Eine Quantitative Untersuchung des European Train Control System mit UML State Machines

Armin Zimmermann und Jan Trowitzsch
Fachgebiet Prozessdatenverarbeitung und Robotik
Institut für Technische Informatik und Mikroelektronik
Technische Universität Berlin
Einsteinufer 17
D-10587 Berlin
Tel.: +49 (0)30 314-73112
Fax: +49 (0)31 314-21116
E-Mail: {azi | joni}@cs.tu-berlin.de

Abstract: In diesem Beitrag wird eine Verhaltensbeschreibung technischer Systeme mit State Machines aus der *Unified Modeling Language* vorgeschlagen, wobei Erweiterungen des *Profile for Schedulability, Performance, and Time Specification* für quantitative Aspekte benutzt werden. Ein so entstehendes Modell wird automatisch in ein stochastisches Petri-Netz umgewandelt, dessen Leistungsmaße dann mit Simulation oder numerischer Analyse bestimmt werden. Als Anwendungsbeispiel dient ein Ausschnitt des zukünftigen Europäischen Eisenbahnleit- und Sicherungssystems ETCS, für das der Zusammenhang zwischen Kommunikationsqualität und minimalem Zugfolgeabstand untersucht wird.

Stichworte: Petri-Netze, UML State Machines, Modelltransformation, Zuverlässigkeitsanalyse von Verkehrssystemen, ETCS

1 Einleitung

Der Entwurf komplexer technischer Systeme erfordert Modellierungs- und Bewertungsverfahren, die das spätere Verhalten frühzeitig bestimmen und damit helfen, geeignete Varianten auszuwählen. Eine Bewertung von Leistungsfähigkeit und Zuverlässigkeit ist mit quantitativen Beschreibungen und Analyseverfahren möglich. Für die industrielle Anwendbarkeit müssen eingeführte Modellierungssprachen verwendet werden, um die zahlreichen im universitären Umfeld existierenden Verfahren besser nutzbar zu machen.

Die Modellklassen der *Unified Modeling Language* sind für industrielle Anwendungen akzeptiert, erlauben jedoch ohne Erweiterungen keine Modellierung und Untersuchung von Eigen-

---

Erscheint in:
Entwurf komplexer Automatisierungssysteme (EKA 2006), Mai 2006, Braunschweig

schaften wie Rechtzeitigkeit, Durchsatz oder Fehlertoleranz. In diesem Beitrag wird eine Verhaltensbeschreibung technischer Systeme mit State Machines aus der *Unified Modeling Language* mit Erweiterungen des *Profile for Schedulability, Performance, and Time Specification* für quantitative Aspekte vorgeschlagen. Ein so entstehendes Modell kann automatisch in ein stochastisches Petri-Netz umgewandelt und dessen Leistungsmaße dann mit Simulation oder numerischer Analyse bestimmt werden. Auf diese Weise können die für diese Netzklasse existierenden Ergebnisse und Softwarewerkzeuge genutzt werden. Die in Vorarbeiten [TrZi2005, TrZiHo2005] vorgeschlagenen Übersetzungsregeln werden im vorliegenden Beitrag erweitert.

Als Anwendungsbeispiel wird ein Teilbereich des zukünftigen *European Train Control System* (ETCS) betrachtet. In dessen Anwendungsvariante Level 3 werden dynamische Streckenabschnitte über Funkkommunikation freigegeben (*moving block operation*). Einzelne Parameter der Zuverlässigkeit und Rechtzeitigkeit sind in den entsprechenden Spezifikationen angegeben. Darüber hinaus ist aber eine zusammenhängende Untersuchung des Verhaltens mit einem stochastischen Modell zur Bestimmung der daraus folgenden Leistungsparameter notwendig. Dieser Beitrag stellt einen Schritt in dieser Richtung dar.

In vorangegangenen Arbeiten [ZimHom2003, TrZiHo2005] wurde ein Fehlermodell für das eingesetzte Kommunikationssystem GSM-R entwickelt und untersucht. Ein Modell der Kommunikation inklusive Fehlern wurde dabei direkt als Petri-Netz erstellt [ZimHom2003, ZimHom2005]. Der vorliegende Beitrag stellt auch die Kommunikation zwischen Zügen und Funkzugbeeinflussungsstelle mit State Machines dar. Eine quantitative Untersuchung des entstehenden Petri-Netzes zeigt den Zusammenhang zwischen Zugabstand und Betriebssicherheit, sowie den Einfluss der Zuverlässigkeit der Kommunikationsverbindung. Dazu werden Methoden zur beschleunigten Simulation seltener Ereignisse eingesetzt. Die Ergebnisse zeigen, dass ein wirtschaftlicher Fahrbetrieb mit der in den Spezifikationen geforderten Zuverlässigkeit von GSM-R kaum erreichbar ist.

Der Großteil der Arbeiten im Bereich Leit- und Sicherungssysteme für Eisenbahnen beschäftigt sich mit qualitativen bzw. funktionalen Aspekten wie dem Nichterreichen sicherheitskritischer Zustände. Für die korrekte und wirtschaftliche Funktion eines komplexen kommunikationsbasierten und verteilten Echtzeitsystems wie ETCS müssen auch quantitative Eigenschaften untersucht werden. Zufällige Einflüsse und Fristen erfordern stochastische nicht-Markovsche Modelle.

Die Kommunikationsstruktur von ETCS wurde in [JaMeSch1998, MeSch1999] mit farbigen Petri-Netzen modelliert. Das Modell wird für eine Einteilung in Module, eine Visualisierung des Verhaltens und die Überprüfung verschiedener Szenarien eingesetzt.

Ein funkbasiertes Signalisierungssystem wird in [DaKl2001] im Kontext eines beschrankten Bahnübergangs untersucht. Dazu werden *live sequence charts* eingesetzt und mit dem Softwarewerkzeug StateMate [Har1998] analysiert. Eine ETCS-Funkzugbeeinflussungsstelle wird in [ChiCiPo1999] mit *message sequence charts* formal modelliert, und dafür verschiedene Szenarien validiert. Eine Simulationsstudie der Deutschen Bahn [Osb2002] vergleicht den derzeitigen

Fahrbetrieb mit dem durch ETCS erreichbaren. Dazu wird ein spezielles Simulationswerkzeug für den Fahrbetrieb eingesetzt, dass einzelne Zugbewegungen auf einer angenommenen Beispielstrecke abbildet und untersucht. Es wird geschlussfolgert, dass die Streckenauslastung für das Beispiel mit ETCS im Vergleich zu heute um ungefähr 30 Prozent zu steigern ist. Dabei werden aber die in diesem Beitrag untersuchte Funkdatenübertragung und die darin auftretenden Kommunikationsfehler nicht betrachtet.

Für die quantitative Analyse erweiterter UML-Diagramme existieren verschiedene Ansätze in der Literatur, die ihren Ursprung meist im Gebiet der Software-Leistungsbewertung haben. *Generalized Stochastic Petri Nets* (GSPNs, [AjBaCo1995]) werden in den Arbeiten [KiPo1999, KiPo2000, PoKi1999] eingesetzt, um das Verhalten von StateCharts abzubilden. Dabei entsteht aus jedem Zustand eine Stelle, und aus jedem Zustandsübergang eine Transition des Petri-Netzes. Die entstehenden Teilmodelle werden mit Hilfe von UML *collaboration diagrams* zusammengefügt. Ein anderer Ansatz für die systematische Erzeugung von GSPNs wurde in den Arbeiten [Mer2004, BeDoMe2002] vorgeschlagen. Erweiterte UML-Diagramme werden dabei in annotierte GSPN-Module übersetzt, um dann in ein Gesamtmodell integriert zu werden. Die Erweiterung von UML-Modellen um probabilistische Auswahl und stochastische Zeiten wird auch in [HoSmKi2002] vorgeschlagen. Allen genannten Arbeiten ist gemeinsam, dass ausschließlich exponentiell verteilte Zeiten erlaubt sind.

Ein anderer Ansatz zur quantitativen Analyse erweiterter UML-Modelle ist die direkte Untersuchung des Modells ohne vorherige Transformation in eine andere Modellklasse. Dieser Weg wird in [LiThüKl2002] verfolgt, wobei deterministische und exponentielle Aktivitätsdauern erlaubt sind. Der resultierende stochastische Prozess ist ein *generalized semi-Markov process* (GSMP), der unter bestimmten strukturellen Einschränkungen noch numerisch analysierbar ist.

Eine Erweiterung von StateCharts um zeitdiskrete stochastische Verteilungen erlaubt ein nachfolgendes *probabilistic model checking* [JaHe2004] mit *Markov decision processes*. Der genannte Beitrag wendet diese Technik (StoCharts) auf das in einer Vorarbeit [ZimHom2003] entwickelte Modell der Zuverlässigkeit des ETCS-Funkkanals an.

Die weitere Arbeit ist wie folgt gegliedert. Im anschließenden Kapitel wird die Erweiterung von UML State Machines um quantitative Informationen mit dem *Profile for Schedulability, Performance, and Time* beschrieben. Kapitel 3 beschreibt dann, wie derartige Modelle automatisch in ein stochastisches Petri-Netz übersetzt werden können. Das darauf folgende Kapitel führt das Anwendungsbeispiel ein. Ein UML State Machine Modell der ETCS-Zugkommunikation wird vorgestellt und in ein Petri-Netz umgewandelt. Anschließend wird der Zusammenhang zwischen Zugabstand und durch Kommunikationsfehler erzwungenen Nothalts berechnet. Ein im Softwarewerkzeug TimeNET [ZiFrGe2000] implementiertes Simulationsverfahren für seltene Ereignisse (RESTART, [ViViGa1994]) liefert quantitative Ergebnisse.

2 State Machines und das Profile for Schedulability, Performance, and Time

Die *Unified Modeling Language* (UML, [OMG2003]) ist eine Sammlung semi-formaler Modelle, mit denen unter anderem Struktur sowie Verhalten von Software und technischen Systemen beschrieben werden können. Erweiterungen zur Abbildung von Zusatzinformationen wie Zeiten und Wahrscheinlichkeiten schlägt das *UML Profile for Schedulability, Performance, and Time* (SPT Profile, [OMG2002]) vor. Für die quantitative Modellierung und Bewertung eignen sich Verhaltensmodelle, und dabei besonders State Machines. *Sequence charts* und *collaboration diagrams* beschreiben einzelne Abläufe eines Systems, und sind daher nicht direkt für eine Untersuchung des gesamten Verhaltens nutzbar. UML State Machines ähneln StateCharts [Har1998], und sind u.a. durch die Verfügbarkeit entsprechender Softwarewerkzeuge industriell akzeptiert.

Auf eine Beschreibung von UML State Machines sowie des SPT Profiles muss hier aus Platzgründen verzichtet werden. Im folgenden wird lediglich kurz beschrieben, wie die für eine quantitative Bewertung nötigen Informationen in einer State Machine mit den Erweiterungen des SPT Profile abgebildet werden können. Für genauere Informationen wird auf die bereits zitierten Vorarbeiten verwiesen.

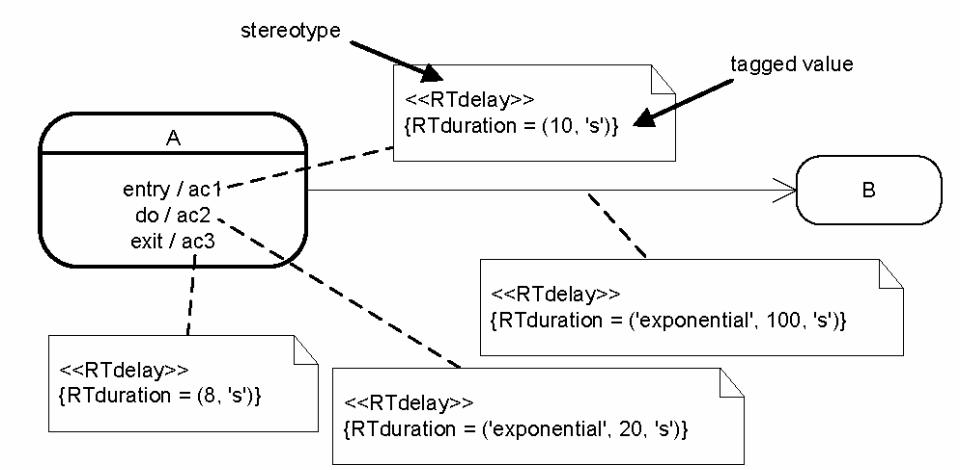

Abbildung 1: Beispiel einer UML State Machine mit SPT-Annotationen.

Abb. 1 zeigt eine einfache State Machine, die um Annotationen aus dem SPT Profile erweitert ist. Die wichtigsten Elemente einer State Machine sind Zustände (A und B im Bild) sowie Transitionen. Letztere modellieren Zustandsübergänge, und werden durch Pfeile zwischen den Zuständen dargestellt. Eine einfache State Machine befindet sich immer in höchstens einem Zustand, dem einzelne Aktivitäten zugeordnet sein können. In Abb. 1 hat Zustand A eine Aktivität ac1, die beim Eintritt in den Zustand ausgeführt wird (*entry*), sowie Aktionen ac2 und ac3, die während des Verweilens im Zustand (*do*) sowie beim Verlassen (*exit*) ausgeführt werden.

Transitionen markieren einen Zustandsübergang und können von Bedingungen abhängen sowie Ereignisse erzeugen. Komplexere Zustände (*composite states*) sind in Regionen unterteilt, um nebenläufige Vorgänge abzubilden. Jede einzelne Region beschreibt dann einen eigenen Ablauf,

der einen lokalen Zustand hat. Damit ist eine sehr viel einfachere Beschreibung paralleler und synchronisierter Vorgänge als mit einem klassischen Automatenmodell möglich.

Verschiedene Pseudozustände (*pseudostates*) erweitern die Ausdrucksmöglichkeiten. Sie entsprechen Zuständen, die sofort wieder verlassen werden, um damit z.B. Verzweigungen, Vereinigungen oder Entscheidungen in Zustandsübergängen zu modellieren.

Im SPT Profile werden so genannte *stereotypes* und *tagged values* definiert, mit denen UML-Modelle um quantitative Informationen erweitert werden können. Abb. 1 enthält beispielhaft einige solche Annotationen. Mit dem *stereotype* RTdelay kann die Dauer eines Vorgangs spezifiziert werden, für ac1 im Bild beispielsweise 10 Sekunden. *Tagged values* bestehen aus einem *property name* und einem zugeordneten Wert, wie z.B. {RTduration=(8,'s')}. Neben der Angabe einer festen Zeit ist auch die Spezifikation einer Verteilungsfunktion für eine stochastische Dauer möglich, wie im Bild für ac2 zu sehen. Der Zahlenwert gibt dann den Erwartungswert der Dauer als bestimmenden Parameter der Exponentialverteilung an. Für die Spezifikation von Wahrscheinlichkeiten von Zustandsübergängen wird die Verwendung des *Stereotypes* PAstep mit dem *tagged value* PAprob vorgeschlagen.

3 Übersetzung von UML State Machines in Petri-Netze

Im Folgenden erläutern wir unseren Ansatz für die Transformation von UML State Machines in Stochastische Petri-Netze. Ziel ist dabei die quantitative Untersuchung der State Machines. Grundlegende Transformationsregeln wurden in [TrZi2005, TrZiHo2005] bereits detailliert beschrieben, worauf an dieser Stelle verzichtet wird.

Grundsätzlich basiert der Ansatz auf der Idee der Zerlegung der UML State Machines in ihre Elemente wie z.B. States, Pseudostates und Transitionen. Für jedes dieser Elemente wird eine Transformationsregel in ein Fragment eines stochastischen Petri-Netzes spezifiziert. Dabei werden auch die zusätzlichen Informationen des SPT Profiles übertragen. Von besonderem Interesse sind hierbei zusätzliche Zeitinformationen wie z.B. durch den *RTdelay* Stereotype. Die resultierenden Petri-Netz-Fragmente werden anschließend entsprechend der ursprünglichen Zusammenhänge zu einem Gesamtmodell zusammengesetzt [TrZiHo2005].

3.1 Stochastische Petri-Netze

Petri-Netze sind ein klassisches Modell für diskrete Ereignissysteme, in denen Synchronisation und nebenläufige Prozesse eine Rolle spielen. Ein Petri-Netz ist ein bipartiter Graph, dessen Knoten als Stellen und Transitionen bezeichnet werden. Stellen können Marken enthalten, und deren Verteilung auf die Stellen entspricht einem Zustand des Modells. Kanten verbinden Stellen mit Transitionen bzw. umgekehrt, und beschreiben damit die Abhängigkeit der aktiven Elemente (Transitionen) von Marken in Stellen sowie die Veränderung der Markierung durch ihr Schalten. Hier soll keine genaue Definition gegeben werden, statt dessen wird auf die umfangreiche Literatur zu diesem Thema verwiesen. Einen Überblick geben u.a. [Mur1989, Rei1985].

Für die quantitative Modellierung und Bewertung technischer Systeme existieren Erweiterungen, die den Transitionen Zeiten (entsprechend den Aktivitätsdauern) oder Schaltwahrscheinlichkeiten für den Konfliktfall zuordnen, vgl. [AjBaCo1995]. Im weiteren werden sog. *extended deterministic and stochastic Petri nets* (eDSPNs, [Ger2000]) verwendet, die die folgenden Transitionsarten erlauben. Abb. 2 zeigt beispielhaft ein Modell eines redundanten Zweikomponentensystems mit Reparatur. Komponenten entsprechen Marken in den Stellen; anfangs sind beide intakt (zwei Marken in p1).

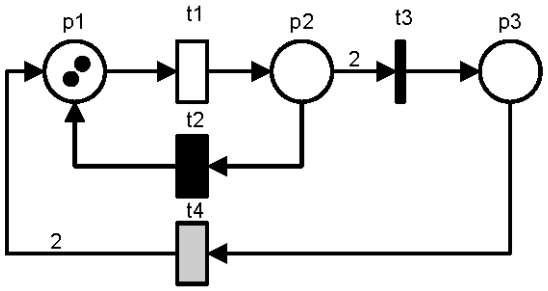

Abbildung 2: Beispiel eines stochastischen Petri-Netzes.

Zeitlose Transitionen werden als dünner Balken dargestellt (t3 im Bild), und beschreiben Zwischenaktivitäten ohne Dauer im Modell. Die ihnen zugeordneten Schaltwahrscheinlichkeiten erlauben die nichtdeterministische Auflösung von Konflikten. Zeitbehaftete Transitionen werden abhängig von der ihnen zugeordneten Schaltzeitverteilung grafisch als leeres Rechteck (t1, exponentielle Verteilungsfunktion), schwarzes Rechteck (t2, deterministische Schaltzeit), bzw. grau gefülltes Rechteck (t4, allgemeine Verteilungsfunktion) dargestellt.

3.2 Übersetzung einfacher Zustände und Transitionen

Die grundlegende Transformation einfacher Zustände und die einhergehenden festen Bezeichnungsregeln werden in [TrZi2005] detailliert vorgestellt. Hier wird kurz erläutert, wie zeitliche Informationen übertragen werden. Unabhängig davon, ob die optionalen Aktivitäten innerhalb eines Zustands spezifiziert sind oder nicht, wird immer der logischen und zeitlichen Abfolge der Aktivitäten entsprochen. Für jede Aktivität, die nicht spezifiziert ist oder keine zusätzliche Zeitinformation trägt, wird innerhalb des aus der Transformation resultierenden Petri-Netzes eine zeitlose Transition erzeugt.

Innerhalb eines jeden Zustands kann Zeit während der Ausführung der optionalen *entry*-, *do*- und *exit*-Aktivitäten verbraucht werden. Vergehende Zeit wird innerhalb eines Petri-Netzes durch eine Marke (*Token*) in einer Stelle und einer nachfolgenden zeitbehafteten Transition modelliert. Abhängig von den Zeitinformationen, die mittels SPT Profile hinzugefügt sind, wird den aus der Transformation resultierenden Transitionen ein entsprechendes Zeitverhalten zugeordnet. Tab.1 zeigt einige mögliche Annotationen aus dem SPT Profile für den Stereotyp *RTdelay*. Konstante Zeiten resultieren in deterministischen Transitionen, exponentiell verteilte Zeiten resultieren in exponentiellen Transitionen. Eine Syntax für die Spezifikation von allgemeinen Schaltzeitverteilungen, wie sie in eDSPN-Modellen möglich ist, wird derzeit entwickelt.

Bedeutung	Resultierende Transition im Petri-Netz und Schaltzeit	Annotation
Feste Zeit	Deterministisch, 8 Sekunden	RTduration = (8,'s')
Zufällige Dauer, exponentiell verteilt	Exponentiell, mittlere Schaltzeit 32 Sekunden	RTduration = ('exponential', 32,'s')
Zufällige Dauer mit bekanntem Quantil	Exponentiell, mittlere Schaltzeit ergibt sich aus $0.80 = F(5) = 1 - e^{-5\lambda}$	RTduration = ('percentile', 80, (5, 's'), 'exponential')
Komplexere Dauer	Allgemeine Schaltzeit, stückweise zusammengesetzt	

Tabelle 1: Annotationen für das Stereotyp *Rtdelay*

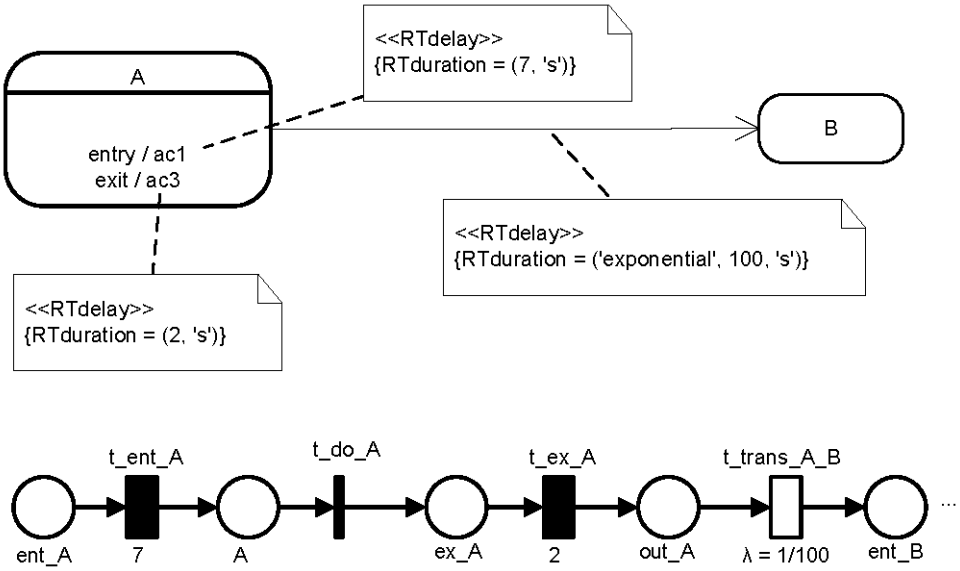

Abbildung 3: Übersetzung eines einfachen Zustandsübergangs.

Ist kein Zeitverhalten angegeben oder aber z.B. eine optionale Aktivität nicht spezifiziert, so resultiert das in einer zeitlosen Transition. Abb.3 zeigt ein Beispiel für die Transformation eines einfachen Zustandsübergangs unter Beachtung der Zeitinformationen aus dem SPT Profile. Die fehlende *do*-Aktivität in Zustand *A* resultiert in der zeitlosen Transition *t_do_A*. Die festen Zeiten für die optionalen *entry*- und *exit*-Aktivitäten resultieren jeweils in entsprechenden deterministischen Transitionen: *t_ent_A* und *t_ex_A*. Dem Zustandsübergang vom Zustand *A* zum Zustand *B* ist eine exponentiell verteilte Verzögerung mit einem Mittelwert von 100 Sekunden zugeordnet, so dass die resultierende exponentielle Transition *t_trans_A_B* eine Rate von 1/100 aufweist.

3.3 Übersetzung spezieller Konstrukte

Zu den speziellen Konstrukten und Elementen einer State Machine gehören z.B. Pseudostates, die Synchronisation von Regionen oder die Verwendung von Zählervariablen. Pseudostates sind Abstraktionen transienter Knoten innerhalb der State Machine. Sie haben eine spezielle Semantik, die bei der Transformation in ein Petri-Netz entsprechend beachtet werden muss. In [TrZi2005] wurden solche Transformationsregeln bereits unter anderem für *Choice-*, *Junction-*, *Fork-*, *Join-* und *Initial*-Pseudostates vorgestellt. Im folgenden werden nur zwei Erweiterungen erläutert.

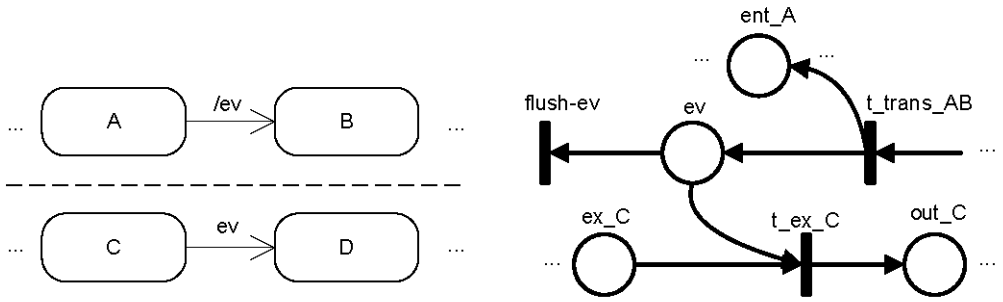

Abbildung 5: Synchronisation zwischen Regionen und ihre Übersetzung.

Die Synchronisation zwischen Regionen kann durch den Austausch eines Ereignisses (*event*) erfolgen. Abb. 5 zeigt ein Beispiel dafür. In der oberen parallelen Region (durch eine gestrichelte Linie getrennt) hat der Zustandsübergang von *A* nach *B* als Post-Kondition das Event *ev*. Es wird also ein solches Event durch den Zustandsübergang generiert. In der unteren Region wird hingegen der Zustandsübergang von *C* nach *D* nur dann möglich, wenn die Bedingung *ev* erfüllt ist, also wenn das Event *ev* in der oberen Region bereits generiert wurde. Wird das Event generiert, der Zustandsübergang von *C* nach *D* aber aus anderen Gründen nicht möglich ist, wird das Event verworfen.

Bei der Transformation in ein Petri-Netz wird eine Stelle gleichen Namens für den Event erzeugt (Stelle *ev* in Abb.5). Diese Stelle verbindet die erzeugten Petri-Netz-Komponenten für die einzelnen beteiligten Regionen dergestalt, dass der Zustand *C* (*ex_C*) nur verlassen werden kann (*t_ex_C*), wenn eine Marke in der Stelle *ev* vorhanden ist. Das ist der Fall, wenn der Zustandsübergang von *A* nach *B* erfolgt ist (*t_trans_AB*). Falls eine Marke in Stelle *ev*, aber keine in *ex_C* vorhanden sein, wird das Event (die Marke) durch das Schalten einer Transition mit geringerer Priorität (*flush_ev*) verworfen.

Die Verwendung von Zählervariablen im State Machine-Modell ist oft sinnvoll, wenn z.B. einzelne Zustandsübergänge nur mit einer bestimmten Häufigkeit auftreten dürfen. Ein einfaches Beispiel für die Verwendung eines Zählers ist in Abb. 6 dargestellt. Die verwendete Zählervariable *counter* wird beim Zustandsübergang von *A* nach *B* inkrementiert und beim Übergang von *B* nach *A* auf den Wert *0* zurückgesetzt.

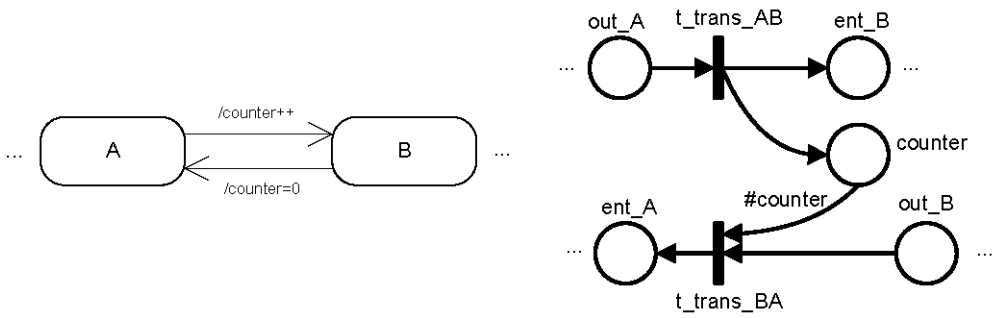

Abbildung 6: Zählervariablen in State Machines und ein entsprechendes Petri-Netz.

Bei der Transformation in ein Petri-Netz wird für eine solche Zählervariable eine entsprechende Stelle gleichen Namens erzeugt (Stelle *counter* in Abb. 6). Das Inkrementieren wird durch das Hinzufügen einer Marke abgebildet. Das Zurücksetzen der Variable auf Null geschieht durch das Schalten der Transition t_trans_BA, die alle Marken der Stelle durch ihre markierungsabhängige Kantenbeschriftung #counter aus der Stelle abzieht.

4 Ein Anwendungsbeispiel

Das zukünftige Europäische Eisenbahnleit- und Sicherungssystem ETCS (*European Train Control System*) wird mit dem Ziel einer europaweit einheitlichen und effizienten Steuerung eingeführt. Das herkömmliche Betriebsverfahren mit Blocksicherung wird in der dritten Anwendungsvariante (ETCS Level 3) aufgehoben und durch eine kontinuierliche Zuweisung freier Gleisbereiche ersetzt. Damit soll die durch die bisherige Blocksicherung erzwungene geringere Streckenauslastung verbessert werden. Die an den Strecken bisher notwendige elektromechanische Leit- und Sicherungstechnik (optische Signale, Achszähler etc.) kann dann durch ein funkbasiertes Verfahren ersetzt werden, um den Wartungsaufwand zu reduzieren. Teilaufgaben klassischer Stellwerke werden von Funkzugbeeinflussungsstellen (*Radio Block Center*, RBC) übernommen, die einen großen Streckenbereich und den darauf ablaufenden Zugverkehr verwalten. Eine zuverlässige und zeitgerechte Datenübertragung über die Funkstrecke sowie Datenverarbeitung in Zug und RBC sind daher kritische Faktoren für einen effizienten und sicheren Fahrbetrieb. Im folgenden wird ein Ausschnitt mit UML State Machines modelliert, in ein stochastisches Petri-Netz umgewandelt und daraus Leistungsmaße berechnet. Damit wird in Fortsetzung der bisherigen Arbeiten [ZimHom2003, ZimHom2005] erstmals der Zusammenhang zwischen Zuverlässigkeit und Echtzeitverhalten von Kommunikation und Datenverarbeitung, dem kleinsten möglichen Zugabstand, sowie der Wahrscheinlichkeit eines Nothalts quantifiziert.

4.1 Das Europäische Zugleit- und Sicherungssystem ETCS

Innerhalb des Projekts ERTMS/ETCS (*European Rail Traffic Management System/European Train Control System*) wird ein standardisiertes Zugsicherungssystem entwickelt, das die derzeit zahlreichen nationalen Systeme ersetzen soll. In der Anwendungsvariante ETCS Level 3 überprüft jeder Zug in regelmäßigen Abständen seine Vollständigkeit, und meldet seine Position an

das zuständige RBC. Für die Positionsbestimmung werden Achsumdrehungen gezählt. Aufgrund der Fehler durch Schlupf wird regelmäßig die Position an Balisen kalibriert, deren Position bekannt ist.

Jedes RBC überwacht die Positionen, Geschwindigkeiten und geplanten Routen der Züge in seinem Verantwortungsbereich. Es teilt dazu den einzelnen Zügen freie Gleisabschnitte zu, die von diesen dann sicher befahren werden können. Dies geschieht mit so genannten *movement authority messages*. Dieses Verfahren wird als *moving block operation* oder Fahren im beweglichen Raumabstand (im Gegensatz zum herkömmlichen Fahren im festen Raumabstand mittels Blocksicherungsverfahren) bezeichnet.

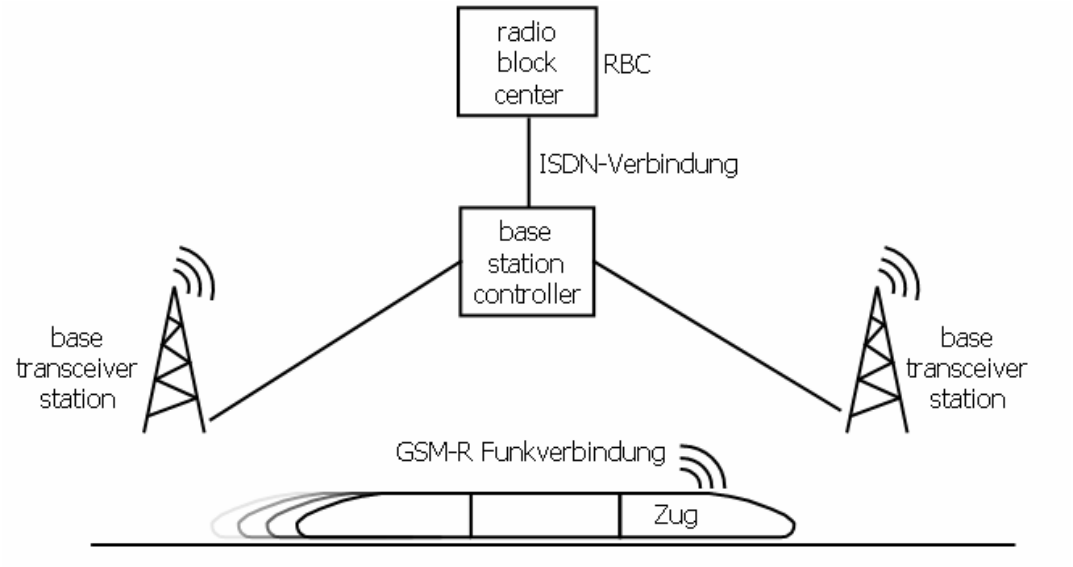

Abbildung 7: Vereinfachte Struktur der ETCS-Funkkommunikation.

Die Datenübertragung zwischen Zügen und RBC ist offensichtlich von entscheidender Bedeutung, denn sonst würde kein Zug mehr über die für ihn freie Strecke informiert werden können. Dies würde einen Hochgeschwindigkeitsbetrieb unmöglich machen. Abb. 7 stellt die Struktur der Funkkommunikation vereinfacht dar. Die Verbindung zwischen RBC und Zügen wird drahtlos über GSM-R (*global system for mobile communications - railway*) abgewickelt, einer Variante des aus dem Mobilfunk bekannten GSM-Systems [CorAnt2000]. *Base transceiver stations* der Funkschnittstelle übertragen die Daten an einen *base station controller*, von wo sie das Radio Block Center über eine ISDN-Verbindung erreichen. Die Funkkommunikation wurde im Projekt EIRENE (*European Integrated Railway Radio Enhanced Network*) genauer spezifiziert und entworfen [EIR1999]. Die EURORADIO-Schicht der Kommunikationsverbindung spezifiziert die Anforderungen an die Funkkommunikation [ERT2000a, Ken2002].

Im folgenden wird der zeitkritische Ablauf für die Bestimmung der freien Strecke betrachtet. Dabei werden *worst-case*-Annahmen laut Spezifikation verwendet, um die garantiert erreichbare bestmögliche Streckenauslastung zu berechnen. Ein Zug prüft zunächst seine Vollständigkeit, was laut Spezifikation zwei bis fünf Sekunden dauert. Anschließend sendet er die Position des Zugendes zu Beginn des Vollständigkeitstests („*min safe rear end*“) zum RBC. Dies geschieht in

regelmäßigen Abständen Δt, laut Spezifikation nicht öfter als alle fünf Sekunden. Offensichtlich ist die Genauigkeit der Streckeninformation um so höher, je öfter jeder Zug seine Position meldet; wir nehmen daher im folgenden Δt = 5sec an.

Die Positionsmeldung wird über GSM-R an das RBC gesendet, was laut Spezifikation im Mittel zwischen 400 und 500 Millisekunden dauert. Die Verarbeitung der empfangenen Daten im RBC dauert 500 Millisekunden, innerhalb derer unter anderem eine neue *movement authority message* für den folgenden Zug erzeugt wird. Die Übertragung dieser Nachricht dauert wiederum im Mittel zwischen 400 und 500 Millisekunden.

Die Funkkommunikation über GSM-R ist jedoch nicht sicher, und Pakete können verzögert werden oder verloren gehen. Der Zug darf dabei trotz der hohen gefahrenen Geschwindigkeiten und langen Bremswege nie den ihm zugewiesenen freien Streckenabschnitt verlassen. Jeder Zug muss daher nach dem Ablauf einer u. a. von der aktuellen Geschwindigkeit und der zugewiesenen freien Streckenlänge abhängigen Frist entscheiden, wann eine Weiterfahrt nicht mehr sicher ist und daher im äußersten Fall eine Notbremsung ausgelöst werden muss.

Unter ETCS Level 3 soll im absoluten Bremsabstand gefahren werden. Dieser beinhaltet neben dem tatsächlichen Bremsabstand bis zum sicheren Zugendes des vorausfahrenden Zuges einen Sicherheitszuschlag. Die Größe dieses Zuschlages beeinflusst damit die Toleranz gegenüber GSM-R Kommunikationsfehlern sowie den minimal möglichen Zugabstand. Wir betrachten zwei Züge Zug 1 und Zug 2, die auf freier Strecke mit gleicher Geschwindigkeit v hintereinander herfahren, wobei die Zugköpfe den Abstand s haben. Ziel ist nun die Berechnung der Frist d nach dem Empfang der letzten *movement authority message*, nach der Zug 2 ohne weitere Nachricht bremsen muss. Abb. 8 stellt die Zusammenhänge grafisch dar.

Vom Abstand s müssen Zuglänge (etwa 410m für ICEs), Ortsfehler (max. 20m) und Bremsweg (geschwindigkeitsabhängig 2300-2800m) abgezogen werden. Die Summe dieser drei Parameter wird im folgenden mit l = 3000m angesetzt.

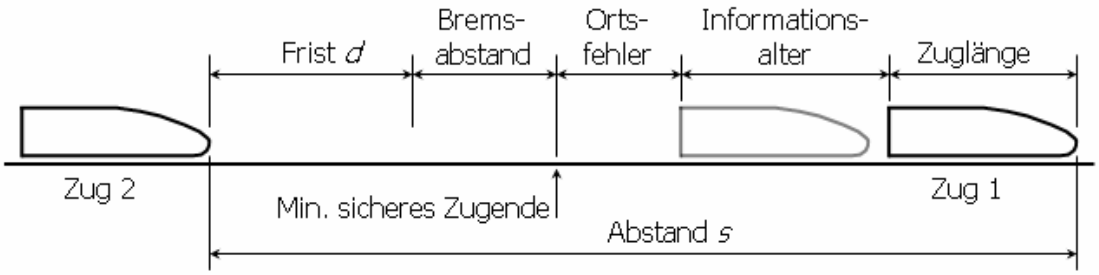

Abbildung 8: Zugabstand und Frist für Streckenfreigabe.

Im ungünstigsten Fall ist Zug 1 nach dem Bestimmen der Vollständigkeit havariert oder hat einen Teil der Wagen verloren. Von der verfügbaren Wartezeit muss daher außerdem noch die Zeitdifferenz a zwischen Empfang der Nachricht durch Zug 2 und dem Bestimmen der Vollstän-

digkeit von Zug 1 abgezogen werden. Entsprechend der Detailinformationen in den Spezifikationen liegt das entsprechende Alter der Information *a* zwischen fünf und neun Sekunden.

Die Frist *d* kann dann entsprechend $d = \frac{s - l}{v} - a$ berechnet werden, wobei im folgenden von v = 83 m s^{-1} entsprechend der Fahrgeschwindigkeit derzeitiger ICEs ausgegangen wird.

4.2 Ein Modell der Zugkommunikation

Die Fähigkeit, Datenpakete mit Positions- und Integritätsberichten sowie Autorisationspakete für die Zugbewegung austauschen zu können, ist entscheidend für den zuverlässigen Betrieb von ETCS. In diesem Abschnitt wird ein Modell für den *moving block* Betrieb und den dafür notwendigen Datenaustausch aufgestellt, wobei die Zuverlässigkeit des Kommunikationskanals beachtet wird. Das Modell wird als UML State Machine konstruiert und basiert auf den folgenden zu Beginn der Arbeiten verfügbaren Informationsquellen über das quantitative und qualitative Verhalten des Kommunikationssystems und seine Betriebsstörungen:

- Eine Spezifikation von *Quality of Service*-Parametern (maximale Verzögerung beim Verbindungsaufbau etc.) ist gegeben durch *Euroradio form fit functional interface specification* (FFFIS) [ERT2000a].

- Erlaubte Größenordnungen der Parameter für einzelne Systementwurfsvariablen, wie z.B. die minimale Zeit zwischen dem Senden zweier aufeinander folgender Positionsberichte durch den Zug, sind in *ERTMS Performance Requirements for Interoperability* [ERT2000b] spezifiziert. Empfehlungen für die nationale Festlegung derartiger Parameter im Rahmen der europaweit geltenden Grenzen werden in [ThoHo2004] gegeben.

- Definitionen von Anforderungen bzgl. Verlässlichkeit, Verfügbarkeit, Wartbarkeit und Sicherheit und die akzeptierte Anzahl an Betriebsstörungen per Passagierkilometern können *ETCS RAMS specification* [ERT1998] entnommen werden.

- Einige zusätzliche Annahmen (mittlere Zeit, um den der Zug für einen Integritätscheck benötigt etc.) wurden aus der Beschreibung von Simulationsexperimenten der Deutschen Bahn übernommen [Osb2002].

- Eine weitere detaillierte Beschreibung von *Quality of Service*-Parametern für die Kommunikation wird durch [Göl2002] bereitgestellt, wobei diese als akzeptiertes Kriterium für zukünftige Messungen und Tests aktueller ETCS Kommunikationsaufbauten dient.

Im Folgenden gehen wir von einem Worst-Case-Szenario basierend auf den Anforderungen aus, da sonst kein funktionierendes Gesamtsystem garantiert werden kann. Ein Modell für den Austausch von Datenpaketen mit Positionsinformationen und für das Anhalten des Zugs bei Kommunikationsproblemen zwischen Zug und RBC wird beschrieben. Das Ziel ist dabei die Untersuchung der Abhängigkeiten zwischen dem maximalen Durchsatz an Zügen und der Verlässlichkeit des Kommunikationssystems.

Die UML State Machine in Abb. 9 beschreibt die ETCS-Zugkommunikation. Es sind fünf parallele Regionen vorhanden, die nachfolgend im Einzelnen erläutert werden. Die oberste Region modelliert das Generieren von Positions-/Integritätspaketen durch Zug 1. Ein solches Paket wird alle 5 Sekunden generiert, wobei ein Event *TrainSend* erzeugt wird. Das Senden der Datenpakete vom Zug zum RBC über den Kommunikationskanal wird in der darunter liegenden Region beschrieben. Der Kommunikationskanal kann die Zustände *Empty* (keine Sendeaktivität) oder *Full* (Paket wird gesendet) haben. Mit dem Event *TrainSend* ist ein neues Datenpaket zum Senden an das RBC bereit. Dieses Paket wird mit einer Wahrscheinlichkeit von 1.88% nicht korrekt übertragen, was in der Region durch einen *Choice*-Pseudostate und die entsprechenden *PAprob* Annotationen an den ausgehenden Transitionen modelliert wird. Der Wert ergibt sich daraus, dass laut Spezifikation von einer Bitfehlerrate von 10^{-4} ausgegangen wird und die Paketlänge 190 Bit beträgt, und damit P(Fehler) = 1- $(1\text{-}10^{-4})^{190}$ = 1.88%. Die korrekte Übertragung eines Pakets dauert im Mittel 0.45 Sekunden, wobei nur die Gesamtübertragungszeit gegeben ist, also eine Unterscheidung zwischen den Zeiten der Funkübertragung und im ISDN-Kanal nicht möglich ist. Wird der Kanal wieder frei, so wird bei dem entsprechenden Zustandsübergang das Event *RBCreceive* erzeugt.

Dieses führt dazu, dass in der nachfolgenden Region, die das Verhalten innerhalb des RBC modelliert, der Zustandsübergang von *Idle* nach *Busy* stattfindet. Die Verarbeitung des eingegangenen Pakets dauert 0.5 Sekunden. Beim anschließenden Übergang in den *Idle*-Zustand wird dann das Event *RBCsend* erzeugt.

Die nächste Region beschreibt das Senden einer *movement authority message* durch das RBC an Zug 2. Die einzigen Unterschiede zum Senden vom Zug zum RBC sind die unterschiedlichen Events, die eine Rolle spielen. *RBCsend* aktiviert den Übergang vom *Empty*- in den *Full*-Zustand, wobei natürlich wiederum ein Fehler bei der Übertragung auftreten kann. Nach einer korrekten Übertragung wird das Event *TrainReceive* erzeugt.

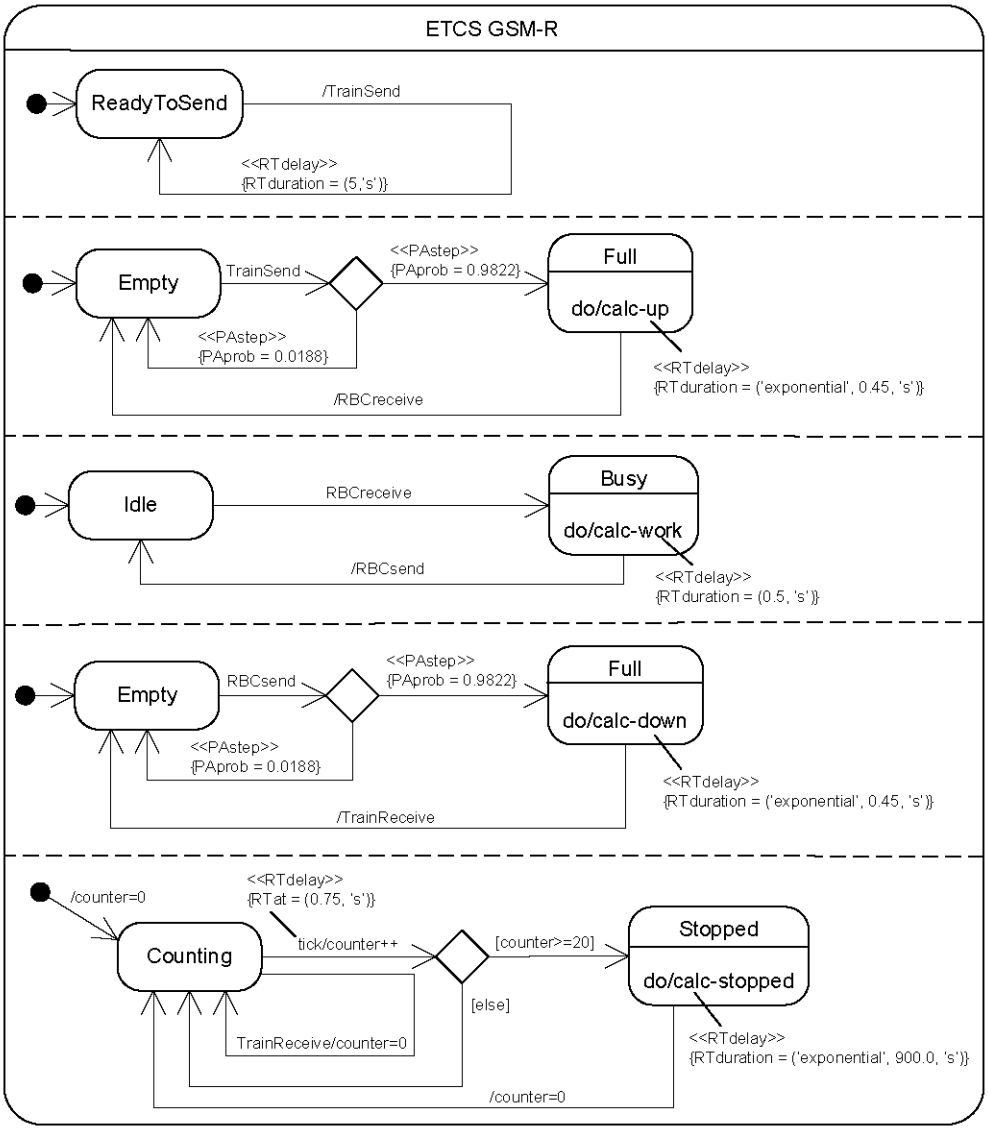

Abbildung 9: UML State Machine Modell der ETCS-Zugkommunikation.

Die unterste Region modelliert die Überwachung der Frist, innerhalb derer eine neue Nachricht empfangen werden muss. Dafür wird eine Zählervariable (*counter*) verwendet. Es gibt zwei Zustände: *Counting* und *Stopped*. Alle 0.75 Sekunden wird bei einer beispielhaften Frist von 15 Sekunden ein Event *Tick* generiert, was jeweils zu einer Inkrementierung der Zählervariablen *counter* führt. Sobald *counter* einen Wert von 20 erreicht hat, führt der Zug einen Nothalt durch. Dafür wird eine mittlere Zeit von 900 Sekunden (15 Minuten) angenommen. Anschließend wird *counter* auf 0 zurückgesetzt und der Zug fährt wieder los. Ist der Wert von *counter* kleiner als 20, so wird in den Zustand *Counting* zurückgesprungen und mit dem nächsten *Tick* erneut inkrementiert. Befinden sich die Region im Zustand *Counting* und das Event *TrainReceive* wird empfangen, so wurde eine neue *movement authority message* erhalten und die Zählervariable *counter* wird auf 0 zurückgesetzt.

4.3 Übersetzung und resultierendes Petri-Netz

Das im Abschnitt 4.2 vorgestellte Modell der ETCS-Zugkommunikation wurde mit den Regeln aus Abschnitt 3 in ein stochastisches Petri-Netz übersetzt. Das Ergebnis enthält noch einige Abfolgen zeitloser Transitionen und Stellen, die auf das Modellverhalten keinen Einfluss haben. Sie entstehen durch die Übersetzung jedes einzelnen State Machine-Elements, die im Modell nicht alle mit Werten oder Aktionen belegt sind. Daher wurde das Petri-Netz-Modell mit zwei einfachen strukturellen Regeln vereinfacht, die den beschriebenen stochastischen Prozess und damit das Verhalten lediglich um verschwindende Zustände (*vanishing states*) reduzieren.

Zunächst werden alle Sequenzen Stelle – zeitlose Transition – Stelle gesucht, in denen die Transition die einzige Verbindung der Stellen ist, und nicht in strukturellem Konflikt mit anderen Transitionen steht. Dann können die beiden Stellen verschmolzen und die Transition gelöscht werden. Die zweite Vereinfachung betrachtet analog dazu alle einfachen Abfolgen Transition – Stelle – zeitlose Transition. In diesem Fall kann die Stelle gelöscht und die Transitionen miteinander verschmolzen werden. Die Reduktionsregeln sind in Abb. 10 schematisch dargestellt.

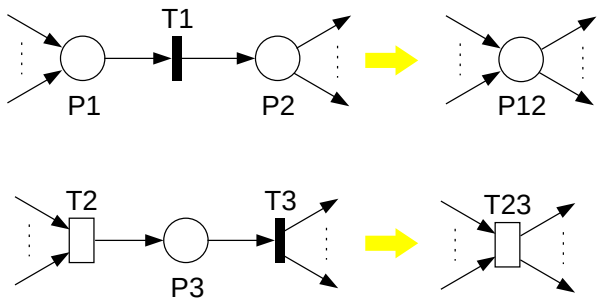

Abbildung 10: Einfache Reduktionsregeln für stochastische Petri-Netze

Das nach Anwendung dieser Regeln entstehende Modell für das Anwendungsbeispiel zeigt Abbildung 11. Die fünf den Regionen der State Machine entsprechenden Teile des Petri-Netzes sind durch gestrichelte Linien getrennt.

Der Anfangszustand der Teilmodelle wird durch das Schalten der Transition t_init erreicht. Danach hat jedes Teilmodell ein eigenes lokales Verhalten, dessen Zustände durch den Ort einer Marke bestimmt wird. Das RBC kann beispielsweise in den Zuständen c_idle oder busy sein, je nachdem, ob gerade gewartet oder eine Nachricht verarbeitet wird. Die Erzeugung neuer Nachrichten im vorderen Zug findet ständig alle 5 Sekunden statt, daher hat die deterministische Transition t_trans_rts_rts eine Schaltzeit von 5 und ist nach ihrem Schalten sofort wieder aktiviert. Die Modelle des Kommunikationskanals zwischen Zug und RBC in Hin- und Rückrichtung sind sehr ähnlich. Nach dem Empfang wird mit dem Schalten einer der in Konflikt stehenden zeitlosen Transitionen t_choice_e und t_choice_f die Paketverlustwahrscheinlichkeit abgebildet. Den Transitionen sind entsprechende Schaltwahrscheinlichkeiten durch die Übersetzung zugeordnet. Die Dauer der gesamten Übertragung wird durch die Transitionen t_do_full bzw. t_do_full2 mit exponentiell verteilter Schaltzeit abgebildet.

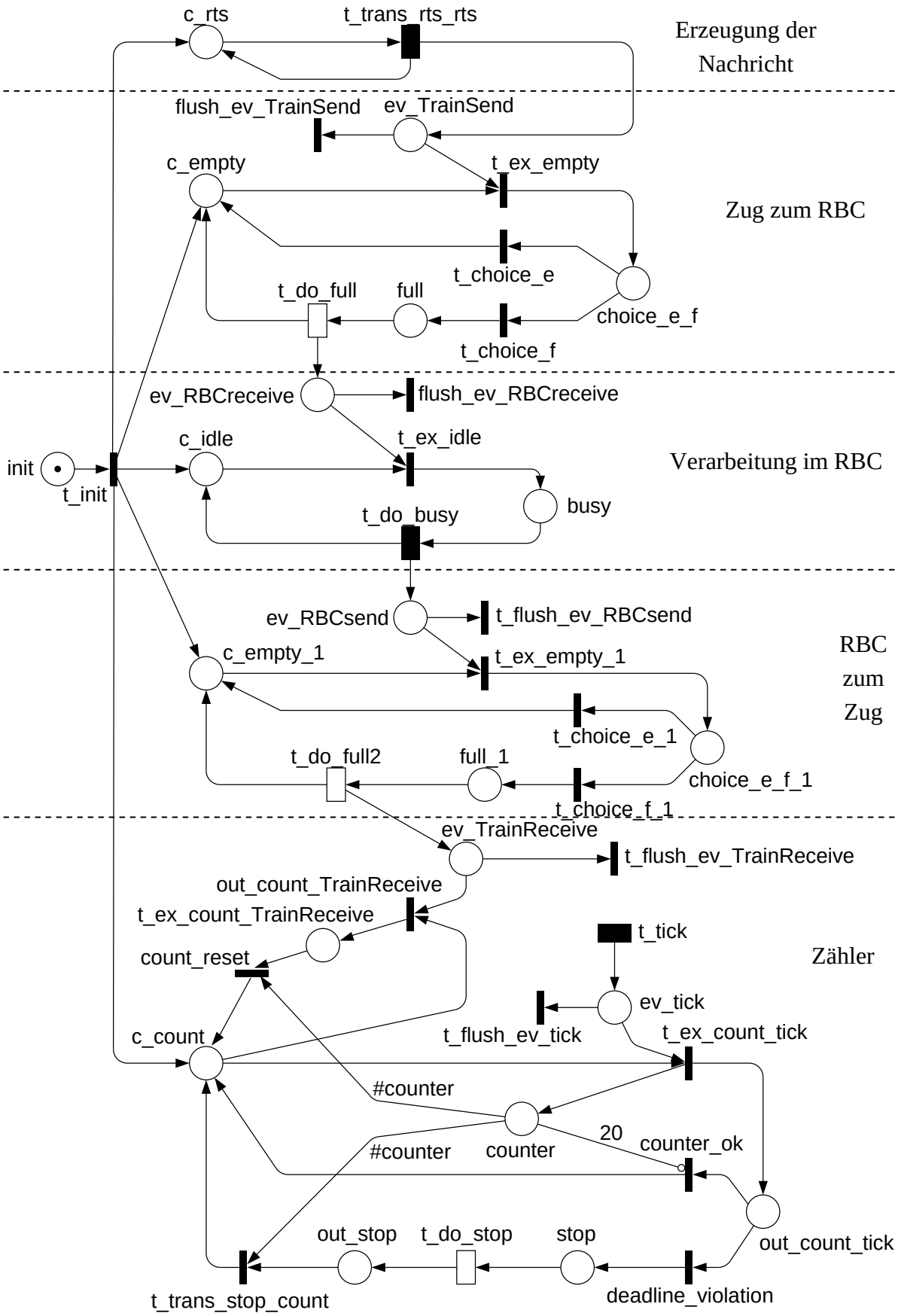

Abbildung 11: Resultierendes Petri-Netz für die ETCS Zugkommunikation.

Austausch von Nachrichten zwischen den Teilmodellen findet über jeweils gleiche Konstruktionen statt. Eine vom Zug zum RBC zu sendende Nachricht entspricht beispielsweise einer Marke in Stelle ev_TrainSend, die dann sofort entweder gesendet wird (Transition t_ex_empty schaltet aufgrund ihrer höheren Priorität, wenn eine Marke in c_empty vorhanden ist) oder verloren geht (Transition flush_ev_TrainSend schaltet). So wird die Semantik der Synchronisation aus State Machines wie in Abschnitt 3.3 beschrieben nachgebildet.

Das unterste Teilmodell beschreibt das Verhalten des Zählers für die Frist bis zur Einleitung eines Nothalts. Im Grundzustand befindet sich eine Marke in c_count. Dann können zwei Ereignisse eintreten: entweder wird eine neue Nachricht empfangen (t_ex_count_TrainReceive schaltet, und im folgenden werden alle Marken aus der Stelle counter mit der markierungsabhängigen Kantengewichtung #counter von count_reset gelöscht), oder ein Zeittakt wird ausgelöst (t_tick schaltet). Danach wird entweder wieder der Grundzustand eingenommen (counter_ok schaltet, falls der Zähler noch nicht 20 erreicht hat), oder ein Nothalt wird ausgelöst. Nach dessen Ende (t_do_stop schaltet) beginnt der Ablauf wieder.

4.4 Ergebnisse der quantitativen Bewertung

Das in Abschnitt 4.3 dargestellt Modell kann nun einer Leistungsbewertung unterzogen werden. Dazu wird zunächst ein Maß P_Stop = P{#counter >= 20} definiert, das die Wahrscheinlichkeit dafür misst, dass der Zug sich im Zustand eines Nothalts durch Überschreiten der Frist befindet. Eine Analyse im eingeschwungenen Zustand ergibt dann die mittlere Wahrscheinlichkeit im Betrieb bzw. den mittleren Zeitanteil, den ein Zug in diesem unerwünschten Zustand verbringt.

Die Bestimmung des Maßes ist mit keinem bekannten numerischen Analyseverfahren möglich, da mehrere Transitionen mit nichtexponentieller Schaltzeit gleichzeitig aktiviert sind. Standard-Simulationsverfahren sind nur begrenzt einsetzbar, da die praxisrelevanten Wahrscheinlichkeiten für einen Nothalt sehr gering sind, und daher das bekannte Problem seltener Ereignisse zu unakzeptabel hohen Laufzeiten führt. Die Untersuchungen wurden daher mit dem RESTART-Verfahren durchgeführt [ViViGa1994], einer Variante von *importance splitting* für die beschleunigte Simulation seltener Ereignisse. Zur Modellierung wurde das Softwarewerkzeug TimeNET [ZiFrGe2000] eingesetzt, das eine RESTART-Implementierung für stochastische Petri-Netze enthält [KeHo1996]. Die Anzahl von Marken in der Stelle counter wird für die Definition von Schwellwerten für das RESTART-Verfahren benutzt. Das Werkzeug bestimmt die Schwellen automatisch mit Hilfe einer Vorabsimulation.

Abb. 12 zeigt den Zusammenhang zwischen Zugabstand und der sich daraus ergebenden Haltewahrscheinlichkeit. Der Einfluss des Alters der empfangenen Information wird durch die zwei entsprechend 5 bzw. 9 Sekunden dargestellten Kurven gezeigt. Ab einem Abstand s von etwa 4.5km zeigen die Kurven eine nahezu logarithmische Abhängigkeit der Wahrscheinlichkeit vom Abstand. Für eine (sicherlich noch viel zu hoch gegriffene) mittlere Anzahl von höchstens einem durch Kommunikationsfehler erzeugten Nothalt pro Zug und Betriebsjahr müsste der Abstand mindestens s = 6km betragen.

Die Ergebnisse wurden auf einem Intel Pentium III Mobile PC mit 1GHz unter Linux durchgeführt. Die Effizienz des RESTART-Algorithmus wird daran deutlich, dass selbst für den aufwändigsten Simulationslauf (Wahrscheinlichkeit des seltenen Ereignisses liegt in der Größenordnung 10^{-16}) nach lediglich zehn Minuten Rechenzeit das Konfidenzintervall zum Niveau 90% bereits so klein ist, dass seine Breite nur noch etwa 10% des Ergebniswertes beträgt. Eine Standardsimulation hätte innerhalb dieser Zeit keine statistisch verwertbaren Ergebnisse produziert.

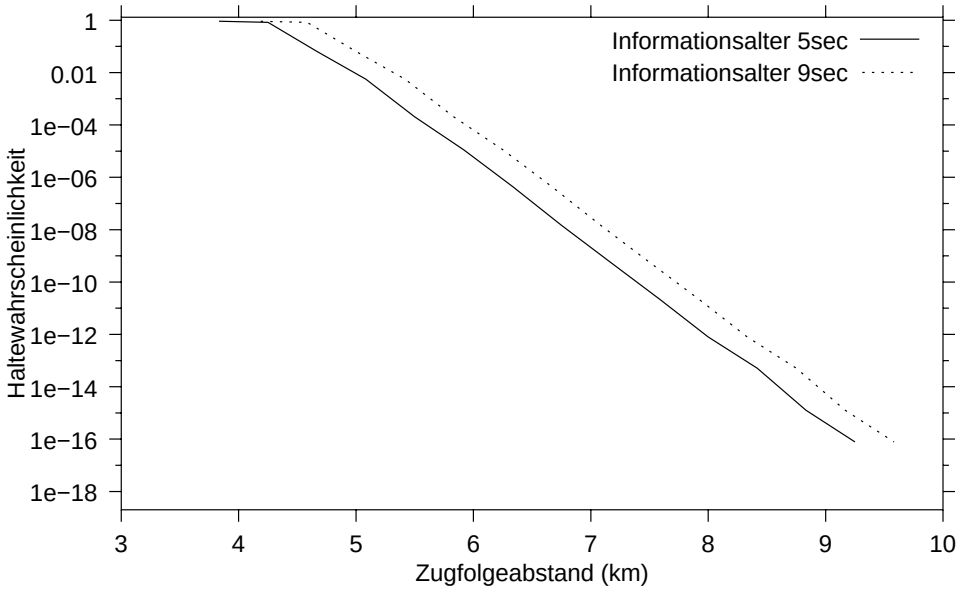

Abbildung 12: Wahrscheinlichkeit eines Nothalts in Abhängigkeit vom Zugabstand.

Die Untersuchung zeigt den hohen Einfluss der unsicheren Kommunikation über GSM-R (Paketverluste und –verzögerungen) auf die maximal mögliche Streckenauslastung unter ETCS Level *3*. Die für ETCS teilweise vorhandene Vorstellung des „Fahrens im Bremsabstand“ ist unrealistisch. Für einen sicheren Fahrbetrieb sind offensichtlich deutlich größere Zugabstände nötig. Ein genauerer Vergleich mit dem derzeitigen Betriebsverfahren durch Blocksicherung (Fahren im festen Raumabstand) kann [ZimHom2005] entnommen werden.

5 Zusammenfassung

Der vorliegende Beitrag beschreibt, wie das Verhalten technischer Systeme mit State Machines entsprechend der *Unified Modeling Language* abgebildet und anschließend quantitativ untersucht werden kann. Dazu werden zunächst Erweiterungen des *UML Profile for Schedulability, Performance, and Time Specification* genutzt, um die notwendigen zusätzlichen Informationen wie die Dauer und Eintrittswahrscheinlichkeit von Aktivitäten in State Machines abzubilden. Es wird eine Methode vorgeschlagen, mit der die entstehenden Modelle automatisch in ein stochastisches Petri-Netz umgewandelt werden können. Leistungsmaße sind dann mit verfügbaren Petri-Netz-Softwarewerkzeugen über Simulation oder numerische Analyse bestimmbar.

Als Anwendungsbeispiel wird ein Ausschnitt der Kommunikation zwischen Zügen und Radio Block Centers des zukünftigen Europäischen Zugleit- und Sicherungssystems ETCS betrachtet.

Der sichere und rechtzeitige Austausch von Kommunikationspaketen ist notwendig, um einen sicheren und effizienten Bahnbetrieb zu gewährleisten, da in der geplanten letzten Ausbaustufe dazu keine gleisseitigen Anlagen mehr eingesetzt werden. Der Beitrag stellt ein vereinfachtes State Machine Modell der Kommunikation sowie seine Übersetzung in ein stochastisches Petri-Netz vor. Die anschließende quantitative Untersuchung mit Hilfe effizienter Simulationsverfahren für seltene Ereignisse zeigt, wie groß der Einfluss der Kommunikationsgüte auf den kürzesten möglichen Zugfolgeabstand ist. Die Ergebnisse lassen es fraglich erscheinen, ob unter den in den Spezifikationen geforderten Bedingungen ein effizienter Fahrbetrieb für Hochgeschwindigkeitszüge realisierbar ist. Weitere detailliertere Untersuchungen sind notwendig, um Sicherheit und Wirtschaftlichkeit technischer Lösungen in diesem Bereich genauer zu bewerten.

Die beschriebenen Methoden werden derzeit als Erweiterung des Softwarewerkzeugs TimeNET implementiert. UML State Machine Modelle können direkt in der grafischen Benutzungsoberfläche eingegeben oder aus einem UML-Editor übernommen werden. Übersetzung, Analyse und Ergebnisanzeige sind in TimeNET implementiert. Außerdem ist eine Übertragung der Methode auf die aktualisierten Spezifikationen von ETCS geplant.

Die Forschungsarbeiten werden von der Deutschen Forschungsgemeinschaft im Graduiertenkolleg 621 „Stochastische Modellierung und quantitative Analyse großer Systeme in den Ingenieurwissenschaften“ gefördert.

6 Literatur

[AjBaCo1995] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. Series in parallel computing. John Wiley, 1995.

[BeDoMe2002] Bernardi, S., Donatelli, S., Merseguer, J.: From UML Sequence Diagrams and Statecharts to analysable Petri Net models. Proc. 3rd Int. Workshop on Software and Performance (WOSP), S. 35–45, Rom, 2002.

[CorAnt2000] Coraiola, A., Antscher, M.: GSM-R network for the high-speed line Rome-Naples. Signal und Draht, Vol. 92 (5), S. 42–45, 2000.

[ChiCiPo1999] Chiappini, A., Cimatti, A., Porzia, C., Rotondo, G., Sebastiani, R., Traverso, P., Villafiorita, A.: Formal specification and development of a safety-critical train management system. SAFECOMP, S. 410–419, 1999.

[DaKl2001] Damm, W., Klose, J.: Verification of a radio-based signaling system using the STATEMATE verification environment. Formal Methods in System Design, Vol. 19 (2), S. 121–141, 2001.

[ERT1998] EEIG ERTMS User Group: ERTMS/ETCS RAMS Requirements Specification. UIC, Brüssel, 1998.

[ERT1999] EEIG ERTMS User Group: ERTMS/ETCS System Requirements Specification. UIC, Brüssel, 1999.

[ERT2000a] EEIG ERTMS User Group: Euroradio FFFIS, UIC, Brüssel, 2000.

[ERT2000b] EEIG ERTMS User Group: Performance Requirements for Interoperability. UIC, Brüssel, 2000.

[EIR1999] EIRENE Project Team: EIRENE System Requirements Specification. UIC, Brüssel, 1999.

[Ger2000] German, R.: Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets. John Wiley and Sons, 2000.

[Göl2002] Göller, M., Lengemann, L.: Measurement and evaluation of the quality of service parameters of the communication system for ERTMS. Signal und Draht, Vol. 94 (1+2), S. 19–26, 2002.

[Har1998] Harel, D., Politi, M.: Modeling Reactive Systems with Statecharts: The StateMate Approach. New York, Wiley, 1998.

[HoSmKi2002] Hopkins, R. P., Smith, M. J., King, P.: Two approaches to integrating UML and performance models. Proc. 3rd Int. Workshop on Software and Performance, S. 91–92, 2002.

[JaHe2004] Jansen, D. N., Hermanns, H.: Dependability checking with StoCharts: Is train radio reliable enough for trains? Proc. 1st Int. Conf. on the Quantitative Evaluation of Systems (QEST), Enschede, Niederlande, S. 250–250, 2004.

[JaMeSch1998] Jansen, L., Meyer zu Hörste, M., Schnieder, E.: Technical issues in modelling the european train control system. Proc. 1st CPN Workshop, DAIMI PB 532, Aarhus University, S. 103–115, 1998.

[KeHo1996] Kelling, C., Hommel, G.: A framework for rare event simulation of stochastic Petri nets using RESTART. Proc. of the Winter Simulation Conference, S. 317–324, 1996.

[Ken2002] Kendelbacher, D., Stein, F.: EURORADIO – communication base system for ETCS. Signal und Draht, Vol. 94 (6), S. 6–11, 2002.

[KiPo1999] King, P., Pooley, R.: Using UML to derive stochastic Petri net models. Proc. 15th UK Performance Engineering Workshop, Bristol, UK, S. 45–56, 1999.

[KiPo2000] King, P., Pooley, R.: Derivation of Petri net performance models from UML specifications of communications software. Proc. 11th Int. Conf. on Tools and Techniques for Computer Performance Evaluation, Schaumburg, Illinois, USA, S. 262–276, 2000.

[LiThüKl2002] Lindemann, C., Thümmler, A., Klemm, A., Lohmann, M., Waldhorst, O.: Performance Analysis of Time-enhanced UML Diagrams Based on Stochastic Processes. Proc. of the 3rd Workshop on Software and Performance (WOSP), Rom, S. 25-34, 2002.

[Mer2004] Merseguer, J.: On the use of UML State Machines for software performance evaluation. Proc. 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2004.

[MeSch1999] Meyer zu Hörste, M., Schnieder, E.: Modelling and simulation of train control systems using Petri nets. Formal Methods – World Congress on Formal Methods in the Development of Computing Systems (FM'99), LNCS 1709, Springer Verlag, 1999.

[Mur1989] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE. Vol. 77 (4), S. 541-580, 1989.

[OMG2003] Object Management Group: Unified Modeling Language Specification v.2.0. http://www.uml.org, 2003.

[OMG2002] Object Management Group: UML Profile for Schedulability, Performance, and Time. http://www.uml.org, 2002.

[Osb2002] Osburg, J.: Performance investigation of arbitrary train control techniques. Signal und Draht, Vol. 94 (1+2), S. 27–30, 2002.

[Rei1985] W. Reisig: Petri nets. Springer Verlag Berlin, 1985.

[PoKi1999] Pooley, R., King, P.: The Unified Modeling Language and Performance Engineering. IIEE Proceedings Software. Vol. 146 (2), 1999.

[ThoHo2004] Thomas, K., Holgate, D. J.: Baseline ERTMS Parameters Recommendations for National Values. UK Rail Safety & Standards Board and AEA Technology Rail, 2004.

[TrZi2005] Trowitzsch, J., Zimmermann, A.; Real-time UML state machines: An analysis approach. Workshop on Object Oriented Software Design for Real Time and Embedded Computer Systems, Net.ObjectDays 2005, Erfurt, Germany, 2005.

[TrZiHo2005] Trowitzsch, J., Zimmermann, A., Hommel, G.: Towards Quantitative Analysis of Real-Time UML Using Stochastic Petri Nets. 13th IEEE Int. Workshop on Parallel and Distributed Real-Time Systems Denver, Colorado, 2005.

[ViViGa1994] Villén-Altamirano, M., Villén-Altamirano, J., Gamo, J., Fernández-Cuesta, F., Enhancement of the accelerated simulation method RESTART by considering multiple thresholds. Proc. 14th Int. Teletraffic Congress. Elsevier, S. 797–810, 1994.

[ZiFrGe2000] Zimmermann, A., Freiheit, J., German R., Hommel, G.: Petri net modelling and performability evaluation with TimeNET 3.0. 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, LNCS 1786, Schaumburg, Illinois, USA, S. 188–202, 2000.

[ZimHom2003] Zimmermann, A., Hommel, G.: A train control system case study in model-based real time system design. Proc. 11th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS03), Nizza, 2003.

[ZimHom2005] Zimmermann, A., Hommel, G.: Towards modeling and evaluation of ETCS real-time communication and operation. Journal of Systems and Software, Vol. 77, S. 47–54, 2005.