeordnet. Anschließend kann unter Verwendung einer Binärtabelle jede mögliche Kombination von Namen abgeleitet und als Schlüssel in die Tabelle eingetragen werden. Bei der Berechnung des Wertes für einen Schlüssel S wird von jedem Eingang E_i der Wert $P_S(E_i)$ genommen. Weil die Schlüssel des ODER-Gatters Kombinationen aus allen enthaltenen Markov-Modellen darstellen kann es vorkommen, dass sich für Eingänge des Gatters kein Wert für S zuordnen lässt. In diesem Fall wird der längste Teilschlüssel von S aus der Tabelle gewählt. Sollte es überhaupt keine Übereinstimmung für einen Eingang E_i geben, dann wird der Standardwert $P_{nil}(E_i)$ verwendet.

Beispiel

Abbildung 2 zeigt auf der rechten Seite das Gatter $ODER2$ mit drei Eingängen. Das Gatter $ODER$ (links) ist, wie aus der Tabelle ersichtlich, abhängig von $M1$. $M2_A$ ist ein Signal aus einem anderen Markov-Modell $M2$ und hat deshalb die Schlüssel nil und $M2$. B ist ein gewöhnliches Basic Event mit Standardschlüssel *nil*.

In Phase 1 wird die Optionstabelle initialisiert. Da es keine zwei Eingänge gibt, die vom selben Markov-Modell abhängig sind, kann der Wert für nil mit der Standardformel (1) und den Werten $P_{nil}(ODER) = 0{,}0397$, $P_{nil}(M2_A) = 0{,}02$ und $P_{nil}(B) = 0{,}01$ berechnet werden. In Phase 2 werden nun die relevanten Markov-Modelle ermittelt. Laut den Tabellen der Eingänge lautet die Liste $\{M1, M2\}$. In Phase 3 werden schließlich die Schlüssel generiert und die Tabelle mit Werten gefüllt. Aus den binären Variablen b_{M1} und b_{M2} werden die Kombinationen $\{M1\}$, $\{M2\}$ und $\{M1, M2\}$ hergeleitet. Der Wert $P_{M1}(ODER2)$ wird mit den Werten $P_{M1}(ODER) = 0{,}01$, $P_{M1}(M2_A) = P_{nil}(M2_A) = 0{,}02$ und $P_{M1}(B) = P_{nil}(B) = 0{,}01$ berechnet. Analog dazu $P_{M2}(ODER2)$ mit $P_{M2}(ODER) = P_{nil}(ODER) = 0{,}0397$, $P_{M2}(M2_A) = 0$ und $P_{M2}(B) = P_{nil}(B) = 0{,}01$, und $P_{M1,M2}(ODER2)$ mit $P_{M1,M2}(ODER) = P_{M1}(ODER) = 0{,}01$, $P_{M1,M2}(M2_A) = P_{M2}(M2_A) = 0$ und $P_{M1,M2}(B) = P_{nil}(B) = 0{,}01$. Die Werte in Abb. 2 sind Rundungen.

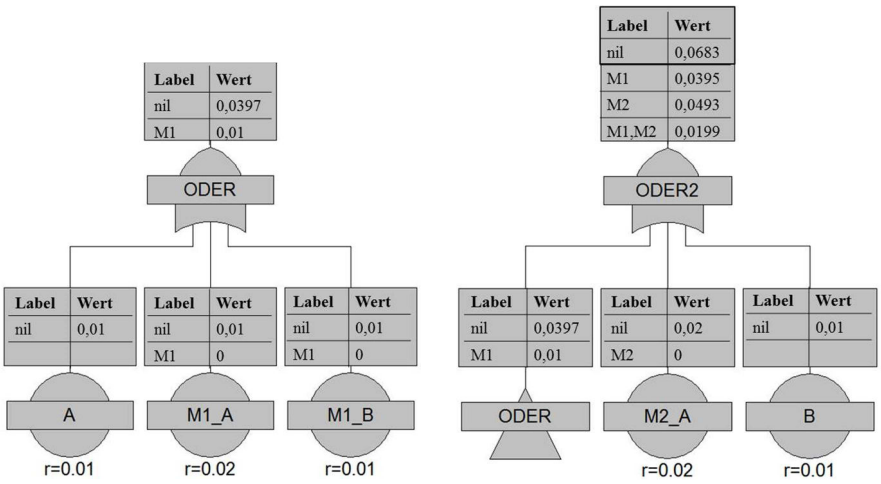

Abbildung 2: Zwei ODER-Gatter mit je drei Eingängen

3.4 UND-Gatter

Analog zum ODER-Gatter wird die Optionstabelle zunächst initiiert (Phase 1) und dann mit den ermittelten Markov-Modellen (Phase 2) der Rest der Tabelle berechnet (Phase3).

Phase 1: Initiierung der Optionstabelle

Die Optionstabelle eines UND-Gatters enthält, wie auch bei einem ODER-Gatter, einen Eintrag für den Schlüssel nil. Allerdings führen mehrere Signale aus dem gleichen

Markov-Modell beim UND-Gatter, wie bereits gezeigt, zu einem logischen Widerspruch. Der Wert für den Schlüssel nil repräsentiert deshalb nur die durch konventionelle Fehlerbaumberechnung ermittelte Eintrittswahrscheinlichkeit, wenn keine zwei Eingänge des Gatters vom gleichen Markov-Modell abhängig sind. Die Berechnung des Wertes für den Schlüssel nil erfolgt nach den folgenden Schritten:

1. Die Tabellen aller Eingänge werden auf Nullen durchsucht und die entsprechenden Schlüssel in einer Liste L gespeichert. Enthält diese Liste Dubletten oder den Schlüssel nil, dann ist der Wert des UND-Gatters null. Dies lässt auf einen logischen Fehler in der Modellierung schließen, denn das Gatter kann nicht beim Eintritt des Top-Level Events mitwirken. Die Situation sollte vom Systemintegrator eingehend untersucht werden.
2. Für jeden Eingang E_i des UND-Gatters wird eine Kopie der Liste L aus Schritt 1 erstellt und alle Schlüssel S entfernt, für die $P_S(E_i) = 0$ gilt. Aus der Liste wird der längst mögliche Schlüssel S_i gebildet, für den es in der Tabelle von E_i einen Eintrag gibt. Gibt es keinen solchen Schlüssel, dann gilt $S_i = nil$.
3. Berechne den Wert des UND-Gatters unter Verwendung der entsprechenden Werte für S_i für alle Eingänge E_i.

Abbildung 3 zeigt als Beispiel links das Gatter UND mit drei Eingängen: Das Gatter $ODER$ (siehe Abb. 2), ein Basic Event aus $M1$, und ein reguläres Basic Event. Die Tabellen der Eingänge enthalten nur für Schlüssel $M1$ von Eingang $M1_C$ eine Null. Deshalb wird für die Berechnung von $P_{nil}(UND)$ der Wert $P_{nil}(M1_C) = 0{,}02$ verwendet. Weil dieser Wert von $M1$ abhängt müssen die anderen Eingänge des Gatters von $M1$ unabhängig bleiben, was durch die Verwendung der Werte $P_{M1}(ODER) = 0{,}01$ und $P_{M1}(B) = P_{nil}(B) = 0{,}01$ gewährleistet wird. Nach der Berechnungsformel für UND-Gatter, $P(UND) = \prod_{i=1}^{n} P(E_i)$, gilt $P_{nil}(UND) = 2{,}0 * 10^{-6}$.

Phase 2: Ermittlung der relevanten Markov-Modelle

Diese Phase ist analog zum ODER-Gatter.

Phase 3: Füllen der Optionstabelle

Aus der Liste der relevanten Markov-Modelle werden, wie beim ODER-Gatter beschrieben, die weiteren Schlüssel für die Einträge der Optionstabelle generiert. Enthält die Tabelle eines Eingangs für den Schlüssel S oder einen Teil des Schlüssels S eine Null, dann ist $P_S(UND) = 0$. Die restlichen Schlüssel werden berechnet, indem für jeden Eingang aus der entsprechenden Tabelle ebenfalls der Wert für den Schlüssel S bzw. den längsten vorhandenen Teilschlüssel von S genommen wird. Gibt es für einen Eingang E_i keinen solchen Teilschlüssel, dann wird der Standardwert $P_{nil}(E_i)$ verwendet.

Beispiele

Das Gatter $UND2$ auf der rechten Seite von Abb. 3 ist abhängig von $M1$ und $M2$. Daraus ergeben sich die Schlüssel nil, $M1$, $M2$ und $\{M1, M2\}$ für die Tabelle von $UND2$. Da die Eingänge keine gemeinsamen Abhängigkeiten haben gilt $P_{nil}(UND2) = P_{nil}(ODER) * P_{nil}(M2_A) * P_{nil}(B) = 7{,}94 * 10^{-6}$. Eingang $M2_A$ hat für den Schlüssel $M2$ eine Null,

deshalb gilt $P_{M2}(UND2) = 0$ und $P_{M1,M2}(UND2) = 0$. Für Schlüssel $M1$ gilt $P_{M1}(UND2) = P_{M1}(ODER) * P_{nil}(M2_A) * P_{nil}(B) = 2{,}0 * 10^{-6}$.

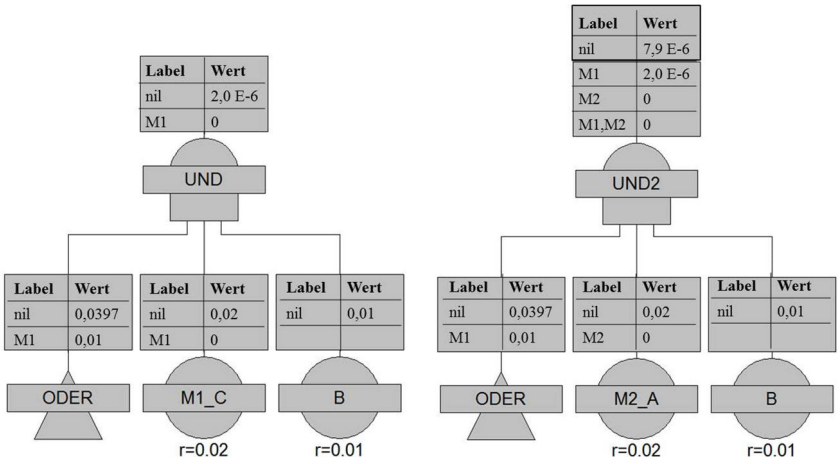

Abbildung 3: Zwei UND-Gatter mit je drei Eingängen

Abbildung 4 zeigt rechts das Gatter UND mit zwei Eingängen aus Markov-Modellen und dem Gatter $ODER$ (links) als weiteren Eingang. Wegen der beiden Eingangssignale $M1_B$ und $M2_B$ darf $ODER$ nicht von $M1$ und $M2$ abhängig sein, da sonst das Gatter UND nicht erfüllt werden kann. Aus $P_{M1,M2}(ODER) = 0$ folgt aber $P_{nil}(UND) = 0$. In diesem vereinfachten Beispiel ist das Ergebnis direkt nachvollziehbar, denn keines der Markov-Modelle kann sich in Zustand B befinden (Eingänge $M1_B$ und $M2_B$), wenn es sich gleichzeitig in Zustand A befinden muss (Eingang $ODER$). Das Gatter UND kann niemals erfüllt werden und der Systemintegrator muss in diesem Fall die Logik und die Struktur des Fehlerbaums eingehend überprüfen und entsprechend berichtigen.

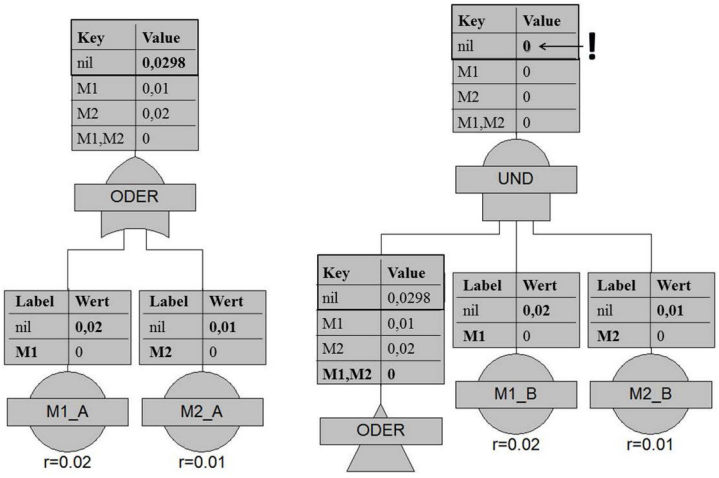

Abbildung 4: Erkennung eines Modellierungsfehlers

4 Zusammenfassung und Ausblick

Die vorgestellte Arbeit befasst sich mit der Integration modularer Sicherheitsanalysen und untersucht im Speziellen die Möglichkeit, mehrere Ausgangssignale aus einem Markov-Modell in einem Fehlerbaum verwenden zu können. Mit Hilfe von Optionstabellen, in denen bedingte Wahrscheinlichkeiten und Abhängigkeiten für Basic Events und Gatter des Fehlerbaums gespeichert werden, kann der Einfluss der Signale aus Markov-Modellen von den Basic Events bis hin zum Top-Level Event verfolgt werden. Mit den gewonnenen Informationen werden zum einen Berechnungsfehler an ODER-Gattern vermieden und zum anderen logische Modellierungsfehler durch zwei sich gegenseitig ausschließende Basic Events unterhalb eines UND-Gatters erkannt. Die erkannten Fehler können dann durch den Systemintegrator gezielt berichtigt werden.

Als weiterführende Arbeit wurde die mehrfache Verwendung von Ausgangssignalen im Fehlerbaum bereits methodisch erarbeitet. Als nächster Schritt ist eine Toolintegration in Enterprise Architect geplant. Des Weiteren soll der Ansatz auf dynamische Fehlerbäume ausgeweitet werden. Außerdem ist eine Verfeinerung der Abhängigkeiten zwischen den Basic Events angedacht. So könnten mit einem modifizierten Ansatz die unterschiedlichsten kausalen und temporalen Abhängigkeiten zwischen Basic Events im Fehlerbaum verfolgt und berücksichtigt werden.

Literaturverzeichnis

[An01] Apthorpe, R.: A probabilistic approach to estimating computer system reliability. In Proceedings of the Fifteenth Systems Administration Conference (LISA XV)(USENIX Association: Berkeley, CA), p. 31, 2001.

[BB03] Bouissou, M.; Bon, J.-L.: A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes, Reliability Engineering & System Safety, Volume 82, Issue 2, November 2003, Pages 149-163, ISSN 0951-8320, http://dx.doi.org/10.1016/S0951-8320(03)00143-1.

[BP96] Barlow, R.E.; Proschan, F.: Mathematical theory of reliability. SIAM. Classics in applied mathematics; 1996.

[DBB92] Dugan, J.B.; Bavuso, S.J.; Boyd, M.A.: Dynamic Fault Tree Models for Fault Tolerant Computer Systems. In: IEEE Transactions on Reliability 41 (1992), September, n. 3, p. 363–377.

[IEC04] IEC 61165, Ed.2: Application of Markov techniques. International Electrotechnical Commission, IEC, 2004.

[Ka05] Kaiser, B.: State/event Fault Trees: A safety and Reliability analysis Technique for Sotware-Controlled Systems, PHD Thesis, Technische Universitaet Kaiserslautern, Fachbereich Informatik, 2005.

[LB11] Lin, Z.; Bai, Z.: Probability Inequalities. Springer Berlin Heidelberg, p. 1, 2011.

[Ve81] Veseley, W.E.: Fault Tree Handbook. Division of the System Safety Office of Nuclear Reactor Regulation, US Nuclear Regulatory Commission, Washington DC, 1981

[Wa09] Walker, M.: Pandora – A Logic for the Qualitative Analysis of Temporal Fault Trees. PhD Thesis, University of Hull, UK, 2009.

[Zi11] Zixian, L.; Xin, N.; Yiliu, L.; Qinglu, S.; Yukun, W.: Gastric esophageal surgery risk analysis with a fault tree and Markov integrated model. In Reliability Engineering & System Safety 96 (12), pp. 1591–1600, 2011.