Thomas Freytag, Andreas Eckleder (Hrsg.)

Algorithmen und Werkzeuge für Petrinetze

16ter Workshop, AWPN 2009
Karlsruhe, 25. September 2009
Proceedings

Herausgeber:

Thomas Freytag, Andreas Eckleder
Duale Hochschule Baden-Württemberg Karlsruhe, 76133 Karlsruhe, Germany
`freytag@dhbw-karlsruhe.de, andreas@eckleder.de`

ISSN 1613-0073 (CEUR Workshop Proceedings)
Online-Proceedings verfügbar unter http://CEUR-WS.org/Vol-501/

BibTeX-Eintrag für Online-Proceedings:

```
@proceedings{awpn2009,
   editor = {Thomas Freytag and Andreas Eckleder},

   title = {Proceedings of the 16th German Workshop
 on Algorithms and Tools for Petri Nets, AWPN 2009,
 Karlsruhe, Germany, September 25, 2009},
   booktitle = {Algorithmen und Werkzeuge f\"ur Petrinetze},

   publisher = {CEUR-WS.org},
   series = {CEUR Workshop Proceedings},
   volume = {501},
   year = {2009},
   url = {http://CEUR-WS.org/Vol-501/}
   }
```

Copyright © 2009 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. Re-publication of material from this volume requires permission by the copyright owners.

Vorwort

Seit 1994 bietet der Workshop "Algorithmen und Werkzeuge für Petrinetze" (AWPN) ein gemeinsames Forum für Entwickler und Anwender petrinetzbasierter Technologie. Außerdem bildet er dank des traditionell geringen finanziellen Aufwands für die Teilnahme und der deutschsprachigen Ausrichtung eine Möglichkeit für Nachwuchswissenschaftler(innen), Erfahrungen bei einer wissenschaftlichen Veranstaltung zu sammeln.

Im Jahr 2009 findet der Workshop in seiner 16ten Ausgabe zum zweiten Mal nach 1996 in Karlsruhe statt, erstmals an der Dualen Hochschule Baden-Württemberg (DHBW). Veranstalter ist wie immer die Fachgruppe Petrinetze und verwandte Systemmodelle der Gesellschaft für Informatik.

Es gab sieben eingereichte Beiträge, die alle nach kurzer Prüfung durch die Fachgruppenleitung in das Programm aufgenommen wurden. Ein Begutachtungsprozess fand wie auch in den vergangenen Jahren nicht statt. Wir hoffen, dass die Vorträge eine gute Grundlage für rege Diskussionen bieten.

Die Organisatoren danken der Fakultät für Wirtschaft der DHBW Karlsruhe für die finanzielle und logistische Untersützung der Ausrichtung.

September 2009 Thomas Freytag und Andreas Eckleder

Steering Committee

Jörg Desel (Stellvertreter)	Katholische Universität Eichstätt-Ingolstadt
Ekkart Kindler	Technical University of Denmark
Kurt Lautenbach	Universität Koblenz-Landau
Robert Lorenz	Universität Augsburg
Daniel Moldt	Universität Hamburg
Rüdiger Valk	Universität Hamburg
Karsten Wolf (Sprecher)	Universität Rostock

Bisherige AWPN-Workshops

1. Berlin 1994
2. Oldenburg 1995
3. Karlsruhe 1996
4. Berlin 1997
5. Dortmund 1998
6. Frankfurt 1999
7. Koblenz 2000
8. Eichstätt 2001
9. Potsdam 2002
10. Eichstätt 2003
11. Paderborn 2004
12. Berlin 2005
13. Hamburg 2006
14. Koblenz 2007
15. Rostock 2008

Inhaltsverzeichnis

Modellierung und Mining Kollaborativer Learnflows

Robin Bergenthum, Jörg Desel, Andreas Harrer, Sebastian Mauser

Fachgebiet Informatik, Katholische Universität Eichstätt-Ingolstadt
vorname.name@ku-eichstaett.de

Zusammenfassung Basierend auf Ideen aus dem Bereich der Geschäftsprozessmodellierung werden zwei Ansätze zur Modellierung kollaborativer learnflows entwickelt und es wird gezeigt wie sich entsprechende Lernprozessmodelle automatisch aus Protokolldateien von Lernsystemen erzeugen lassen.

1 Einleitung

Während sich die Repräsentation, Verarbeitung und Computerunterstützung von Geschäftsprozessen etabliert hat und methodisch gereift ist, werden hingegen auf dem verwandten Gebiet für Lehr- / Lernprozesse erst in den letzen Jahren verstärkte Anstrengungen unternommen. Aus diesem Anlass diskutierten wir in [1] die Gemeinsamkeiten, Unterscheidungsmerkmale und einen potentiellen Methodentransfer zwischen Geschäftsprozessen und Lernprozessen. An dieser Stelle entwickelten wir auch erste Ansätze zur Modellierung von Gruppenlernprozessen mit Hilfe von Petri-Netzen und der Generierung von Netzmodellen aus Protokollinformationen (logfiles) durch mining-Algorithmen.

Von besonderem Interesse ist dabei, wie kollaborative Arbeit bzw. Lernen geeignet repräsentiert werden können und welche Rollen bzw. Gruppenzusammensetzungen für einzelne Aktivitäten notwendig bzw. erwünscht sind. Dabei sind insbesondere die Spezifika von kollaborativen Lehr- / Lernprozessen gegenüber Geschäfts- prozessen zu berücksichtigen, was eine direkte Nutzung existierender Ansätze aus dem Bereich der Geschäftsprozessmodellierung (z.B. [2]) einschränkt bzw. Erweiterungen notwendig macht:

- Für den Geschäftsprozess ist die Durchführung des Prozesses und der damit verbundenen Aktivitäten ein Mittel zur Erreichung eines bestimmten Endprodukts, wobei die Qualität aber weniger die Beteiligung der einzelnen eingebundenen Akteure im Vordergrund steht. Bei Lernprozessen ist hingegen wesentlich, dass die Lernenden einen Lernprozess durchlaufen, bei dem einzelne Aktivitäten Lerngelegenheiten bieten; das Ergebnis des Prozesses ist - abgesehen von formalen Prüfungen - weniger wichtig als das (vollständige) Durchlaufen des Prozesses für die Teilnehmer. Daher sollten die einzelnen Akteure bei der Modellierung von Lernprozessen größere Berücksichtigung finden.
- Das Rollenkonzept im workflow engineering beruht i.A. auf der Verantwortlichkeit bzw. Kompetenz für eine bestimmte Menge von Aktivitäten, die nach anfänglicher Zuweisung von Rollen für konkrete Akteure festbleibt. Dynamische Einschränkungen der Aktivitätsbearbeitung (in etwa: derselbe Akteur, der das Angebot formuliert

soll auch den Vertrag abschließen) und spezielle Regeln zur Allokation von Akteuren zu Aktivitäten (in etwa: der Akteur mit einer geforderten Rolle, der den wenigsten weiteren Rollen zugeordnet ist, soll zugewiesen werden) sind in verschiedenen Ansätzen, z.B. RBAC (Role-Based Access Control), mit Zusatzkonstrukten explizit formulierbar. Im Gegensatz zu diesem starren Rollenkonzept werden in Lernprozessen Rollen häufig eingesetzt, um bestimmte Fertigkeiten einzuüben und im Laufe eines Lernprozesses werden Rollen dynamisch gewechselt bzw. erworben. Eine Erweiterung eines statischen Rollenmodells hin zu einem dynamischen, das in der Lage ist die Lernhistorie für Rollenfestlegungen heranzuziehen, ist folglich für Lernprozesse vorzunehmen.
– Einzelne Aktivitäten, gelegentlich auch der gesamte Lernprozess, können durch Gruppenarbeit, -diskussion usw. realisiert werden, wobei häufig in kollaborativen Ansätzen diese Gruppenphasen von hoher Bedeutung für die Lernerfahrung sind. Die Möglichkeit der Repräsentation von Gruppen, in der Gruppe notwendigen Rollen und gegebenenfalls dynamische Bildung / Umformung von Gruppen ist somit eine weitere Anforderung an Lehr- / Lernprozesse.

Im Folgenden werden wir aufbauend auf den Konzepten der Geschäftsprozessmodellierung aus [2] einen Prozessmodellierungsansatz präsentieren, der die Besonderheiten der Lehr- / Lernprozessmodellierung berücksichtigt. Neben der Anwendbarkeit des Ansatzes speziell für Lehr- / Lernprozesse, sehen wir auch eine Nutzbarkeit für Geschäftsprozesse, in denen Gruppenaktivitäten und dynamische Rollen wesentlich sind.

Einen ersten entsprechenden Modellierungsansatz haben wir schon in [1] skizziert. Wir haben vorgeschlagen Lernprozesse wie im Workflowbereich üblich mit Petrinetzen (oder entsprechenden Dialekten von Petrinetzen wie Aktivitätsdiagrammen) zu repräsentieren. Die Akteursallokation haben wir durch Zuweisung von benötigten Rollen zu Aktivitäten durchgeführt, wobei ein globaler Pool mit in Rollen eingeteilten Akteuren angenommen wird. Dabei haben wir bestehende Workflowkonzepte dadurch erweitert, dass sich die Rollen der Akteure bei der Durchführung von Aktivitäten verändern können.

Da die Akteure und insbesondere deren Rollenwechsel bei der Lernprozessmodellierung eine zentrale Rolle spielen, schlagen wir hier vor diesen Ansatz zu verfeinern, indem wir die dynamische Rollenbelegung der Akteure explizit durch ein Zustandsdiagramm modellieren (natürlich lassen sich hier auch hierarchische Rollenbeziehungen darstellen). Ein Akteur kann seine Rolle, i.e. seinen Zustand, ändern, wenn er eine Aktivität durchführt. Rollenwechsel finden also durch Synchronisation der Übergänge der Zustandsdiagramme mit Aktivitäten des Prozessmodells statt.

In einem zweiten Modellierungsvorschlag gehen wir noch einen Schritt weiter, indem wir den globalen Akteurspool auflösen und die Zustandsdiagramme, welche die Akteure repräsentieren, als Marken in den Kontrollfluss des Prozessmodells einbetten. Dadurch lässt sich insbesondere der Fortschritt der Akteure innerhalb des Lernprozesses durch ihre „Aufenthaltsorte“modellieren. Bei diesem Ansatz haben wir uns von den existierenden Ideen zur Modellierung mit „Netzen in Netzen“inspirieren lassen. Insbesondere gibt es Arbeiten (z.B. [3]) zur Modellierung von Multiagentensystemen, organisationsübergreifenden workflows und adaptiven workflows mit Objektnetzen.

Die zwei Modellierungsansätze fokussieren auf die Repräsentation dynamischer Rollen und berücksichtigen (Lern-) Gruppen nur implizit durch kollaborative Aktivitäten. Wir geben daher anschließend einen Ausblick auf Erweiterungen der zwei Ansätze zur expliziten Modellierung von Gruppen.

Zusätzlich zu den Modellierungsansätzen diskutieren wir die Möglichkeiten und das Vorgehen für eine automatisierte Synthese von solchen Modellen aus realen Protokollinstanzen als Ansatz zum collaboration flow mining. Hierbei erweitern wir die in [1] als Analogie zum workflow mining [4] vorgestellte Idee des learnflow mining. Während sich dieses aber noch auf das Auffinden von Kontrollflussstrukturen beschränkte, stellt sich in dem hier betrachteten Rahmen die weitere Herausforderung Informationen über dynamische Rollen und entsprechende Kollaborationsregeln aus den Protokollinstanzen zu gewinnen. Ein verwandter Ansatz aus dem Bereich der Geschäftsprozessmodellierung ist das auf starre Rollen und Organisationseinheiten beschränkte organizational mining [5].

In Kapitel 2 stellen wir die neuen Modellierungsansätze an einem Beispiel, welches schon in [1] verwendet wurde, vor. Mit diesem Beispiel erklären wir die zentralen Ideen des collaboration flow mining in Kapitel 3.

2 Modellierungsansätze

Als Beispiel betrachten wir im Folgenden Gruppen von je drei Schülern, die unterstützt durch das Tool FreeStyler (www.collide.info) lernen, wie sich verschiedene Faktoren (z.B. Lichtverhältnisse, CO2-Gehalt, ...) auf das Wachstum von Pflanzen auswirken (für Details vgl. [1]). FreeStyler stellt hierfür verschiedene Registerkarten zur Verfügung, auf denen Fragen formuliert, einfache Modelle gezeichnet oder Daten aus einem Simulationsprogramm importiert werden können. Die Menge der Registerkarten ist somit in unserem Beispiel die Menge der unterstützten Aktivitäten (Ei = **Ei**nführung in die Thematik, Fr = Erarbeitung der wissenschaftlichen **Fr**agestellung, Pl = **Pl**anung, Mo = **Mo**dellierung der Beziehungen zwischen den Faktoren, Hy = Aufstellung einer Forschungs**hy**pothese, E1 & E2 = **E**xperimente zur Hypothesenprüfung, Da = Studium existierender **Da**ten, An = **An**alyse der Daten mitsamt Überprüfung der Hypothese, Pr = **Pr**äsentation der Forschungsergebnisse). Einige dieser Lernaktivitäten (Ei, Hy, An jeweils mit allen drei Schülern und Pl, Mo, Pr jeweils mit zwei Schülern) erfordern bestimmte Arten von Kollaboration zwischen den Schülern. Ein Lernprozessmodell soll nun modellieren in welcher Reihenfolge und von wem die Registerkarten bearbeitet werden sollen, wobei Letzteres von den Rollen abhängt, die die Schüler innerhalb der Gruppe einnehmen.

Erstes Modell: Das Lernprozessmodell in Abbildung 1 stellt den Prozessaspekt des Beispiels dar, der durch einen Zustandsautomaten, der ein Rollendiagramm repräsentiert, in Abbildung 2 ergänzt wird. Eine Aktivität im Prozessmodell kann nur dann durchgeführt werden, falls die an der Transition angeschriebene Anzahl von Rollen im globalen Akteurspool vorhanden ist: beispielsweise erfordert die kollaborative Aktivität Planung (Pl) 2 Akteure in der Rolle Schüler. Beim Schalten der Transition werden für die betreffenden Akteure Rollenveränderungen vorgenommen, die im Automatenmodell einem Zustandswechsel mit dem Transitionsnamen als Eingabezeichen entsprechen; in unserem Beispiel gehen also durch die Planungsaktivität beide Schüler in die Rolle Modellierer über. Als Konvention zur Vereinfachung des Zustandsautomaten setzen wir voraus, dass bei Aktionen, die im Diagramm nicht explizit einen Rollenwechsel verursachen, die bisherige Rolle erhalten bleibt: Die Aktivität Modellierung (Mo) führt für einen Akteur, der sich in der Rolle Modellierer befindet, keinen Rollenwechsel herbei und kann deshalb im Diagramm entfallen.

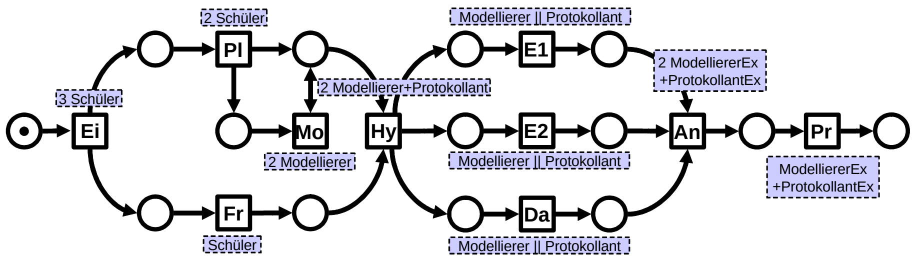


Abbildung 1. Erstes Modell: Prozessfluss repräsentiert als Stellen-Transitions-Petrinetz mit Rollenbeschriftungen

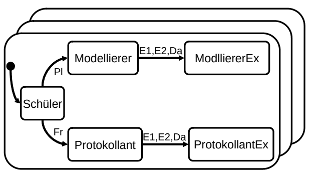


Abbildung 2. Erstes Modell: Rollendiagramm repräsentiert als Zustandsautomat

Der Lernfortschritt und die Lernhistorie einzelner Akteure werden bei diesem Modellierungsansatz in das Rollendiagramm einkodiert: Beispielsweise wird durch Ausführen von Experiment1 (E1) ein Lernfortschritt durch einen Rollenwechsel erreicht, der ein Ausführen von Experiment2 (E2) und Daten (Da) verhindert. Diese Repräsentation fortschrittsabhängiger Aspekte wird im folgenden alternativen Modellierungsansatz eleganter adressiert.

Zweites Modell: Die Abbildungen 3 und 4 stellen in ähnlicher Form den Prozessaspekt und den Rollenaspekt im alternativen Modellierungsansatz dar. Hierbei wird für die Prozessrepräsentation ein hierarchisches Petrinetz verwendet, bei dem die Marken selbst Rollenautomaten sind, die jeweils einen Akteur repräsentieren, dessen Rolle durch seinen aktuellen Zustand dargestellt wird und dessen Fortschritt im Lernprozess durch die Platzierung im Netz erkennbar ist. In ähnlicher Weise wie im ersten Modellierungsansatz wird eine Aktivität durchgeführt, sofern mindestens soviele Akteure in Rollen, wie an der Transition angeschrieben, vorhanden sind, allerdings müssen diese Akteure nun lokal im Vorbereich der Transition vorhanden sein. Die Kantengewichte geben hierbei an wieviele Akteure aus welcher Stelle benötigt werden und in welche Stellen wieviele Akteure fortschreiten (dabei muss die Anzahl der Akteure für jede Transition erhalten bleiben: „Männchenerhaltungssatz“). Für den Kontrollfluss darf das Modell zusätzlich auch Stellen mit schwarzen Marken enthalten (wie zwischen den Transitionen Planung und Modellierung).

Durch die Lokalisation jedes Akteurs als Marke im Prozessnetz sind fortschrittsabhängige Aspekte bereits explizit im Petrinetz repräsentiert. Deshalb ist nun beispielsweise ein Rollenwechsel beim Ausführen von Experiment1 (E1) nicht mehr nötig, was

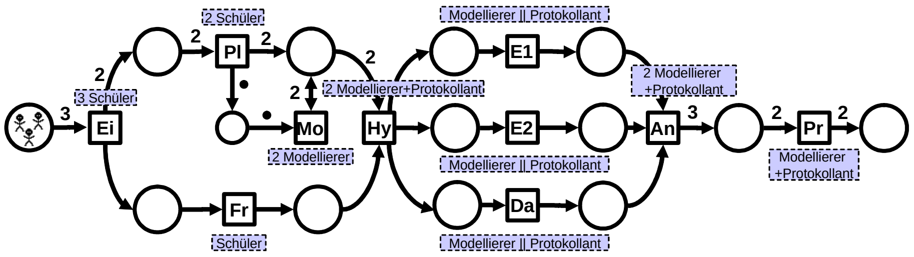

2 Schüler
Pl
Modellierer || Protokollant
E1
2 Modellierer+Protokollant
2 Modellierer +Protokollant
3 Schüler
Ei
Mo
Hy
E2
An
Pr
2 Modellierer
Modellierer || Protokollant
Modellierer +Protokollant
Fr
Da
Schüler
Modellierer || Protokollant

Abbildung 3. Zweites Modell: Prozessfluss repräsentiert als hierarchisches Petrinetz

sich im Rollenautomaten von Abbildung 4 gegenüber demjenigen in Abbildung 2 widerspiegelt. Übergänge im Rollenautomaten werden wie bereits im ersten Ansatz durch feuernde Transitionen als Eingabezeichen ausgelöst.

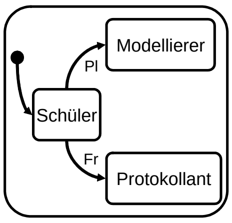


Abbildung 4. Zweites Modell: Rollendiagramm repräsentiert als Zustandsautomat

Zusammenfassend lässt sich festhalten, dass beide vorgeschlagenen Modellierungsansätze dynamische Rollen, kollaborative Aktivitäten und den Fortschritt im Lernprozess geeignet repräsentieren können. Allerdings unterscheiden sich die beiden Ansätze bezüglich der Eindeutigkeit und semantischen Klarheit der Modellierung, wobei kein Ansatz dem anderen eindeutig überlegen ist: Der erste Ansatz trennt die Modellierung von Prozess und Rollen weitgehend voneinander und verwendet einfache anonyme Marken im Petrinetz, wohingegen der zweite Ansatz komplexe Marken verwendet, die jeweils einen Rollenautomaten mit einem aktuellen (Rollen-)Zustand repräsentieren, was zudem gegebenfalls die optische Lesbarkeit des graphischen Modells erschwert (vergleiche Abbildungen 1 und 3). Andererseits repräsentiert im zweiten Ansatz der Rollenautomat ausschließlich semantisch klar definierte Rollen und Übergänge, was im ersten Ansatz eventuell durch fortschrittsabhängige Pseudo-Rollen modelliert werden muss, falls die Lernhistorie berücksichtigt werden soll. Dies zeigt sich klar im Vergleich der Abbildungen 4 und 2, in dem der zweite Ansatz wesentlich klarer die notwendigen Rollen darstellt.

Explizite Gruppenmodellierung: Gruppen sind in den zwei Modellen implizit über kollaborative Aktivitäten berücksichtigt. Zur expliziten Gruppenmodellierung bieten die zwei Modellierungsansätze zwei Möglichkeiten. Zum einen lassen sich Gruppen durch verschiedene Prozessinstanzen repräsentieren. Hier fehlen allerdings noch Konzepte zur Modellierung von Abhängigkeiten zwischen Prozessinstanzen um dynamische Gruppenumformierungen darstellen zu können. Andererseits lassen sich Gruppen auch analog zu Rollen explizit in den Zustandsdiagrammen der Akteure modellieren. Bei Gruppen ist es aber im Gegensatz zu Rollen wichtig die Gesamtgruppendynamik darzustellen, welche sich dann nur implizit über die Gruppenzugehörigkeiten der einzelnen Akteure ergibt. Daher wären hier Erweiterungen der Modellierungsansätze interessant, welche zu jedem Zeitpunkt explizit die Lerngruppen darstellen, z.B. geeignete Gruppierungen der Zustandsdiagramme im Akteurspool.

3 Collaboration Flow Mining

In diesem Kapitel zeigen wir einen Ansatz zum Auffinden eines Lernprozessmodells der ersten Form aus Protokollinformationen, d.h. es soll ein Lernprozessmodell erzeugt werden, welches entweder das aufgezeichnete (dem Dozenten evtl. unbekannte) Lernverhalten der Schüler oder durch entsprechendes Filtern auch das erwünschte Lernverhalten der Schüler wiedergibt. Wenn ein Akteur unterstützt von einem Informationssystem eine Aktivität ausführt, entstehen Ereignisse und durch Aufzeichnung der Ereignisse Protokollinstanzen. Jede Aufzeichnung eines Ereignisses soll Informationen über den zugehörigen Prozess, die zugehörige Prozessinstanz, den Name der Aktivität, den Zeitpunkt ihrer Ausführung und die ausführenden Akteure (mehrere bei kollaborativen Aktivitäten – u.U. müssen diese noch aus mehreren Ereignissen mit dem selben Zeitstempel zusammengesetzt werden) enthalten. Die Ereignisse werden erst nach Prozess und Prozessinstanz und innerhalb einer Prozessinstanz in der Reihenfolge ihres Ausführungszeitpunktes geordnet. Damit ergibt sich für jede Prozessinstanz eine Folge von Aktivitäten mit zugeordneten Akteuren. Diese Abläufe lassen sich als Ausgangspunkt für verschiedene Mining Algorithmen verwenden, um Prozessmodelle zu erzeugen. Derart erzeugte Prozessmodelle können zur Verifikation, Analyse oder zur Steuerung des operationalen Prozesses durch ein Informationssystem benutzt werden.

Das Tool Freestyler zeichnet die Aktivitäten der Schüler als Ereignisse auf. Abbildung 5 zeigt einen Auszug aus einem Beispielprotokoll von Freestyler für den betrachteten Lernprozess. Darunter zeigt Abbildung 5 einen sich aus dem Beispielprotokoll ergebenden Ablauf von Aktivitäten mit zugeordneten Akteuren. Der Dozent hat die Möglichkeit die Menge der Lernabläufe zu filtern, indem er nach gewissen Kriterien (z.B. nachträglich gemessener Lernerfolg) unerwünschte Lernabläufe entfernt und speziell erwünschte Lernabläufe zusätzlich vorgibt. In diesem Fall wird dann durch mining ein Modell für einen erwünschten Lernprozess erzeugt, während ohne Filterung durch den Dozenten ein Modell für den tatsächlich von den Schülern durchgeführten Lernprozess generiert wird.

Protokoll-Datei

Prozess	Prozessinstanz	Aktion	Schüler	Zeit
Photosynthese	Gruppe A	Einführung	Andi, Basti, Robin	10:03:12
Photosynthese	Gruppe A	Fragestellung	Robin	10:06:43
Photosynthese	Gruppe B	Einführung	Bert, Caro, Hans	10:07:33
...	...	...	...	...

Lernabläufe

Gruppe A	(Einführung; Andi,Basti,Robin), (Fragestellung; Robin), (Planung; Andi,Basti), (Modellierung; Andi,Basti), (Hypothese; Andi,Basti,Robin), (Experiment1; Andi), (Experiment2; Robin), (Daten; Basti), (Analyse; Andi,Basti,Robin), (Präsentation; Andi,Robin)

...

Projektion der Lernabläufe auf einzelne Schüler

Einführung, Planung, Modellierung, Hypothese, Experiment1, Analyse, Präsentation
Einführung, Planung, Modellierung, Hypothese, Daten, Analyse
Einführung, Fragestellung, Hypothese, Experiment2, Analyse, Präsentation
...

Lernabläufe für Rollenannotationen

Gruppe A	(Einführung; -,-,-), (Fragestellung; Ei), (Planung; Ei,Ei), (Modellierung; EiPl,EiPl), (Hypothese; EiPlMo,EiPlMo,EiFr), (Experiment1; EiPlMoHy), (Experiment2; EiFrHy), (Daten; EiPlMoHy), (Analyse; EiPlMoHyE1,EiPlMoHyDa,EiFrHyE2), (Präsentation; EiPlMoHyE1An,EiFrHyE2An)

...

Abbildung 5. Beispielprotokoll.

Wir nehmen im Folgenden ein vollständiges Protokoll für den im letzten Kapitel modellierten Lernprozess an, d.h. wir betrachten die Menge aller bzgl. dieses Lernprozesses möglichen Lernabläufe. Vernachlässigt man in dieser Menge von Lernabläufen die Akteure, so lässt sich aus der resultierenden Menge von Aktivitätsfolgen mit bekannten mining-Verfahren [1, 4] automatisch ein Modell für den Kontrollfluss des Lernprozesses erzeugen. Beispielsweise erzeugt ein in VipTool implementiertes mining-Verfahren das in Abbildung 1 gezeigte Petrinetzmodell noch ohne Rollenannotationen. Um nun zusätzlich Rollenannotationen und ein Zustandsdiagramm für die dynamischen Rollen der Schüler zu generieren schlagen wir im Weiteren ein spezielles mining-Verfahren vor.

Zuerst betrachten wir für jeden Lernablauf und jeden an dem Ablauf beteiligten Schüler die Folge von Aktivitäten, die der Schüler in dem Ablauf durchführt. Abbildung 5 illustriert diese Projektionen der Lernabläufe auf die Lernenden für den betrachteten Ablauf. All diese Folgen von Aktivitäten werden nun in einem deterministischen Zustandsdiagramm in Baumform zusammengefasst. Die Zustände sind dann durch die in der Vergangenheit durchgeführten Aktivitäten eindeutig bestimmt und werden entsprechend benannt. Für unser vollständiges Protokoll ergibt sich das Diagramm in Abbildung 6, welches schon ein erstes Rollenmodell darstellt. Jede Rolle ergibt sich also durch die in einer bestimmten Reihenfolge bisher durchgeführten Aktivitäten. Um diese Rollen konsistent als Beschriftungen im Petrinetz zu verwenden, muss in jedem Lernablauf jeder Akteursname durch die Rolle, die den Aktivitäten entspricht, welche der Akteur in der Vergangenheit der betrachteten Aktivität in dem Ablauf durchgeführt hat, ersetzt werden (siehe Abbildung 5 unten). Die Rollenbeschriftung einer Aktivität im Petrinetz ergibt sich nun aus allen Rollen bzw. bei kollaborativen Aktivitäten Rollenkombinationen, die in irgendeinem Lernablauf zusammen mit der Aktivität vorkommen.

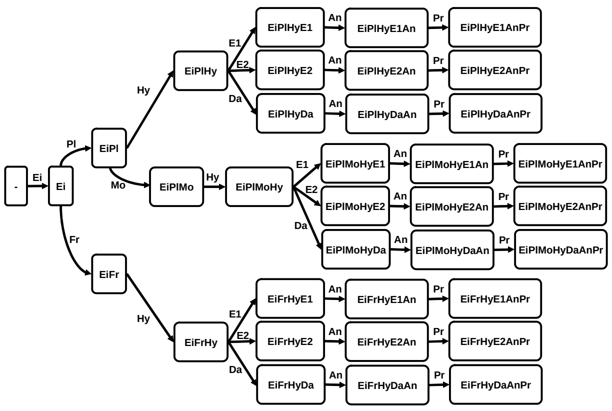


Abbildung 6. Rollendiagramm mit feinster Granularität.

Für den betrachteten Modellierungsansatz lässt sich zeigen, dass aus einer vollständigen Ablaufmenge eines Modells mit diesem mining-Verfahren immer ein verhaltensäquivalentes Modell erzeugt wird. Insbesondere ist das aus dem Beispiel generierte Modell äquivalent zu dem Lernprozessmodell aus Abbildung 1. Allerdings entsteht hier eine sehr feine Rollenaufteilung, die auf der vollständigen Aktivitätshistorie eines Akteurs basiert.

Im Weiteren ist nun das Ziel das Rollenmodell zu vereinfachen, indem bestimmte Rollen zusammengefasst werden. Hierzu haben wir die folgenden Vereinfachungsregeln für Rollendiagramme entwickelt. Die Petrinetzbeschriftungen müssen jeweils konsistent abgeändert werden.

– Rollenübergänge, durch Aktionen an denen alle in einer Prozessinstanz vorkommenden Akteure beteiligt sind, können weggelassen werden.
– Rollen, die dieselben Folgerollen (bzw. keine Folgerollen) mit denselben Übergangsaktivitäten haben (ersichtlich aus Abbildung 6), können zusammengefasst werden, falls sie für jede ausgehende Aktivität in der Gesamtheit der Lernabläufe genau mit denselben Rollen zusammen vorkommen (ersichtlich aus Abbildung 5 unten). Dabei dürfen Übergänge zwischen den zu verschmelzenden Rollen vernachlässigt werden.
– Als letzte Reduktion können einmalig Rollen ohne Ausgänge entfernt werden.

Aus Platzgründen können wir diese Regeln hier nicht näher erläutern und illustrieren. Es lässt sich zeigen, dass diese Regeln unter der Vollständigkeitsannahme des Protokolls wieder zu einem äquivalenten Modell führen. In unserem Beispiel ergibt sich ein Rollendiagramm, welches isomorph zu dem in Abbildung 2 ist. Damit lässt sich bis auf die Rollennamen, welche in einem Protokoll aber auch nicht auftauchen, das ursprüngliche Lernprozessmodell aus einem vollständigen Protokoll reproduzieren. Die Rollennamen müssten daher nachträglich vom Dozenten vergeben werden.

Typischerweise muss davon ausgegangen werden, dass nicht alle möglichen Abläufe eines Prozesses aufgezeichnet werden und damit Protokolle unvollständig sind. Für solche Protokolle sind Anpassungen der mining-Verfahren nötig. Hier sind Heuristiken interessant um auf im Protokoll „fehlende“Abläufe zu schließen und diese in das Prozessmodell zu integrieren. Für die Kontrollflussperspektive wurden hierzu im Bereich des process mining etliche Verfahren vorgeschlagen. Die Rollendiagramme betreffend sehen wir Möglichkeiten zur Anwendung von Verfahren der strukturellen Äquivalenz und der verallgemeinerten Blockmodellierung.

Literatur

1. Bergenthum, R., Desel, J., Harrer, A., Mauser, S.: Learnflow mining. In: DeLFI, LNI 132, GI (2008) 269–280
2. Aalst, W., Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
3. Aalst, W., Moldt, D., Wienberg, F., Valk, R.: Enacting interorganizational workflows using nets in nets. In: Workflow Management Conference. (1999) 117–136
4. Aalst, W.: Finding Structure in Unstructured Processes: The Case for Process Mining. In: ACSD 2007, IEEE (2007) 3–12
5. Song, M., Aalst, W.: Towards comprehensive support for organizational mining. Decision Support Systems **46**(1) (2008) 300–317

Net Agents for Activity Handling in a WFMS

Kolja Markwardt, Daniel Moldt, and Thomas Wagner

University of Hamburg - Department of Informatics
http://www.informatik.uni-hamburg.de/TGI

Abstract. Workflow Management Systems (WFMS) are used to organize work processes between different people within an organization or between organizations. In this paper we will describe an agent-based WFMS, built with Petri nets, to utilize the formal soundness of Petri nets and the flexibility of multi-agent systems to enhance the usefulness of a WFMS. The focus of this paper lies in the way activities are handled in the WFMS. We will first discuss the way this works in the system as of now. Then we will go on and describe a way to use Activity Agents to add a further flexibility to the activity handling of the system. These Activity Agents will be responsible for providing the functionality, including materials and user interface associated with an activity.

Keywords: Activity Agents, Reference Nets, Workflow Management Systems

1 Introduction

Workflow Management Systems (WFMS) are used regularly within companies. In research and practice the question of coupling the different WFMS of cooperating organizations arises, leading to inter-organizational WFMS. Agent-based WFMS are one answer in this field. In [6, 7] we proposed an agent-based WFMS and a process infrastructure for agent systems. The general model is quite elaborated, while the implementation has not been discussed deeply. Following the proposal of [5] the ideas of tools and materials (see [10] for this general approach) is introduced to multi-agent systems (MAS). Adding the concept of tools and materials to our Mulan reference model adds further structuring capabilities for the modeller.

In this contribution we will explain in Section 2 the technological basics of our WFMS. Section 3 explains our current agent-based WFMS. Then Section 4 provides the information about the tool and material approach and its adaptation to our agent-based approach. On top that the central idea, the Activity Agent is introduced. How to implement the afore mentioned Activity Agent concept is described in Section 5. The paper concludes with a short summary and outlook in Section 6.

2 Basics

The technological foundation for this contribution are the Mulan and Capa agent architectures. Mulan stands for **Mul**ti-**a**gent **n**ets, which is also the main idea behind it. It was described in [8]. Every element of Mulan, including agent behavior, agent knowledge and agents themselves, is modelled using the reference net formalism introduced in [4].

Capa (**C**oncurrent **A**gent **P**latform **A**rchitecture) is an extension of Mulan, which was introduced in [2]. Its main focus is on communication between platforms and making the Mulan principles fully compliant with the FIPA standards. The reference net tool Renew serves as both development and runtime environment. A description of Renew can be found in [4].

Within the WFMS workflows are modelled using a variation of the workflow nets described in [9]. This variation of workflow nets uses the task transition introduced in [3]. A task transitions actually represents three regular transitions. The three transitions model the request of a work item and the cancellation or confirmation of an activity.

3 An Agent-Based WFMS

In its current version the system supports basic WFMS functionality. It provides the means to administrate the system, add and edit workflow definitions and instantiate and execute workflow instances.

The functionality of the WFMS is provided by a number of different agent types. The system's core is made up of a trio of agents, which provide the internal function of the WFMS. These agents are the Workflow Engine agent (WFEngine agent), the Workflow Enactment Service agent (WFES agent) and the Workitem Dispatcher agent (WiDi agent). The WFEngine agent is responsible for firing the internal transitions of the tasks and for initiating the workflow instances. The WiDi agent distributes work items and activities to the users, if they are eligible for them. The WFES agent is located between the other two agents. It manages different workflow instances on the workflow engine. These three agents interact with each other to organize the functionality to provide work items and activities to the user. For each user a user agent exists, which manages the interactions between the WFMS and the user's GUI. Its main responsibility lies in invoking interactions upon actions of the user within the GUI and in initiating display updates within the GUI. The other agent types offer the functionality to manage and authenticate logged in users as well as access to the database.

Currently the WFMS supports two kinds of tasks. Simple tasks can only be accepted and then be marked as completed or canceled. They represent actions, which have to be completed outside of the functionality the WFMS offers. The other kind of tasks is form tasks. When a form task is assigned to a user, a form window is opened, in which the user enters the necessary information. Forms can consist of an arbitrary number of labels, text boxes, check boxes and radio buttons. When the form task is completed the data entered into a form is read into the system and can be used in later form tasks.

The subject of this contribution concerns the the way activities are handled within the system. Because of this it is important to give a description of the way activities are assigned in the current version.

While workflow instances are active within the system, eligible users can request the work items, which are currently activated in the workflow nets. When a user wishes to request a work item he initiates the associated interaction. If this interaction is successful the WFEngine agent fires the internal request transition of the task and creates the activity.

When any changes in a workflow net occur a decision component (DC) net, a special part of the WFEngine agent, is automatically informed. This listener DC net contains a cycle, which is responsible for handling reported changes in the set of activities and is always repeated when a new change occurs. The cycle checks which activities have changed and need to be updated. The cycle ends with the initiation of the UpdateActivityList interaction. The UpdateActivityList updates the internal lists of the WFEngine agent, the WFES agent and the WiDi agent.

After the UpdateActivityList interaction has been completed, the OfferActivityList interaction is started, in which the WiDi agent informs all user agents connected to him about the previously updated status of their activities. These updated activities are then displayed for the user and can be executed.

4 Tool- and Activity-Agents

In this paper we propose a new way of handling activities in the WFMS by using a special kind of Tool Agents, called Activity Agents. We will first describe the notion of Tool Agents and how they can be used to build flexible tool-based applications. Then we will describe the special incarnation of Activity Agents and the way they will be integrated into the WFMS architecture.

4.1 Tool Agents

Tool Agents are a way to use multi-agent systems to build tools for supporting individual users work as well as collaborative efforts. This follows the notions of the tools and materials approach [10], applied to multi-agent systems to address distributed workplaces.

Overview The main idea about the tool agent concept is that each user controls a user agent (UA), which can be enhanced by different tool agents (TA) as shown in [5]. The user agent provides basic functionality like a standard user interface and the possibility to load new tool agents. Those tool agents can then plug into the user agents UI with their own UI parts, offering their functionality to the user. By choosing the specific set of tool agents, the user can tailor his work environment to his specific needs.

Material agents (MA) are used to represent and encapsulate the materials or work objects that are currently worked on, like an insurance claim or a text file. Materials are manipulated by tools and can be created, deleted and moved between workplaces. Tools and materials populate the workspace of the user.

An agent called the Tool Factory is used to manage the different types of tool agents known to the system. It is called by the user agent to create a ne instance of a tool agent to use. Figure 1 shows how these agents work together.

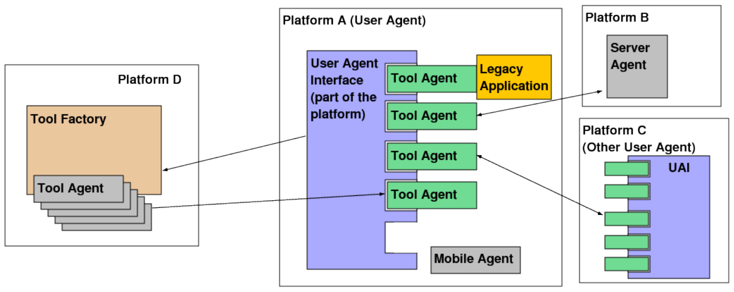


Fig. 1. User and Tool agents

4.2 Activity Agents

The definition of a workflow can contain any number of different types of tasks for a user to perform. In the current systems, all these tasks have to be defined in advance and the user needs to know how to handle them. It seems clear that in this way the user can be a real bottleneck in the deployment of new workflows, if these contain new types of tasks.

Tool Agents already provide a way to enhance the functionality of a User Agent. Therefore, an adaptation of Tool Agents for WFMS will provide a way to handle this problem, this adaptation is called an **Activity Agent**.

Like any Tool Agent it can be used to manipulate materials, but in this case the material and the context for its manipulation is provided by the workflow. An Activity Agent is not requested by the User to perform some task but it is rather assigned to it by the workflow engine to handle an activity. Once it is connected to the User however, it functions like another Tool Agent. It only needs a way to determine that the work on the activity is done, so that it can return the material and feedback on activity completion back to the workflow engine, for example a “finish” or “abort “ button in the GUI.

4.3 The Activity Agent in the Activity handling process

Activity Agents represent activities within the running WFMS. When a user successfully request a work item available to him, the system will automatically start a new Activity Agent. This Activity Agent will be responsible for this activity alone, and will only be active while the activity is being executed by the user. During the execution of the activity the user will exchange information with the Activity Agent, in order to work on the activity. When the activity has been finished, the Activity Agent will transmit the relevant data back to the workflow engine and terminate.

5 Design and Implementation

In this section we will describe our proposed way to implement the Activity Agent. The Activity Agent will be started after the listener DC net of the WFEngine has detected that a new activity has been created. Before the Update-ActivityList interaction is started the listener will check if the activity is flagged as an Activity Agent activity. If the activity is to be executed by an Activity Agent, a new DC net is entered. The purpose of this new DC will be to start the new Activity Agent and add the agent's information to the activity, so that the executor's user agent knows how to interact with the Activity Agent. After this is done, the listener DC net can continue and start the UpdateActivityList interaction. Within The UpdateActivityList and OfferActivityList interactions only the internal lists have to be modified, in order to incorporate the added information.

While the Update- and OfferActivityList will not have to be changed much, the handling of activities in general and within the user GUI require changes. Concerning the handling of activities changes have to be made, because in the proposed version the actual handling of the activity will be done by the Activity Agent. In the current version the activities are manipulated within the GUI and then passed to the user agent, who, upon completion of the activity, sends them to the WFEngine agent. In the new version the user agent will not directly communicate with the WFMS's core in this matter, but will only communicate with the Activity Agent in order to manipulate materials involved in the activity (e.g. forms). When the user has finished his work on the activity he will inform the Activity Agent, who then informs the WFMS's core.

The reason for changes to the GUI is, that in the current version tasks are simply displayed in a list (simple tasks) or a generic form window is opened (form tasks). In both cases, the interface to complete the activity is embedded into the general GUI. Since the Activity Agent is designed to be able to offer arbitrary functionality with a possibly specialized GUI, this GUI has to be offered by the agent itself, because otherwise the main user GUI has to be changed and enhanced, whenever a new type of task is added. In order to support this, the GUI has to be changed so that selecting an activity will contact the Activity Agent who will in turn invoke his own GUI.

6 Summary and Outlook

We have proposed in this paper a way of enhancing an Agent-based WFMS with flexible activity-handling procedures. Specialized agents will be used to plug into the users' system and provide them with the functionality needed to perform their tasks within the workflow process. The underlying Petri nets allow first of all an appropriate modelling of the system and its processes. In addition the agents and their behaviour become a well defined semantics. Following our Paose approach (see [1]) models are continuously transformed, so that they can be executed. Reference nets support this directly. Introducing the tool and material ideas into our multi-agent systems provides us with a highly expressible modelling basis. Activity agents directly rely on this. The flexibility introduced into the agent-based WFMS by the addition metaphor of the tool allows to introduce new activities much easier than the former way of explicitly defining all necessary parts redundantly.

Activity agents will directly be used in our next version of our distributed and no longer centralized agent-based WFMS. There it will take care in a quite generic way of activities of the workflows, the core of workflows in general.

References

1. Lawrence Cabac, Till Dörges, Michael Duvigneau, Christine Reese, and Matthias Wester-Ebbinghaus. Application development with Mulan. In Daniel Moldt, Fabrice Kordon, Kees van Hee, José-Manuel Colom, and Rémi Bastide, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 145–159, Siedlce, Poland, June 2007. Akademia Podlaska.
2. Michael Duvigneau. Bereitstellung einer agentenplattform für petrinetzbasierte agenten. Diploma thesis, University of Hamburg, Department of Computer Science, December 2002.
3. Thomas Jacob. Implementierung einer sicheren und rollenbasierten workflowmanagement-komponente für ein petrinetzwerkzeug. Diploma thesis, University of Hamburg, Department of Computer Science, 2002.
4. Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.
5. Kolja Lehmann and Vanessa Markwardt. Proposal of an agent-based system for distributed software development. In Daniel Moldt, editor, *Third Workshop on Modelling of Objects, Components and Agents (MOCA 2004)*, pages 65–70, Aarhus, Denmark, October 2004.
6. Christine Reese, Jan Ortmann, Daniel Moldt, Sven Offermann, Kolja Lehmann, and Timo Carl. Fragmented workflows supported by an agent based architecture. In Manuel Kolp, Paolo Bresciani, Brian Henderson-Sellers, and Michael Winikoff, editors, *Agent-Oriented Information Systems III 7th International Bi-Conference Workshop, AOIS 2005, Utrecht, Netherlands, July 26, 2005, and Klagenfurt, Austria, October 27, 2005, Revised Selected Papers*, volume 3529 of *Lecture Notes in Computer Science*, pages 200–215. Springer-Verlag, 2006.
7. Christine Reese, Matthias Wester-Ebbinghaus, Till Dörges, Lawrence Cabac, and Daniel Moldt. Introducing a process infrastructure for agent systems. In Mehdi Dastani, Amal El Fallah, João Leite, and Paolo Torroni, editors, *LADS'007 Languages, Methodologies and Development Tools for Multi-Agent Systems*, volume 5118 of *Lecture Notes in Artificial Intelligence*, pages 225–242, 2008. Revised Selected and Invited Papers.
8. Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2004.
9. Wil M.P. van der Aalst. Verification of workflow nets. *Lecture Notes in Computer Science*, 1248/1997:407–426, 1997. Application and Theory of Petri Nets 1997.
10. Heinz Züllighoven. *Object-Oriented Construction Handbook*. dpunkt Verlag, 2005.

Parametric Petri Net Model for Ethernet Performance and Qos Evaluation

Dmitry A. Zaitsev[1] and Tatiana R. Shmeleva[2]

Department of Communication Networks,
Odessa National Academy of Telecommunications,
Kovalska, 1, Odessa 65029, Ukraine
Web[1]: http://www.geocities.com/zsoftua, E-mail[2]: tishtri@rambler.ru

Abstract. Parametric model of switched Ethernet in the form of a colored Petri net is presented. The model is invariant regarding the structure of the network; it has fixed number of nodes for any given tree-like network. Special measuring fragments, which accomplish the model, provide the evaluation of the network throughput and the frame delivery time directly in the process of simulation. The anomaly of Ethernet switches' mutual blocking has been revealed.

Keywords: switched Ethernet, colored Petri net, parametric model, delivery time, throughput, mutual blocking.

1 Introduction

At present Ethernet technology dominates the sector of local area networks. Moreover, 1Gb and 10Gb standards allow positioning Ethernet as a universal networking technology, because providers widely apply «Ethernet over DWDM» solutions in backbone networks. Design of effective local and backbone networks requires reliable estimations of throughput and the quality of service. Recently the model driven development of telecommunication networks and devices becomes prospective. It is based on express-evaluations of characteristics obtained in the shortest time for new project decisions that determine the relevance of the present research.

Colored Petri Nets [1] and CPN Tools [2] are successfully used for modeling Ethernet [3-5], TCP/IP and MPLS [6], wireless Bluetooth [7] networks. Colored Petri Nets allow not only the modeling of telecommunication networks, but also the estimation of their characteristics via special measuring fragments [4,8] during the process of simulation.

The mentioned papers are based mainly on the modular approach to the telecommunication networks models construction: a model of a network is composed of DTE (workstation, server) and DCE (switch, router) submodels, which were built earlier. The essential disadvantages of this approach are the following: the necessity of the model rebuilding for each new structural scheme of the network, great number of used Petri net elements that considerably delay the processes of models construction and analysis.

The parametric model of switched Ethernet presented in [5] has a fixed structure for an arbitrary tree-like network; its elements are switches, workstations and servers. The definite structure of a network is an input in the form of packed matrices as the marking of corresponding Petri net places. However, in [5] only the principles of a parametric Petri models construction were studied and the questions about the evaluation of the modeled networks characteristics were not considered.

The goal of the present work is constructing the measuring fragments for parametric model of switched Ethernet for the evaluation of throughput (traffic), the quality of service (frame delivery time), the size of the switches internal buffers. Moreover, for the confirmation of the built models adequacy, the technique for the mentioned characteristics measuring on real-life networks was developed.

2 Parametric Model of Switched Ethernet

The model presented in [5] was refined in the following way: the expressions in the transitions guards were simplified via variables superposition; the limitations of the switches internal buffer size were added; the places, which describe the dump of frames for the following calculation of characteristics, were added.

The model is represented in Fig. 1; the declarations of colors (**color**), variables (**var**) and functions (**fun**) used in the model and measuring fragments are represented in Fig. 2. The peculiarity of the parametric model is the special tags added to frames, which contain switch and port numbers and provide the frames reentrance. The model has a fixed structure and contains 14 places and 8 transitions of Petri net for an arbitrary tree-like structure. The components of the model are switches, workstations and servers. The left part of Petri net models all the Ethernet switches (the names of the elements names do not have any suffix), the right upper part – all the workstations (the names of the elements have **WS** suffix), the left lower part – all the servers (the names of the elements have **S** suffix); the pair of places **inPorts, outPorts** model all segments. Names "in/out" are chosen with respect to the switches; they model the full-duplex mode of work. Additional places **received, sent** model the frames dump by DTE, places **inSW, outSW** model the frames dump by switches. Additional names **rcvd, snd, inSWITCH, outSWITCH** are used for the connection with model pages, which calculate characteristics described in the next section.

A definite structure of modeled network is specified by the marking of places **swtab, SwichLink, Attach.** The place **swtab** contains the switching tables of all the switches. The switching tables are represented by corteges **swi** (destination address, port, switch). The place **SwichLink** describes the connections of switches (uplinks), which are represented by corteges **swl** (switch1, port1, switch2, port2). The place **Attach** describes the DTE connection, which is represented by corteges **swi** (address, port, switch). In Fig. 1 the marking of the places corresponds to the Railway dispatcher center LAN represented in Fig. 3.

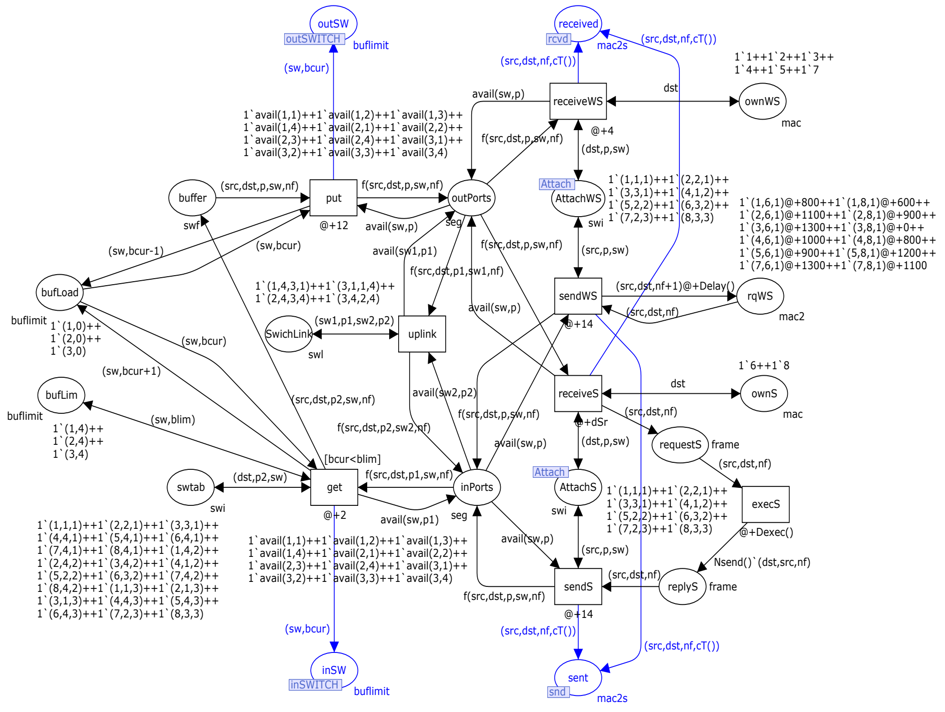


Fig. 1. Parametric model of switched network

Moreover, the model contains the following parameters: workstations addresses **ownWS,** servers addresses **ownS**, matrix of workstations requests to servers **rqWS**, switch buffer size limit **bufLim**. Random functions **Delay(), Dexec(), Nsend()** define the periodicity of workstations requests, the duration of the workstation request execution by server and number of the servers response frames correspondingly.

In the model, the time delays are represented in MTU (Model Time Unit), the sizes – in frames, which have maximal length. Time and date size scaling was considered in [4]. In the model represented in Fig.1, the time delay corresponds to 100 Mbps Ethernet, 1MTU=10 ms, the maximal frame size is 12304 bit. Delays of the sending transitions **sendWS, sendS** contain the time of frames transmission through segment, delays of the receiving transitions **receiveWS, receiveS** contain only the delays, which correspond to the devices performance (Ethernet adapters, switches). Frequency of workstations requests is 10-20 μc; the time of requests execution by server is 1-2 μc; requests length is 1 frame, servers answers length is 10-20 frames.

The model represented in Fig.1 describes the work of all the switches with obligatory frames buffering and all the ports having the same transmission speed. For frames buffers of switches and servers, the random choice discipline was realized. In the evaluation of real-life networks characteristics more complicated variants of models are used. Queues with FIFO discipline are realized for buffers. The transition **Direct** is added for direct forwarding of received frame into the output port (if the queue is empty and the destination port is free). Delay matrices are used for modeling different ports transmission speeds.

The description of model elements which are added for posterior evaluation of characteristics using measuring fragments should be considered in detail. Additional elements model the processes of the frames dump studied in Section 5. At the instant of the frame transmission by a terminal device into a segment via transitions **sendWS**, the copy of its header containing sender address **scr**, destination address **dst**, ordinal number of frame for the device **nf**, and also the time stamp of current model time obtained by function **cT()** are saved into the place **sent**. In the similar way at the instant of the frame receiving by a terminal device via transitions **receiveWS**, **receiveS**, the copy of its header is saved into the place **received**. Moreover, the dump is executed at the receiving/transmitting frames by switches into places **inSW**, **outSW** that model the work of switches statistical subsystem or external packets analyzers attached to the corresponding ports. In the present research such information as switch number **sw** and the current size of switch buffer **bcur** for each input/output frame are saved.

```
colset mac=int with 1..8;
colset portnum=int with 1..4;
colset swch=int with 1..3;
colset nfrm=INT;
colset mac2=product mac*mac*nfrm timed;
colset mac2s=product mac*mac*nfrm*INT timed;
colset sfrm=product mac*mac*nfrm*INT timed;
colset frm=product mac*mac*portnum*swch*nfrm
timed;
colset nseg=product swch*portnum;
colset seg=union f:frm+avail:nseg timed;
colset swi=product mac*portnum*swch;
colset swf=product mac*mac*portnum*swch*nfrm
timed;
colset frame=product mac*mac*nfrm timed;
colset swl=product swch*portnum*swch*portnum;
colset buflimit = product swch * INT;
colset pairch=product mac*mac*INT;
colset zero=int with 0..0;
colset pairch0=product mac*mac*zero;
colset dex= int with 100..200;
colset nse = int with 10..20;
colset Delta= int with 1000..2000;
var src,dst: mac;
var sw,sw1,sw2:swch;
var p,p1,p2: portnum;
var i,t,t1,t2,q,mt,dt,mx,s,pt,m,a,av : INT;
var blim, bcur, bmax: INT;
var nf,nf1: nfrm;
val bitms=12304*10;
fun Dexec()=dex.ran();
fun Nsend()=nse.ran();
fun Delay()=Delta.ran();
fun cT()=IntInf.toInt(!CPN'Time.model_time)
```

Fig. 2. Declarations of colors, variables and functions

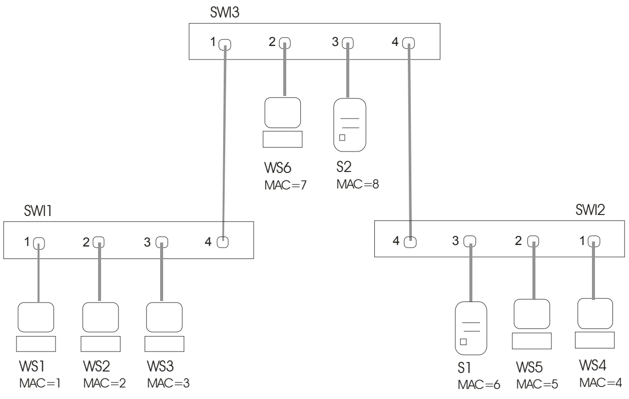


Fig. 3. An example of a switched network

3 Measuring Fragments

The technique of measuring fragments was earlier presented and studied in [4] for nonparametric models. Its main idea consists in the following: as a colored Petri net is a universal algorithmic system, the algorithms of the characteristics calculation may be described by additional fragments of Petri net named by measuring fragments (MF). The model accomplished with measuring fragments implements the calculation of telecommunication network characteristics directly during the process of simulation. The measuring fragments of parametric model for evaluation of traffic, frames delivery time and the size of switches internal buffers are constructed further. Measuring fragments are drawn in red color.

3.1 Network Throughput (traffic) Evaluation

The evaluation of traffic is implemented on the basis of delivered frames dump within terminal devices. It should be noticed that the evaluation might also be implemented on the basis of sent frames dump and the percentage of dropped frames might be obtained. As in the parametric model represented in Fig. 1 the frames dropping process is not modeled, both mentioned evaluations coincide.

The measuring fragment for traffics evaluation is shown in Fig. 4. The fusion place **newFrame** receives the dump of the regular frame received by the terminal device from the network model (Fig. 1). Transition **procFrame** saves its copy into the place **newDbl** for the delivery time evaluation MF and starts the recalculating of characteristics which are stored into places **nFrm**, **nFrmAll**, **trafficAll**, **traffic**. Note that, the formulae of recalculation are represented by the inscriptions of corresponding arcs.

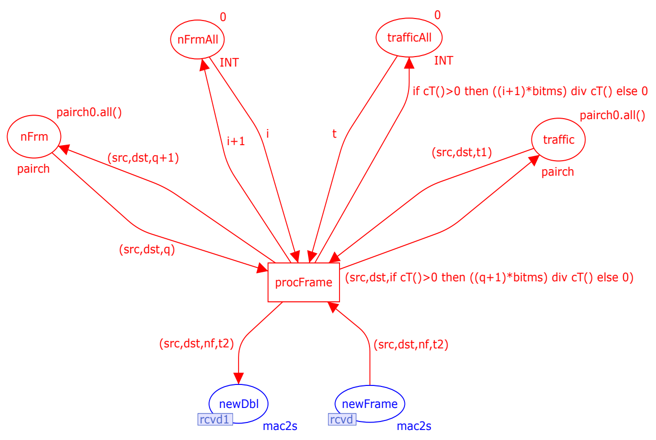


Fig. 4. Measuring fragment for traffic evaluation

The place **traffic** stores the traffic matrix for each pair of MAC-addresses represented with corteges in the form (addr1, addr2, traffic). That allows the evaluation of asymmetrical traffic as the cortege defines the direction of transmission. For the calculation of traffic, the place **nFrm** is used that stores the matrix of transmitted frames quantity in the form (addr1, addr2, quantity). Each firing of the transition **procFrame** increases the quantity of received frames for each pair of addresses: (src, dst, q+1). Traffic is calculated via the division of received frames number by the current model time; the constant **bitms** is used for the reduction of dimension to bit/ms. The simplest formula used for traffic calculation is the following:

$$traffic = {}^{n}\!/_{dt},$$

where n is the amount of delivered information, dt – the timed interval of measuring.

As in the majority of cases the traffic between each pair of devices is a too detailed characteristic, the calculation of an integral characteristic such as the total network traffic represented with the place **trafficAll** is provided. For its calculation the place **nFrmAll** is added that stores the total number of frames received by all the terminal devices.

3.2 Frame Delivery Time Evaluation

The evaluation of frame delivery time is implemented on the basis of calculating the difference between time stamps of the receiving and sending frame for each pair of interacting terminal devices. For the identification of a frame its ordinal number **nf** is used that is unique for each transmitting terminal device.

MF for evaluation of frames delivery time is represented in Fig. 5. Transition **culcDT** calculates frame delivery time **dt**. Transition **culcAVR** starts the recalculation of characteristics stored into places **sumPair**, **sumAll**, **averPair**, **averAll**, **maxAll**, **maxPair**, **quantAll**, **quantPair**.

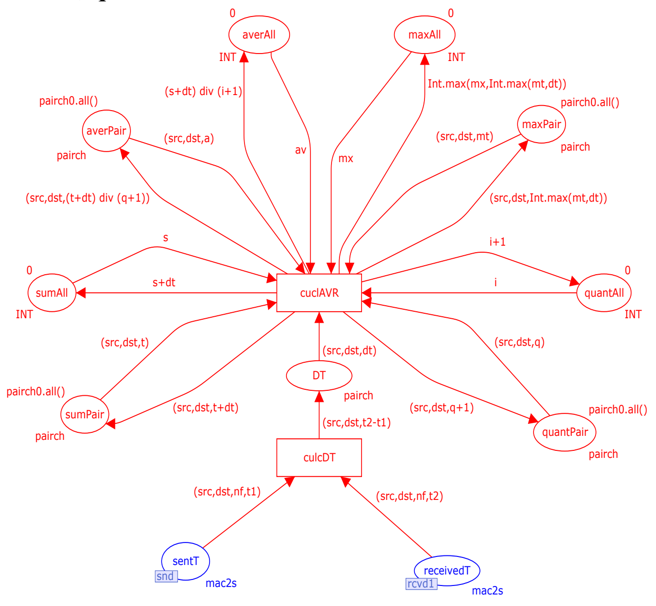


Fig. 5. Measuring fragment for frame delivery time evaluation

Places **sumPair** and **quanPair** store the sum of delivery times and the quantity of delivered frames for each pair of terminal devices correspondingly. They are used for the calculation of the average **averPair** and the maximal **maxPair** frame delivery times for each pair of devices. Note that, at the calculation of averages the information about a newly arrived frame is used: ((t+dt) div (q+1)). The following formula is used for the average delivery time calculation:

$$adt = \frac{(dt_1 + dt_2 + ... + dt_q)}{q},$$

where dt_i is the delivery time of i-th frame, q – total number of delivered frames. Places **sumAll** and **quantAll** store the sum of delivery times and the total number of all the delivered frames correspondingly. They are used for calculation of the average **averAll** and maximal **maxAll** delivery times for all the frames transmitted within the network.

3.3 Switch Buffer Size Evaluation

During the equipment choice and also during the telecommunication networks devices design, the task of determining the devices optimal characteristics is solved. For switches with the given transmission speed on the ports (for instance: 100Mbps, 1Gbps) such characteristics are the average time of frame switching which may be implicitly evaluated on the base of producer information about the number of frames processed in the unit of time [4] and also the size of switch internal buffer of frames.

In Fig. 6 the measuring fragment for the evaluation of switch internal buffer size is represented. Parametric model (Fig. 1) allows the assigning of buffer size limit into place **bufLim** and implements the dump of frames received and sent by switches.

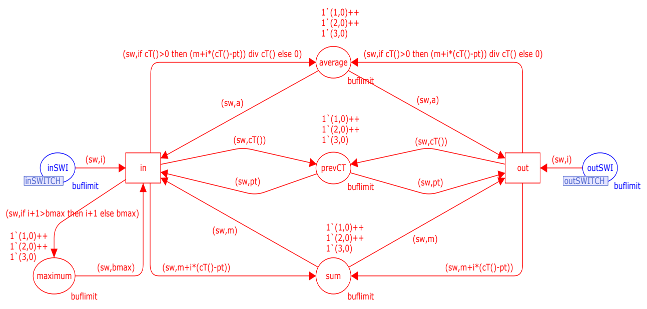


Fig. 6. Measuring fragment for switch buffer size evaluation

On receiving a frame by switch, the number of the switch and the current size of its buffer are stored into the place **inSWI**. The same information is stored into the place **outSWI** at the transmitting of a frame by switch. MF calculates the maximal actual size of the buffer in the place **maximum** and also the average size of the buffer in the place **average**. Auxiliary places **sum** and **prevCT** serve for the storing the sum of products and the value of the previous time instant of the size measurement for each switch. Let us consider the formula of average buffer size calculation in detail:

$$a = \frac{(i_1 \cdot dt_1 + i_2 \cdot dt_2 + \ldots + i_k \cdot dt_k)}{dt},$$

where i_j is the size of the buffer on the time interval dt_j, dt – the total interval of time measurement. As the measurement starts from the zero instant of time, the length of the total time interval equals to the current model time **cT()**. For calculating the current interval dt_j, values of the last measurement time instant pt stored into the place **prevCT** for each switch are used: $dt_j = cT() - pt$. The sum of the products represented in the numerator is accumulated into the place **sum** for each switch separately.

The construction of other measuring fragments is also possible, for instance, for the evaluation of collisions percentage at hubs usage, evaluation of application systems response times etc. In [4] the measuring fragments (for nonparametric Ethernet models) for the evaluation of application system GID-Ural VNIIZT response time which includes network delivery times and time of request processing by server were presented. Such an integral characteristic is the basic one during real-time systems design.

4 Computational Experiments with the Model

For obtaining reliable evaluations of telecommunication networks characteristics, the special organization of computational experiments with the model was implemented. As the processes of requests generation and processing into client-server system are represented with random functions, their interaction with the communication equipment defines a stochastic process. That is why the statistical approach based on calculating the average of distribution and central statistical moments is applied. In the majority of cases two magnitudes: the average of distribution and dispersion are used.

The simulation of the net dynamics was implemented on rather prolonged intervals of model time that correspond to a few minutes of real time. At first, the existence of state stable mode of the model behavior was studied. Then the evaluation of characteristics in state stable mode was implemented.

For each time interval dt_i not less than twenty individual experiments were implemented. Then average a_{dt_i} and dispersion σ_{dt_i} were calculated for each characteristic on the chosen interval. Measurements and calculations were repeated for doubled time interval and so on. If the averages and dispersion coincided $a_{dt_i} = a_{dt_{i+1}}, \sigma_{dt_i} = \sigma_{dt_{i+1}}$ then the decision about a state stable mode existence was adopted. It should be noticed that the absence of a state stable mode might be easily observed, for instance, in case of the increase of requests frequency by factor of 100.

However, the mentioned observance is not concerned with the telecommunication equipment, but with the increase of average numbers of unsent frames into terminal equipment. Telecommunication equipment work normally providing the delivery of frames under peak load due to the modeling flow control facilities stipulated by the standards. Examples of the tables which illustrate the growth of queues in non state stable mode are shown in [8].

Further, in the state stable mode the evaluations of characteristics for various parameters combinations of hardware and software such as the requests frequency, the time duration of processing and the size of switch internal buffer were implemented. In Fig. 7 the current marking of MF for delivery time calculation (Fig. 5) obtained on time interval 168009MTU=1,68 s is represented.

Thus, the average frame delivery time equals to 36 MTU=0,36 μs; maximal delivery time equals to 415 MTU=4,15 μs. From the matrix of delivery times **maxPair**, it might be seen that the maximal delivery time is reached at the delivery of frames sent by the server **S1** (MAC=6) to workstation **WS3** (MAC=3). The shown fragments of matrices acknowledge that the transmission of the frame among the pairs of workstations as well as among the pairs of servers does not take place (corresponding values are zero).

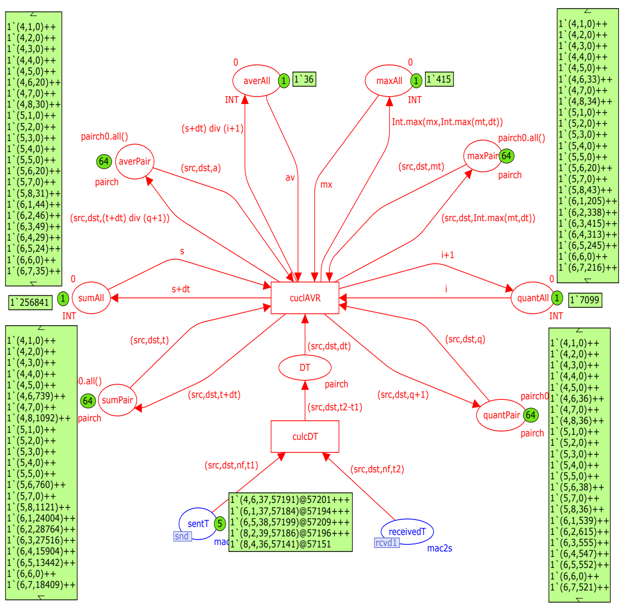


Fig. 7. Results of delivery time evaluation

5 Measuring Real-Life Networks Characteristics

The adequacy of models to real-life objects is the basic question of a research. It is expedient to consider the mentioned characteristics measurement and evaluation technique for real-life networks and apply it to the models construction and the comparison of obtained results. The in-situ measurements of functional characteristics have been implemented on real-life local area networks; an example of the network is represented in Fig. 3.

The easiest way of the measuring is to settle it on the network DTE with WinDump packet analyzer for MS Windows (TCPDump for Unix). The results of the measurements have also been confirmed with software SoftPerfect Network Protocol Analyzer.

WinDump is the software with the command line interface which provides the recording of transmitting the Ethernet frames accomplished with time stamps into a file. Then the content of the file can be viewed and analyzed. WindDump is optimized as to resources consumed and can work in the background mode for a long time, not reducing the computer performance. The following command line provides the recording of the frames into the SavedFrames file:

```
WinDump -w SavedFrames
```

For the analysis of the frames transmission process and calculation of the frames delivery time, the following command line is used:

```
WinDump -ttt -r SavedFrames
```

Option `-ttt` is used for the automatic calculation of timed interval between frames; option `-r` provides the reading of the earlier saved information from the SavedFrames file. An example of the obtained dump of frames is represented in Fig. 8.

```
000252 IP 192.168.0.158.1172 > 192.168.0.130.139: P 854:917(63) ack 840 win 64957
000854 IP 192.168.0.130.139 > 192.168.0.158.1172: . 840:2300(1460) ack 917 win 64502
000141 IP 192.168.0.130.139 > 192.168.0.158.1172: . 2300:3760(1460) ack 917 win 64502
000029 IP 192.168.0.158.1172 > 192.168.0.130.139: . ack 3760 win 65535
000107 IP 192.168.0.130.139 > 192.168.0.158.1172: . 3760:5220(1460) ack 917 win 64502
000138 IP 192.168.0.130.139 > 192.168.0.158.1172: . 5220:6680(1460) ack 917 win 64502
000024 IP 192.168.0.158.1172 > 192.168.0.130.139: . ack 6680 win 65535
000114 IP 192.168.0.130.139 > 192.168.0.158.1172: . 6680:8140(1460) ack 917 win 64502
000086 IP 192.168.0.130.139 > 192.168.0.158.1172: P 8140:9095(955) ack 917 win 64502
000287 IP 192.168.0.158.1172 > 192.168.0.130.139: . ack 9095 win 65535
000606 IP 192.168.0.158.1172 > 192.168.0.130.139: P 917:980(63) ack 9095 win 65535
000729 IP 192.168.0.130.139 > 192.168.0.158.1172: . 9095:10555(1460) ack 980 win 64439
```

Fig. 8. Dump of frames

Let us consider the dump of frames. The first column contains the intervals in milliseconds between the frames entries, then the IP-address and the port number of the sender and receiver follow. After the colon, the fields of packet header are written, such as the first and the last number of passed bytes, packet length in parenthesis, the number of confirmed byte and the window length. In the above example, 192.168.0.158 is the IP-address of the workstation; 192.168.0.130 is the IP-address of the server. Port number 139 corresponds to MS NetBIOS TCP service, port number 1172 is a randomly selected port number of client software.

Time synchronization and the frames dumps comparison in all the terminal devices allow identifying the frames and calculating their delivery times using simple software. The obtained information is a primary one for the calculation of QoS and throughput characteristics of networks. It is significant that the measuring in DTE gives us objective information about actually delivered frames. Moreover, measurements may be implemented in network devices.

Modern Ethernet switches give us wide spectrum of possibilities for the traffic measurement and analysis. For example, CISCO corporation switches, like the series Catalist 4000, 4900 (4948-10GE, ME 4924-10GE) realize the following monitoring services: the check up of the ports condition and possibilities; the analysis of the ports data Switch TopN, the system of statistics collection RMON, the ports analyzer SPAN (Switched Port Analyzer). Usually the switch has the port for immediate connection to console; moreover, it also stipulates the remote access with Telnet and Web-interface for the input of commands. Testing the state of a switch port is implemented with the following command:

```
show port [port_number]
```

Testing the port resources is implemented with the command:

```
show port capabilities [port_number]
```

Port statistics gathering TopN is started by the command:

```
show top [port_number]
```

Displaying the collected statistics is implemented with the following command:

```
show top report [report_number]
```

The service TopN provides the accumulation of such information as: port capacity, the quantity of sent and received bytes, the number of errors and the quantity of buffers repletion. RMON system is started with the command:

```
set snmp rmon enable
```

RMON system accumulates the information about the quantity of sent bytes, received bytes and the number of errors for each port. Moreover, the supplementary possibilities are provided for alarms generation on special events.

SPAN service gives the possibility to redirect the traffic of selected port to another switch port for analysis. For example, the command:

```
set span [nport1] [nport2]
```

redirects the `nport1` port traffic into `nport2` port. The device, which saves the frames into some file or analyzes them, can be connected to the corresponding port. In this way, it is possible to receive the information similar to the damp of frames obtained with TCPDump program.

For testing the model adequacy, the frames dump was started on terminal devices (workstations and servers) in the local area network of Railway dispatcher center (Fig. 3), equipped with GID Ural-VNIIZT system. The frames receiving time is measured, it is accumulated in the dump for a time interval, which equals to one shift (about 12 hours). The comparison of these results and the results obtained via modeling allows the following conclusion: the average error of delivery time evaluation via modeling amounts to no more than 5%. It is a good enough result that acknowledges the adequacy of the built models.

6 Analysis of Simulation Results

Simple evaluations of throughput and frames delivery time on the basis of maximal transmission speed of the chosen technology (100Mbps, 1, 10Gbps) are not realistic even at single switch usage because of asymmetry, pulsation and other peculiarities of realistic traffic. So, for instance, for a switch with *n* ports of 100Mbps technology the maximal throughput close to $n \cdot 100Mbps$ can be provided only in full duplex mode and upon even *n* merely at transmission of 100Mbps flows among the pairs of terminal devices.

In case of the asymmetry of traffic, the destination port of the frame arrived into the switch may be already busy with the transmission of some other frame that leads to either the storing of frame into the switch buffer or the suppression of transmitting device activity with flow control facilities and repeated transmission; as the result the delivery time is increased. Moreover, compulsory idleness of other ports leads to reduction of actual throughput.

The usage of tree-like structure of a few switches (Fig. 3) complicates the described processes and hampers its analytical evaluation. Thus, the usage of simulation models, that adequately describe the processes of frames switching according to the technologies standards and peculiarities of the traffic generating, is a prospective direction of research.

The traditional incremental way of solving telecommunication networks problems is a simple passage to the next level of technology, for instance, from 100Mbps to 1Gbps. But such solutions might be too expensive in the scale of the whole company. Moreover, a new level of transmission speed might appear insufficient for network bottlenecks.

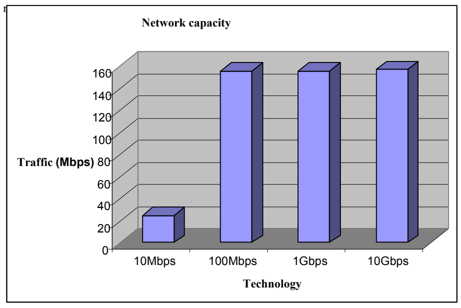
In the present section the results of parametric model (Fig. 1) studied with the help of measuring fragments (Fig. 4, 5, 6) are presented for railway dispatcher centre LAN (Fig. 3) under the choice of various types of switches and Ethernet adapters. The topics of model parameters calculation on the characteristics of real-life equipment were studied in [8]. In Fig. 9 the dependencies of network characteristics on chosen equipment technology are shown.

It should be noticed, that in the regular mode the network should provide the transmission of the whole traffic of servers and workstations. In the considered example of the network each workstation generates two flows of average intensity of 1 frame in 15 ms; in the response to the request each of the servers generates 12 flows (2x6) of 15 frames 15 ms each. Thus, the rough evaluation of the total traffic may be represented as:

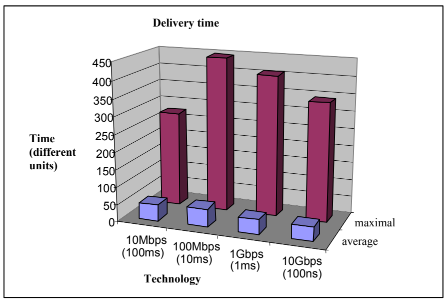
$$traffic = (12 \cdot 16 \cdot 12304)/0{,}015 = 157491200\text{bps} \approx 157\text{Mbps}$$

Starting from the standard transmission speed of the chosen technology and the maximal frame length, the possible minimal (ideal) frame delivery time equals to: 1,23 μs for 10Mbps, 123 ms for 100Mbps, 12,3 ms for 1Gbps and 1,23 ms for 10Gbps. Even at a single switch usage that provides cutting-through transmission without complete buffering, the minimal delivery time is increased. After receiving the frame header, the switch requires definite time for the header analysis and determining the destination port according to the switching table. This time may be estimated on the basis of either declared performance of the switch that is measured in frames per second or the performance of the switch internal bus. So, for instance, declared performance of Intel SS101TX4EU switch equals approximately to 10000 frames per second which corresponds to the frame processing time of about 100 ms; notice that real-life delay may exceed mentioned delay because of the parallel work of ports. For switch Cisco ME 4924 only the performance of internal bus 49 Gbps was declared that corresponds to 251 ns delay. Moreover, real-life performance of Ethernet adapters is distinguished from the maximal transmission speed of the chosen technology. So, for instance, Ethernet adapter Intel Ether Express PRO/100 provides maximal transmission speed 92,1Mbps that corresponds to the frame transmission delay of 144 ms.

From Fig. 9a) it can be seen that 10Mbps technology does not provide the transmission of all the generated flows of frames; the state stable mode is not reached in the system containing networking and terminal equipment that is acknowledged by the growth of the queues length into the place replyS. More fast technologies provide the transmission of all the flows; throughputs differ in the bounds of dispersion. But delivery times (Fig. 9b) differ considerably; note that, different units of delivery time measurement were used for different technologies. The total tendency is that the maximal delivery time exceeds the average merely tenfold. The decrease of maximal delivery time for 10Mbps technology may be explained by the considerable fall of throughput. During the study of the mentioned characteristics for various performance of Ethernet adapters and switches and also buffer sizes of chosen switches, only minor variation of characteristics was revealed merely in the bound of dispersion. Thus, for the considered traffic generated with periodical requests of workstations to servers, only the choice of technology is essential; the differences in the equipment performance actually do not affect the characteristics of the network. Note that, the maximal delivery time (Fig. 5, place maxALL) can be used as an estimation of guaranteed delivery time for real-time systems with hard timed bounds. For systems with soft timed bounds, average delivery time (Fig. 5, place averALL) can be used in estimations.



a) Throughput



b) Delivery time

Fig. 9. Evaluation of network characteristics

During the study of the model under the small sizes of switch internal buffer of frames, an anomaly leading to mutual blocking of all the switches work was revealed. The mentioned anomaly might take place at arbitrary buffer sizes and specific peculiarities of traffic but with the increase of the buffer size its probability reduces considerably. Fig. 10 shows the simplest variant of switches mutual blocking in a network consisting of two switches with the buffers size equaling to 2 frames (value 1 for the model).

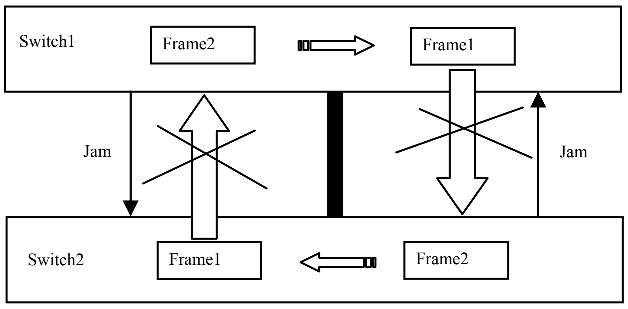


Fig. 10. Mutual blocking of switches

Suppose that two frames have arrived into Switch1 with the destination address of Switch2 terminal devices and at the same time two frames have arrived into Switch2 with the destination address of Switch1 terminal devices. Suppose each of the switches started the transmission of its first frame. As, the frame cannot be located into the buffer, each of the switches suppresses the transmission of the frame using flow control facilities. The mutual blocking of the switches occurs. The considered clinch does not occur in case of using cut-through switches that work without compulsory buffering. In this case Frame1 of Switch1 is directly forwarded by Switch2 to the corresponding port of destination terminal device. Moreover, in real-life switches the operation of the frames dropping is implemented for the frames the storing time of which exceeds the limits. Thus, the clinch is eliminated but the performance of the switch falls dramatically due to the inevitable dropping of the frames.

7 Conclusion

Thus, in the present work the refined parametric model of switched Ethernet was presented that is invariant with respect to network structure and the quantity of attached communication and terminal devices. The model was accomplished by measuring fragments providing the evaluation of throughput, frame delivery time and switches buffer sizes during the process of simulation.

The adequacy of the model has been confirmed by real-life networks characteristics measurements; the analysis of modeling results was implemented for various types of used equipment. The area of obtained results application is the design of telecommunication networks and devices close to optimal.

References

1. Jensen K. Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Springer-Verlag, 1997, Vol.1-3.
2. Beaudouin-Lafon M., Mackay W.E., Jensen M. et al. CPN Tools: A Tool for Editing and Simulating Coloured Petri Nets. LNCS 2031: Tools and Algorithms for the Construction and Analysis of Systems, 2001, 574-580. (http://www.daimi.au.dk/CPNTools)
3. Zaitsev D.A. Switched LAN simulation by colored Petri nets. Mathematics and Computers in Simulation, vol. 65, no. 3, 2004, 245-249.
4. Zaitsev D.A. An Evaluation of Network Response Time using Coloured Petri Net Model of Switched LAN // Proceedings of Fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, October 8-11, 2004, Aarhus (Denmark), 157-167.
5. Zaitsev D.A., Shmeleva T.R. Principes of parametric Petri net models construction for switched networks // Proceedings of First International Simulation and Computer Graphics Conference, October 4-7, 2005, Donetsk (Ukraine), 207-215. In Russ.
6. Zaitsev D.A., Sakun A.L. An Evaluation of MPLS Efficacy using Colored Petri Net Models // Proc. of of International Middle Eastern Multiconference on Simulation and Modelling (MESM'2008), Amman (Jordan), August 26-28, 2008, 31-36.
7. Bereznyuk M.V., Gupta K.K., Zaitsev D.A. Effectiveness of Bluetooth Address Space Usage // Proceedings of 20th International Conference, Software & Systems Engineering and their Applications (ICSSEA 2007), December 4-6, Paris, 2007.
8. Zaitsev D.A., Shmeleva T.R. Switched Ethernet Response Time Evaluation via Colored Petri Net Model // Proceedings of International Middle Eastern Multiconference on Simulation and Modelling, August 28-30, 2006, Alexandria (Egypt), 68-77.

Decomposition into open nets

Stephan Mennicke and Olivia Oanea and Karsten Wolf

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{stephan.mennicke, olivia.oanea, karsten.wolf}@uni-rostock.de

Abstract. We study the decomposition of an arbitrary Petri net into open nets. This means that shared places can be seen as message channels between components. We show that there exists a unique decomposition into atomic components which can be efficiently computed. We further show that every composition of components yields a component and that every component can be built from atomic components. Finally, we briefly discuss potential applications.

1 Introduction

Several authors [6, 1] proposed open nets as a means for open components or web services. An open net is Petri net with an interface consisting of places. Each place in the interface is either an input or an output place.

We study the problem of decomposing an arbitrary Petri net into open nets. This means that we want to find open nets such that the whole net is structurally isomorphic to the composition of the components. The actual challenge is that a shared place must belong to exactly two components where it serves as input place for one component and as output place for the other.

We show that there are elegant and efficient techniques for handling a decomposition into open nets. In fact, every component constitutes of atomic components which can be computed in little more than linear time.

An implementation of the technique teaches as that practically relevant Petri nets decompose into many and small components. Hence, decomposition into open nets can be used for divide-and-conquer algorithms. A potential candidate is Commoner's property [4, 3] since Petri net structures such as siphons and traps relate quite regularly to open net components. Another interesting application could be the decomposition of business process models into web services where the atomic components can be aggregated into components of useful size by applying available role annotations in the activities.

2 Definitions

We use the standard notation $[P, T, F]$ for a (place/transition) Petri net. We require for all nets $|T| > 0$ and disallow isolated nodes. Arc weights are ignored for simplicity; their insertion would not cause any problem.

An open net $[P, T, F, \mathit{In}, \mathit{Out}]$ consists of a Petri net $N = [P, T, F]$, and an interface $I = [\mathit{In}, \mathit{Out}]$ with $\mathit{In} \cup \mathit{Out} \subseteq P$ and $\mathit{In} \cap \mathit{Out} = \emptyset$. In the interface, *In* is the set of input places and *Out* is the set of output places. We require ${}^\bullet\mathit{In} = \emptyset$ and $\mathit{Out}^\bullet = \emptyset$. A set M of open nets is composable iff their sets of transitions is disjoint and each place belongs to at most two nets in M. Thereby, it must be an input place in one net and an output place in the other. The composition $\oplus M$ of a composable set M of open nets is obtained by building the union of the respective constituents.

A decomposition of a Petri net $N = [P, T, F]$ is a set M of open nets such that $\oplus M = [P, T, F, I, O]$, for some I and O. $\{[P, T, F, \emptyset, \emptyset]\}$ is the trivial decomposition of $[P, T, F]$. A decomposition of an open net

$N = [P, T, F, \mathit{In}, \mathit{Out}]$ is a set M of open nets such that $\oplus M = N$. $\{N\}$ is the trivial decomposition of open net N.

Decomposition M_1 of N is finer than decomposition M_2 of N if, for each $N \in M_2$, there is a subset of M_1 which is a decomposition of N. M is a finest decomposition of N if it is finer than every decomposition of N.

3 Existence of a finest decomposition

Throughout this section, we consider a Petri net $N = [P, T, F]$. We prove the existence of a finest decomposition by construction. To this end, we inductively define an equivalence relation R on $P \cup T$. For every decomposition of N, different nodes that are related by R occur as inner nodes in the same component. In the following definition, let X^* be the reflexive and transitive closure of relation X.

Definition 1 (Relation R). *Let $R = \bigcup_{i \in \mathit{Nat}} R_i$ where*
(Base:) $R_0 = (\{[t_1, t_2] \mid \exists p : p \in P \cap {}^\bullet t_1 \cap {}^\bullet t_2$ or $p \in P \cap t_1^\bullet \cap t_2^\bullet\})^*$.
(Step:) $R_{i+1} = (R_i \cup \{[p, t_1], [t_1, p] \mid \exists t_2 : p \in P \cap {}^\bullet t_1 \cap t_2^\bullet, [t_1, t_2] \in R_i\})^*$.

It is easy to see that all the R_i, and thus R, are equivalence relations. Note that the pairs $[p, t_2]$ and $[t_2, p]$ are inserted in the step as well, due to the transitive closure.

Lemma 1. *If $[x, y] \in R$ and $x \neq y$ then x and y occur as internal nodes in the same component in every decomposition of N.*

Proof. (Base:) Assume that $\exists p : p \in P \cap {}^\bullet t_1 \cap {}^\bullet t_2$. Then t_1 and t_2 cannot occur in different components since otherwise p must be an interface place where both components consume tokens, in contradiction to the definition of a decomposition. Both t_1 and t_2 are internal as transitions cannot be part of the interface. The case $p \in P \cap t_1^\bullet \cap t_2^\bullet$ is analogous. Building the reflexive and transitive closure does not destroy the property as "occurrence in the same component" is naturally a transitive relation.
(Step:) By the inductive assumption, $[t_1, t_2] \in R_i$ implies that t_1 and t_2 occur as internal nodes in the same component. As t_1 consumes tokens from p while t_2 produces tokens on p, p cannot be an interface place. Consequently, it must be an internal place in the same component as t_1 and t_2. Again, building the transitive closure is harmless. $\square$

The following few lemmas prepare the actual construction of the finest decomposition.

Lemma 2. *Every non-singleton equivalence class contains transitions.*

Proof. No pairs between two places are inserted into R (except for transitive closures). $\square$

Next we show that classes which contain transitions are not adjacent.

Lemma 3. *If $[p, t] \in F$ or $[t, p] \in F$ ($p \in P, t \in T$) such that $p \not R t$, then $\{p\}$ is an equivalence class w.r.t. R.*

Proof. Assume that p is part of a nontrivial class. Then p is connected to other nodes only in the step of the construction of R. This means that the equivalence class of p contains both a pre-transition and a post-transition of p. By the inductive base, however, one of these transitions must be equivalent to t, so, by transitivity, p, and t would be equivalent, in contradiction to the initial assumption. $\square$

In consequence, every class that contains transitions is surrounded by singleton place classes. Next we show that a singleton place class $\{p\}$ has at most one class from where tokens are produced to p, and at most one class from where tokens are consumed from p.

Lemma 4. *Let $\{p\}$ be an equivalence class w.r.t. R. Then all transitions in ${}^{\bullet}p$ are equivalent and all transitions in $p^{\bullet}$ are equivalent.*

Proof. This is evident from the construction of R_0. □

The above lemmas show that open nets can be constructed by letting a class that contains transition serve as the set of inner nodes while the adjacent singleton place classes are used as interface.

Definition 2 (Open net from equivalence class). *Consider an equivalence class E according to the above relation R that contains at least one transition. The corresponding open net $[P', T', F', I, O]$ is determined by $T' = E \cap T$, $P' = T'^{\bullet} \cup T'^{\bullet}$, $F' = F \cap (P' \cup T')$, $I = (P' \setminus E) \setminus T'^{\bullet}$, and $O = (P' \setminus E) \setminus {}^{\bullet}T'$.*

Lemma 5. *The just defined structure is indeed an open net.*

Proof. We have $|T'| > 0$ since E contains a transition. No node is isolated as only pre-and post-places of T' are considered, and each element of T' has at least one pre- or post-place. I and O are valid sets of input and output places as problematic transitions are removed. □

Lemma 6. *The set of constructed open nets are composable.*

Proof. This is an immediate consequence of Lemma 4. □

Now we are ready to prove the main result of this section.

Theorem 1. *Every Petri net has a unique finest decomposition into open nets.*

Proof. Consider relation R and the just constructed set of open nets. Each node occurs in a class. If the class contains a transition, then all its elements form the inner of some constructed open net. If the class does not contain transitions, it is a singleton place class. Since this place p is not isolated in N, it is connected to at least one class that contains a transition. So, p appears in the interface of one or two of the constructed open nets. Each arc of N appears in one of the constructed nets as well. The constructed set of open nets is composable. Consequently, the composition of this set exists and contains all nodes and arcs of N (and no other nodes or arcs). Hence, the set of open nets is a valid decomposition.

Consider any other decomposition and one of its components C. By Lemma 1, C contains, with every node, the whole equivalence class of that node. If it contains a class which contains transitions, it must also contain all adjacent singleton place classes since otherwise the arcs from or to the place in the singleton class cannot be represented in any composition involving C. Consequently, C is the union (in other words, the composition) of some open nets as constructed above. □

Corollary 1. *The finest decomposition is unique.*

Proof. The relation "finer as" can easily be identified as a partial order. Partial orders, however, cannot have more than one minimum. □

The main result can be interpreted as follows: Every decomposition can be obtained by first building the finest decomposition and then composing some disjoint subsets to larger open nets. The last result of this section considers the same idea the other way round:

Theorem 2. *Let $\mathcal{C}$ be the finest decomposition of N. Let $\mathcal{P}$ be a partition of $\mathcal{C}$. Then $\{\oplus P \mid P \in \mathcal{P}\}$ is a decomposition of N.*

Proof. This can be easily verified as composition is plain union of the ingredients, and composability is not affected by the constructions. □

The results of this section justify the name *atomic components* for the members of the finest decomposition.

4 Constructing a finest decomposition

The main task in constructing the finest decomposition is to compute the equivalence relation R. We represent R as the corresponding partition of $P \cup T$. Initially, the partition consists of singleton nodes of the given net only. We access the partition using two operations. $Find(x)$ takes a node as input and returns the (unique) class in the partition that contains x. $Union(C_1, C_2)$ takes two (not necessarily different) classes as argument and modifies the partition by replacing C_1 and C_2 by $C_1 \cup C_2$. The result is a (coarser) partition. Tarjan [7] showed that a partition can be organised such that an arbitrary sequence of n union and find operations can be executed in $O(nlog * n)$ time where $log*$ is an extremely slowly growing but asymptotically unbounded function. Our considerations result in the procedure sketched in Table 1.

Consequently, the finest decomposition can be computed in little more than linear time.

5 Potential applications

Divide-and-conquer techniques for structural analysis

Some traditional Petri net structures, in particular siphons (a set of places S where $^\bullet S \subseteq S^\bullet$) and traps (a set of places R where $R^\bullet \subseteq {}^\bullet R$) behave quite regularly w.r.t. the proposed way of decomposition. If S is a siphon in $N = [P, T, F]$ and $C = [P', T', F', \mathit{In}, \mathit{Out}]$ is a component of some decomposition, then $S \cap P'$ is a siphon in C. The other way round, if $N = N_1 \oplus N_2$, S_1 is a siphon in N_1, S_2 is a siphon in N_2 and both S_1 and S_2 contain the same interface places, then $S_1 \cup S_2$ is a siphon in N. Traps behave similarly. In future work, we aim at exploiting this observation for a divide-and-conquer algorithm for deciding Commoner's property (every siphon contains an initially marked trap) which allows nice implications to liveness of free-choice nets or deadlock-freedom of arbitrary Petri nets.

Extracting services from business process models

Open nets have been studied as models for web service behavior. Business processes can be modeled as Petri nets as well. A decomposition of a business process model into open nets thus yields a separation of the business process into web service. The finest decomposition is not necessarily a useful decomposition. However, building larger components by composing, for instance, components with identical role annotations may lead to the identification of useful portions of the net which could be separated as a web service.

Table 1. Construction of finest decomposition

```
Input: Net $[P, T, F]$
Output: Set of open nets $M$

Var $U$: SET of SET of Nodes

Init: $U = \{\{x\} \mid x \in P \cup T\}$

FOR ALL $p \in P$,$t_1, t_2 \in T$ DO
    IF $p \in t_1^\bullet \cap t_2^\bullet$ THEN $union(find(t_1), find(t_2))$;
    IF $P \in {}^\bullet t_1 \cap {}^\bullet t_2$ THEN $union(find(t_1), find(t_2))$;
END
REPEAT UNTIL nothing changes
    FOR ALL $p \in P$,$t_1, t_2 \in T$, $t_1 \neq t_2$ DO
        IF $find(t_1) = find(t_2)$ AND $p \in {}^\bullet t_1 \cup t_2^\bullet$ THEN $union(find(t_1, find(p)))$;
    END
END
$M := \emptyset$;
FOR ALL $X \in U$ DO
    IF $X \cap T \neq \emptyset$ THEN
        $TT := X \cap TT$;
        $PP := TT^\bullet \cup {}^\bullet TT$;
        $FF := F \cap (PP \cup TT) \times (PP \cup TT)$;
        $II := (PP \setminus X) \setminus TT^\bullet$;
        $OO := (PP \setminus X) \setminus {}^\bullet TT$;
        $M := M \cup \{[PP, TT, FF, II, OO]\}$;
    END
END
```

6 Examples

Applying our decomposition to academic examples leads to widely varying results. A particular version of the n dining philosophers net yields $2n$ open nets in the finest decomposition. Figure 1 shows the decomposition of the dining philosophers net for $n = 3$. Each open net consists of just two transitions (with a same label) and five interface places. Some standard net that grants concurrent read access and exclusive write access to some data base decomposes into no more than two open nets, regardless of the number of reading and writing processes. In realistic examples, our technique tends to decompose into many and small components, as the following numbers show. Using the implementation available online under `www.service-technology.org/diane`, we checked more than 1700 workflow nets provided by IBM Zurich. The nets had up to 273 places, 284 transitions, and 572 arcs. The resulting open net in the decomposition had at most 66 places, 12 transitions, and 66 arcs. In fact, many resulting open nets consisted of only one transition and a couple of interface places.

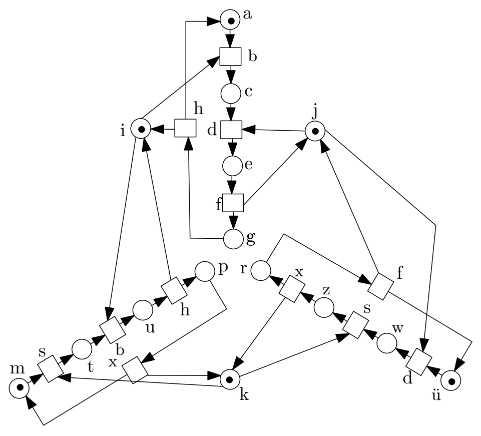


Fig. 1. Dining philosophers net and its decomposition

7 Related Work and Conclusion

Decomposition of Petri nets is a well-studied domain [2, 8]. [5] considers decomposition of nets where cut places and transitions are given and compositional verification of Commoner property for asymmetric choice nets. The idea to cut a net such that the border places form an asynchronous message passing interface, however, appears to be new. The closest approach seems to be the decomposition of a net into conflict clusters. In fact, every conflict cluster of a net is local to a dingle open net component (cf. the base in the inductive definition of R). In contrast to conflict clusters, our components are also closed under backward conflict clusters (see again the definition of R). Given these observations as well as the sketched potential applications, this kind of decomposition could eventually pay off.

References

1. P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional modeling of reactive systems using open nets. In *CONCUR*, volume 2154 of *LNCS*, pages 502–518. Springer, 2001.
2. G. Berthelot. Transformations and decompositions of nets. In *APN 1986*. Springer-Verlag, 1987.
3. F. Commoner. *Deadlocks in Petri Nets*. Applied Data Research, Inc., Wakefield, Massachusetts, Report CA-7206-2311, 1972.
4. M.H.T. Hack. Analysis of Production Schemata by Petri Nets. Master's thesis, MIT, Dept. Electrical Engineering,, Cambridge, Mass, 1972.
5. Li Jiao. Decomposition of nets and verification in terms of decomposition. In *CIMCA/IAWTIC*, pages 804–809. IEEE Computer Society, 2005.
6. E. Kindler, A. Martens, and W. Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *BPM*, volume 1806 of *LNCS*, pages 235–253. Springer, 2000.
7. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.
8. W. Vogler and B. Kangsah. Improved decomposition of signal transition graphs. *Fundam. Inform.*, 78(1):161–197, 2007.

Profiling Services with Static Analysis

Jan Sürmeli

Humboldt-Universität zu Berlin, Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
suermeli@informatik.hu-berlin.de

Abstract. In a service-oriented architecture, *Services* are components that interact with each other via well-defined interfaces. *Open nets* are a special class of Petri nets, designed to model the behavior of open systems. Asynchronous interaction and stateful behavior complicate predicting the combinations of messages that a service can process. We present *profiles* which support the modeler in verifying compliance of the model with given constraints, without knowing its future environment. We explain the computation of profiles by static Petri net analysis.

1 Introduction

In a service-oriented architecture (SOA) [1], *services* interact with each other by exchanging messages over predefined *channels*. Typically, services are understood as software artifacts that offer a functionality over a well-defined *interface*, defining those channels a service uses. Control and data flow of services heavily depend on the interaction with other services. We aim at analyzing the behavior of a service S and thus model S with *open nets* (or service nets, open workflow nets) [2]. We will describe this extension to classical Petri nets in Sec. 2.

Well-developed methods to analyze the behavior of a service S with open nets, already exist alongside different *behavioral correctness criteria* such as *controllability* [3] and *exchangeability* [4]. Those criteria can be verified by *dynamic* techniques and were implemented in a tool chain[1]. Another subject is to check *behavioral constraints* on services and their compositions. We can demand *orders* on messages, *causalities*, *limits* et cetera. Work in this area has been done in [5]. We can think of different levels of abstraction as well as extensions of the classical evaluation of a constraint to true or false, e.g. three valued logics, probabilities or costs. On top of that, services behave dependent on their environment, so we are interested in overapproximating the behavior for all or a subclass of all possible environments. Work on structural analysis on open nets has been done in [6].

In this paper we present an approach to solve a specific class of behavioral constraints based on well-known *static* Petri net analysis techniques. Such a constraint specifies lower and upper *bounds* for the occurrence of events in the interaction with other services. An *event* occurs when a message is sent or received. A constraint might e.g. demand the exchange of a single message or

[1] Available at http://www.service-technology.org/tools

restrict the occurrence of a *linear combination* of events. A *profile* for S is a set of constraints that S satisfies. Since there exists an infinite number of linear combinations of events, there also exists an infinite number of profiles for S, providing different levels of precision. But we can influence this aspect when computing a profile, such that it meets the requirements of the use case. We formally define the class of constraints, the concept of profiles as well as their computation with static Petri net methods in Sec. 3.

Profiles can be applied in different phases in the lifecycle of a service S. The modeler of S can use profiles to prove that the model complies with the specification. Other analysis methods can benefit from profiles, in this case profiles are used as a preprocessor. During implementation of S, profiles can be used to generate test cases, since a profile contains constraints that the model satisfies. After deploying S it will be available in a service repository. A profile can be used to store an abstraction of S that can support behavioral query resolving. A profile, however, does not cover all aspects of S: It is restricted to abstract interaction behavior. We will discuss the application of profiles in Sec. 4. We conclude the paper in Sec. 5 with open issues and ideas for further work.

2 Modeling with Open Nets

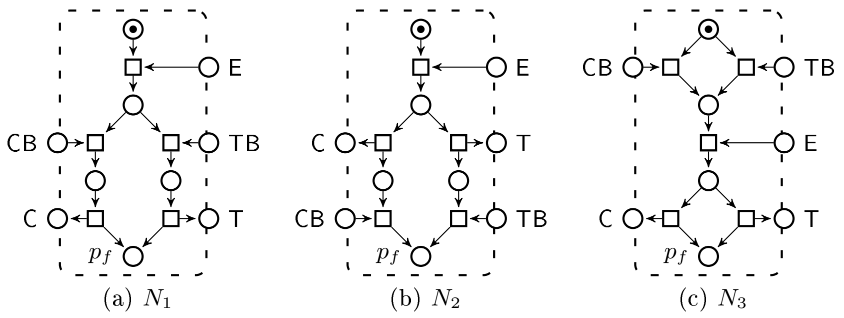


Fig. 1. Three open nets with the final marking $[p_f]$

Two services interact by exchanging messages over *channels* predefined in the interface of a service. We assume an *asynchronous* communication model: Sending and receiving of a message does not occur in the same moment, as opposed to hand-shaking-techniques. Thus, for each exchanged message, two events occur: Sending and receiving. From the viewpoint of one of the involved services however, only one of the two events is observable, namely the event of sending or receiving a message by the service itself. For example, service S sends a message that is later received by service S', then for S the sending event is observable and from the viewpoint of S' only the receiving event occurs.

Furthermore we assume that a service only communicates unidirectional over a message channel: Either messages are sent or received via this channel.

We use the classical syntax and semantics of Petri nets as in [7]. We define the behavior of a Petri net N as the set of all sequential runs in N. *Open nets* are Petri nets that are augmented by a *final marking* and an *interface* for message exchange, the latter is realized by designating some places as input and some as output places. These so called *interface places* are used as connectors for the composition. Other places are called *internal places* and the net structure is that of a classical Petri net.

Figure 1 shows three open nets with the same interface, graphically emphasized by the dashed line. E, TB and CB are input places, C and T are output places. N_1, N_2 and N_3 are models of simple coffee/tea machines that have one button for coffee (CB), one for tea (TB), an input for money (E) and two output slots, one for coffee (C) and one for tea (T). Although the three models have the same interface, they differ obviously in the internal structure.

The modeling of interface places is only the prerequisite for message exchange, message exchange can only occur between a number of open nets that are *composed*. Two two open nets N and N' can be composed if they are *syntactically compatible*, meaning that they do not share internal places or transitions and have compatible interfaces. Two interfaces A and B are compatible if the output (input) channels of A are not used as output (input) channels in B and vice versa. *Composition* is done by union of the net elements, composing initial and final marking and merging identically named interface places that become internal places.

3 Profiles for Open Nets

In this paper, we approach a specific class of constraints: *Lower and upper bounds for event occurrence*. Intuitively a constraint of that class specifies the bounds for legal interaction: An event or a linear combination of several events is only allowed to occur in those bounds. An example for a constraint is $2 \leq a \leq 7$, meaning that the event a occurs at least two and at most seven times. A constraint however need not give an integer value for one or both bounds, it can also specify one of the bounds as *unbounded*, meaning that interaction is not constrained in that direction. For example $2 \leq a \leq \_$ demands that event a occurs at least two times but might occur infinitely often. Dependencies between messages can be expressed easily as well: $1 \leq a+b \leq 1$ states that always exactly one of the two events occur. $0 \leq a-b \leq 0$ demands that a and b occur equally often, since the inequality can easily be transformed to $a = b$. We can not specify temporal orders between messages, causalities or more complex dependencies. The open nets N_1, N_2, N_3 in Fig. 1 all three comply to the constraints $E = 1$, $0 \leq C \leq 1$, $CB + TB = 1$. Only N_1 and N_2 comply with the constraints $CB - C = 0$, $TB - T = 0$, $CB + T = 1$ and $TB + C = 1$ that demand that TB (CB) only occurs with T (C). We can not express order restrictions with such constraints, so we can not forbid a behavior as in N_2.

Formally, a *constraint* c is a quadruple $\langle A, \theta, l, u\rangle$, where A is a finite set of events, $\theta : A \rightarrow \mathbb{Z}$ is a linear combination of events and $l, u \in \mathbb{Z} \cup \{\_\}$. The semantics of such a constraint is as follows: θ, l, u form an inequality ϕ of the form $l \leq \theta \leq u$, with the elements of A as variables. Let N and N' be open nets. A run r in $N \oplus N'$ satisfies c from the viewpoint of N, written $c \vdash_N r$ if and only if the occurrence rate of events in the projection of r to transitions of N is a satisfying assignment for the variables in c. A sequential run is *terminating* if and only if it starts at the initial marking and ends at the final marking. N complies with a constraint c if, for an arbitrary N' and every terminating sequential run r in $N \oplus N'$, $c \vdash_N r$ holds.

A *profile* of N specifies a set of constraints that N complies with. These constraints need not be the strictest ones that apply but might be quite liberal. Computation of a profile for an open net can be done by *static* analysis, avoiding state space construction. Thus, a profile is an *abstraction* of the behavior of N with any arbitrary service N'.

Knowledge of N' can not be assumed, therein we find the first challenge for computing a profile. We approach the problem by overapproximating every possible N' by a canonical open net that has the most liberal interaction behavior, called the *unrestricted environment* of N, denoted as $\widehat{N}$: An open net, only consisting of exactly one transition for each interface place of N and no internal places. One would not encounter $\widehat{N}$ in the practical field, but it proves to be very helpful in analyzing the service in interaction with $\widehat{N}$ and deducing its behavior in an arbitrary environment from the results. The connection between the behavior of $N \oplus \widehat{N}$ and $N \oplus N'$ is the following: Fixing any run in $N \oplus N'$, we can find a run $N \oplus \widehat{N}$, such that the two are equivalent if we just concentrate on N's part. We demonstrate this on the example of the open net N_1 in Fig. 2(a): Figure 2(b) shows the unrestricted environment for N_1 and their composition is depicted in Fig. 2(c).

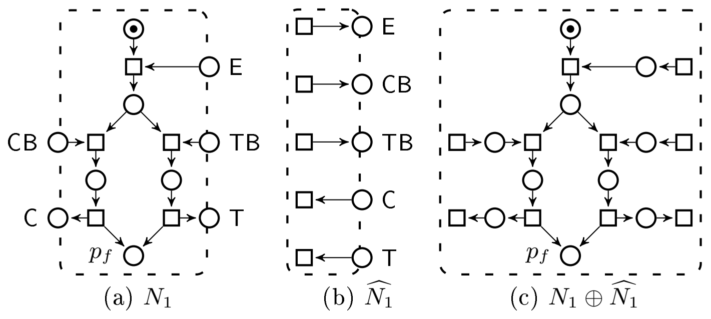


Fig. 2. N_1 (repeated from Fig. 1), its unrestricted environment $\widehat{N_1}$ and $N_1 \oplus \widehat{N_1}$.

However, for computing a profile we do not construct and explore the state space, but use a classical static Petri net method, namely the *state equation*, a canonical system of linear equations, taking into account two markings β, β'. For every run from β to β', there exists a solution m, such that transitions occur as often in the run as given by m. We construct the state equation of $N \oplus \widehat{N}$, setting $\beta = \alpha$ and $\beta' = \omega$. We add an equation for each possible event in N, specifying the transitions of N that let an event occur. The set of all solutions is thus both an overapproximation of all terminating sequential runs r in $N \oplus \widehat{N}$ and gives the occurrence rates for events in r from the viewpoint of N.

Given a set of linear combinations of events, we can now apply *linear programming* to find lower and upper bounds for these combinations. Solutions of these linear programs are constraints that N complies with. Thus we directly construct a profile. The set of linear combinations is the only parameter, the rest is canonical on the net structure. We can distinguish between two general starting points for profiles: (1) there exists a specification, given as constraints before computing the profile, (2) we compute the profile in advance. In the first case, the input for the profile computation can be taken directly from the specification. In the second case, the selection of useful linear combinations has to be done manually, although we can imagine taking into account structural properties like invariants, conflict situations and the like.

The computation of profiles for a given open net has been implemented in the tool Linda[2]. It takes a set of constraints as input and computes an according profile, using the lp_solve-library[3] to solve linear problems. Interoperability with other tools is enforced by usage of the same open net format as the other tools in the above mentioned tool chain.

4 Application of Profiles

Given an open net N and a set of constraints C, we can create a profile ψ to determine compliance of N to the constraints in C. In some cases, however, we are neither able to prove compliance nor non-compliance with this method. We demonstrate the compliance checking process with an example. Let N be

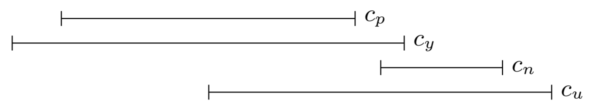


Fig. 3. The bounds given by four constraints c_p, c_y, c_n, c_m.

an arbitrary open net and c_y, c_n, c_u be constraints that restrict the same linear combination. Creating a profile for any of the singleton sets $\{c_y\}$, $\{c_n\}$ and $\{c_u\}$ leads to the same result, we denote it as $\{c_p\}$. Let the bounds given by the constraints c_p, c_y, c_n, c_u be as depicted in Fig. 3. We first check compliance with

[2] Available at http://www.service-technology.org/tools/linda
[3] Available at http://lpsolve.sourceforge.net

c_y: N complies with c_y since the bounds given by the profile imply those of c_y. In contrast to that, N does not comply with c_n: The lower bound of c_n is greater than the upper bound of c_p. In the case of c_u however, we can not decide compliance or non-compliance of N with c_u by the method of profiling.

Given a profile ψ of N, we can also check compliance of N with a constraint c, although the linear combination of c is not directly restricted by any constraint in ψ: We determine a constraint restricting the same linear combination as c that is implied by the constraints in ψ and then do compliance checking as explained above. Finding such an implied constraint can be done by linear programming.

5 Conclusion and Further Work

We have described a class of *constraints* for the interaction behavior of a service. Our approach to the solution is the computation of a *profile*, a set of constraints that a service complies with, using *static* Petri net analysis methods. We have demonstrated how profiling can be used to *check compliance*, answering with *yes*, *no* or *unknown*.

It is part of further work to determine the inputs of profile computation if they can not be derived from a specification, such that a profile can be stored as an abstraction of an open net. Since a profile is an abstraction, we can think of making use of refinement techniques to further explore questions answered with *unknown*. We will embed profiles in different *tools* to analyze services and their composition, so that they benefit from preprocessed information or on-the-fly checking of constraints.

References

1. Gottschalk, K.: Web services architecture overview. http://www.ibm.com/developerworks/web/library/w-ovr/ (2000)
2. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics **1**(3) (2005) 35–43
3. Wolf, K.: Does my service have partners? T. Petri Nets and Other Models of Concurrency **2** (2009) 152–171
4. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding Substitutability of Services with Operating Guidelines. Transactions on Petri Nets and Other Models of Concurrency II, Special Issue on Concurrency in Process-Aware Information Systems **2**(5460) (March 2009) 172–191
5. Lohmann, N., Massuthe, P., Wolf, K.: Behavioral constraints for services. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings. Volume 4714 of Lecture Notes in Computer Science., Springer-Verlag (September 2007) 271–287
6. Oanea, O., Wolf, K.: An efficient necessary condition for compatibility. In Kopp, O., Lohmann, N., eds.: ZEUS. Volume 438 of CEUR Workshop Proceedings., CEUR-WS.org (2009) 81–87
7. Reisig, W.: Petri nets: an introduction. Springer-Verlag New York, Inc., New York, NY, USA (1985)

An Approach to Business Process Modeling Emphasizing the Early Design Phases

Sebastian Mauser, Robin Bergenthum, Jörg Desel, Andreas Klett

Department of Applied Computer Science, Catholic University of Eichstätt-Ingolstadt
sebastian.mauser, robin.bergenthum, joerg.desel,
andreas.klett@ku-eichstaett.de

Abstract. This paper proposes an approach to formal business process modeling emphasizing the early design phases. That means, the focus is on gathering requirements of a business process in an informal environment. First, methods to systematically elicit all requirements are discussed. Then, it is suggested to formally model and validate the elicited requirements before integrating them to a formal business process model and verifying the model w.r.t. the formal requirements. The approach is inspired by techniques which have proven successful in the area of software requirements engineering. The key technique is the application of scenarios to bridge the gap between the informal view on the process by practitioners and the formal business process model.

1 Introduction

Business process modeling is an important part of many software development projects [1, 2] because software is often applied in the context of business processes. But the number and variety of purposes, business process models are used for, is growing. Business process modeling and management has attracted increasing attention going beyond software engineering in recent years [3, 4]. Process models are more and more used for pure organizational purposes like mere documentation, process reorganization and optimization, certification, activity-based costing or human resource planing. Business process models are also applied as input to workflow systems to control and monitor the proper execution of work items.

In this context, it is very important that the models are valid, i.e. that they correctly and completely represent the relevant aspects of the underlying real-world business process. However, although the need for valid process models increases, there usually is little effort on guaranteeing validity in practice. Mostly, process models are constructed ad hoc – usually in workshops – without detailed documentation of the different requirements of the users involved. Also in theory, most approaches to business process design and according tools assume validity of models and concentrate on analysis (e.g. soundness tests) and optimization issues. But analysis and optimization of invalid models is useless and decisions based on invalid models or execution of invalid models will cause errors.

We faced the problem of generating valid process models in a recent industrial project with the purchasing department of the AUDI AG. Correct modeling of processes is very important for AUDI, because in the recent years documentation of business processes more and more became a major part of the requirements for a TÜV (short for Technischer Überwachungs-Verein, Technical Inspection Association in English) certification for German automobile manufacturers. The increasing importance of valid business process models caused AUDI to ask for our academic support in modeling. The practical problems in such a large company make the modeling of valid business processes really difficult. The processes are typically inter-divisional such that the processes of the purchasing department have impact on the whole company. They are supported by a heterogeneous system landscape and include many media breaks. Within our project we developed an approach of how to come to valid process models in such a complex setting.

To design valid business process models significant attention should be paid to early phases of business process design, i.e. to the question how to systematically gather information about a business process in an informal environment. This part of business process modeling has not yet been sufficiently considered in the literature. But we claim that similar problems have been tackled in the field of requirements engineering for software systems. Therefore, the core idea of our approach is to adopt findings of requirements engineering to the area of business process modeling, in particular w.r.t. the early phases of the design process.

In requirements engineering, scenario based specifications proved to be a successful starting point. Consequently, our approach also starts with elicitation of scenarios which are single process instances of a business process model. Our approach suggests to then formalize and validate the process instances up to a level of preciseness and completeness such that formal methods can be applied in the follow-up steps of integrating the scenarios to a formal process model, e.g. an Event-Driven Process Chain (EPC) or a work flow Petri net, and of verifying the model w.r.t. the scenarios.

The approach is developed within our industrial project but we claim that in principle it can be applied for business process modeling in general. We want to present a first proposal for a respective modeling approach in this workshop paper. Still, we are collecting further experiences in the ongoing project.

The paper is structured as follows: The following section provides a rough survey on the state-of-the-art of requirements engineering in software development, with a particular focus on scenarios. In the sequel, some of these ideas are adopted to early phases of business process modeling. Section 3 provides a comparative study of literature concerning the early phases of business process modeling. Section 4 describes our modeling approach. It is structured in several subsections, referring to the different phases of our approach. Finally, Section 5 provides some concluding remarks.

2 Software Requirements Engineering: A Short Review

In software engineering, significant attention has been paid to conceptual modeling bridging the gap between informal information about the information system to be implemented and the final implementation. Main approaches have been structured analysis and structured design, developed in the late 1970's, and object-oriented analysis and design, starting in the late 1980's [5]. In the 1990's it was generally accepted that requirements engineering [6] – the elicitation, documentation and validation of requirements of a system - is a fundamental aspect of software development and thus requirements engineering emerged as a field of study in its own right.

Information system analysts discovered that faulty requirements analysis was a major reason for project failure or unsatisfactory software systems and that the costs of errors grows exponentially with progressing time in the development process, see e.g. [7, 6]. Therefore, in many cases improving the quality of requirements by using more structured approaches and formal models to elicit and articulate user and domain requirements is likely to both improve the quality of delivered information systems and reduce the costs of system development.

In early stages of requirements engineering, user oriented specification models are desired to describe the required behavior of a complex system from the user's viewpoint while for implementation of the system, integrated state-based system models are necessary [5]. For intuitive user oriented behavioral specifications, scenarios, firstly introduced by Jacobson's use cases [8], proved to be the key concept. Domain experts know scenarios of a complex system to be modeled better than the system as a whole. Thus, starting with scenarios helps to gather system specifications which are valid, i.e. they faithfully reflect the real system requirements. Important advantages of using scenarios at the beginning of the requirements engineering process include the view of the system from the viewpoint of users, the ease of understanding (by different groups of stakeholders), the possibility to write partial specifications and to incrementally extend specifications, easy abstraction possibilities, short feedback cycles, the possibilities to directly derive test cases, the documentation of user-oriented requirements and the possibility to derive scenarios from log files recorded by information systems [9, 10]. However, scenarios cannot capture the entire desired behavior of a system in a structured fashion [5]. Therefore, the final phases of requirements engineering dealing with implementation, final system design and documentation, analysis, simulation or optimization issues, require an integrated state-based model regarding the complete reactivity of (each component of) the system. Since we are interested in the early phases of system design in this paper, we focus on the scenario view of a system in the following.

In the literature the topic of modeling software systems by means of scenarios has received much attention over the past years, see e.g. [6, 5]. Popular scenario notations include e.g. the ITU standard of Message Sequence Charts, the UML diagrams suitable to model scenarios, namely Sequence Diagrams, Communication Diagrams, Activity Diagrams and Interaction Overview Diagrams, as well as Live Sequence Charts, Scenario Trees, Use Case Trees or Chisel Diagrams [10]. But such diagrams can often hardly be used and understood by typical users [11]. On the other hand, the complexity of natural language specifications of typical users in real world situations can often hardly be handled by developers [12]. Scenario modeling approaches accounting for such problems are presented in [6, 13]. Notable approaches in requirements engineering developed to generally bridge the gap between natural language specifications and the variety of conceptual modeling languages are the KCPM [14, 13] (Klagenfurt conceptual pre-design model) and the information modeling approach of [12].

Having finally modeled the requirements of a system by scenarios, the next challenge is to come from the scenario view of a system to a state-based system model, which is closer to design and implementation. Also for this problem several methodologies have been proposed, see e.g. [15, 10, 6, 13, 5].

3 Requirements Engineering in Business Process Modeling

We in this paper claim that for modeling valid business processes ideas from software requirements engineering are appropriate. But domain specific problems require the application of (partly) new techniques and approaches in the field of business processes. For instance, when modeling a software system the scope is usually clearly focused around this system and it always has to be kept track of implementation issues [6]. On the other hand, a business process model often has a larger scope, including many systems and even crossing organizational boundaries, and it often includes many implementation independent parts such as interactions between humans [4]. Moreover, the focus of software modeling [5, 15, 6] is on components or objects, communication (dependencies) between components and the distinction between inter- and intra-object behavior, while the emphasis of business process modeling [3, 4] is on global activities (where modularity comes into play by appropriate refinement and composition concepts), dependencies through pre- and post-conditions of activities, and resources for activities.

Detailed ideas how to elicit requirements on a business process were for the first time raised in the articles [1, 16], where some initial work on this topic has been done by adapting the KCPM [14] approach. There are several further articles mentioning the suitability of requirements engineering activities for the actual design of business process models, see e.g. [17–19], but they do neither go into details concerning this topic nor do they discuss a copious application of requirements engineering techniques in particular for the first phases of business process modeling.

Apart from the two highlighted articles, so far, only few research focused on the early phases of business process modeling can be found. There are a lot of methodologies for the modeling of business processes, but, as far as we know, they rarely provide elaborate systematic approaches to gather the information necessary for developing a process model. Notable examples of business process modeling procedures covering some aspects concerned with gathering process requirements include the approaches presented in [2, 18, 19] and some of the approaches mentioned in the survey paper [17]. In particular, the ARIS approach [20, 21] is very successful in practice. But also these approaches lack a detailed discussion of problems and concrete methods concerned with this topic. Rather, as it is true for most other modeling approaches, their focus is on problems relevant in later modeling stages such as the discussion of process modeling techniques and how to apply the techniques to capture certain behavior. In this paper, we are interested in the first phases of modeling, namely how to find the behavior that should be modeled in an informal environment. Only if this behavior is correctly elicited, a model implementing the behavior can be expected to faithfully represent the intended business process. Although, except from [1, 16], we found no comprehensive methods in literature dealing with the early stages on the way to come to a valid business process model, the mentioned business process modeling procedures [2, 18, 19, 17, 20, 21] present some valuable ideas on this topic. Moreover, there are several helpful strategies and assisting procedures supporting the elicitation of information about a process, e.g. the papers [22–26] apply the user view of scenarios in the context of business process design, many papers such as [27, 28] discuss how to formally integrate different views on a process, the surveys [3, 4] include some early modeling strategies, and finally there are several user-oriented modeling techniques (see e.g. [29] for some recent trends) such as design principles (top-down, bottom-up and inside-out approaches), ideas for the management of modeling activities (e.g. terminology, conventions, process model governance and ownership), tool support for several modeling activities (see e.g. http://bpmn.org), reference models (best practices), patterns (http://www.workflowpatterns.com) and modeling guidelines (quality factors).

4 Modeling Approach

In this section we present our comprehensive approach to model business processes. The approach is inspired by the concepts of scenario-based requirements engineering, i.e. we suggest focusing on scenarios of a business process before designing an integrated process model. Looking at scenarios to specify the behavior of a business process has similar advantages as for the software engineering domain, in particular user-oriented intuitive modeling is supported. The starting point of our approach is distributed knowledge about a business process in an informal real-life environment. The aim is to first develop a comprehensive formal specification of the business process by scenarios and some other types of requirements artifacts. The single formal artifacts can easily be checked for correctness according to the real-life requirements ensuring a valid specification. Then the artifacts are integrated into a business process model given by some modeling language and the generated process model is verified w.r.t. the artifacts. It is important to mention that integration and verification heavily benefit from having a valid formal specification, because this allows (semi-) automatic generation of a process model from the specification, e.g. by Petri net synthesis [24, 26] or merging procedures [27, 28], and formal verification whether a process model fulfills the specification is possible. Altogether, the construction of complex process models behaving valid according to the requirements is supported.

Besides the influences from the requirements engineering domain mentioned in Section 2, our methodology adopted several ideas from the articles on business process modeling cited in Section 3, in particular, from the highly related work of [1, 16]. But in contrast to [1, 16], our approach focuses on scenarios, it is seen independently from the software engineering domain and it is less technical but more detailed in the concepts of the first modeling stages. In general, the difference to all other process modeling approaches is that the methodology of this paper concentrates on the early modeling phases of gathering all relevant information in an informal environment and of the transition from the informal setting to more and more complex formal models.

Our approach is divided into the five phases elicitation, formalization, validation, integration and verification (see Figure 1) and the additional orthogonal phase of information management. The first three phases are inspired by the three dimensions of requirements engineering and the respective requirements engineering activities suggested in [6].

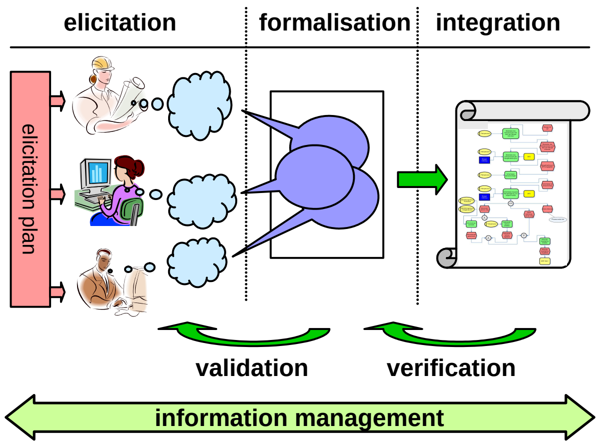
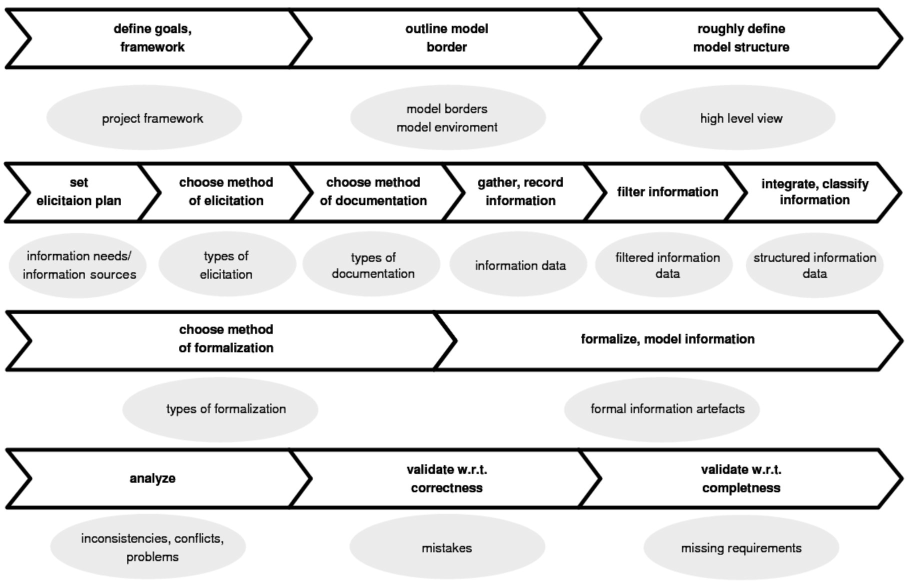


Fig. 1. An approach for business process modeling.

The focus in the elicitation phase is on gathering information. The main problem is to relate and combine the information collected from different information providers in an informal environment into a valid database of knowledge. Therefore, an elicitation plan which determines appropriate strategies for the elicitation procedure has to be assembled. The information collected according to the elicitation plan has to be filtered and documented, yielding a collection of pieces of information. The pieces have to be formalized to information artifacts in the next phase. This enables validation in a follow-up feedback phase. The valid information artifacts document the process requirements which can be integrated into a process model. The process model is finally verified w.r.t. the documented requirements. During all these five phases, it is necessary to manage the progress of information retrieval and to organize all gathered information in the orthogonal phase of information management.

Remark that, while we describe our approach as a sequence of five phases together with one parallel phase, for applying the approach we do not suggest to adhere strictly on the given sequential ordering of phases. On the one hand it is not always possible to generate a process model in one run, such that phases have to be iterated, i.e. it is necessary to repeat phases when information is missing. On the other hand, sometimes it is helpful to move to a next phase, in particular having elicited certain information items, it can be useful to directly formalize and validate them before further proceeding with the elicitation phase.

Next we explain each of these six phases. Thereby, we concentrate on the elicitation, formalization and validation phases and discuss them in more detail. Figure 2 shows all necessary steps of these phases together with the resulting objects. The first two lines refine the elicitation phase, the third line refines the formalization phase and the last line refines the validation phase. The model incorporates ideas from different domains concerned with information retrieval (e.g. [30, 14, 12]).



define goals, framework
outline model border
roughly define model structure
project framework
model borders model enviroment
high level view
set elicitaion plan
choose method of elicitation
choose method of documentation
gather, record information
filter information
integrate, classify information
information needs/ information sources
types of elicitation
types of documentation
information data
filtered information data
structured information data
choose method of formalization
formalize, model information
types of formalization
formal information artefacts
analyze
validate w.r.t. correctness
validate w.r.t. completness
inconsistencies, conflicts, problems
mistakes
missing requirements

Fig. 2. Steps in the elicitation, formalization and validation phase.

4.1 Elicitation

The first three steps at the very beginning of the elicitation phase are the basis of the next six core elicitation steps. When modeling a business process, the starting point is to define scope and aim of the project. It is necessary to set up the project framework which surely influences all decisions made in later steps. Next, the outline of the process has to be defined. This clarifies the border of the process together with the environment and its interfaces. Now, that the business process is set to its context, a first rough structure of the process including aims, related organizational structures and involved documents and systems has to be identified, preferably with the help of a domain expert having a high level view on the process. This helps to get an overview of the information which has to be elicited, i.e. the information needs, and on potential information sources. To actually set up an elicitation plan, it is important to gather more detailed knowledge about available information providers and existing documents describing the process. Such a plan organizes the choice of people or documents which may provide detailed information about parts of the process. For each information to be collected the elicitation plan contains a list of information providers and documents. Gathering information often leads to the following specific problems: the information contains redundancies and repetitions, homonyms and synonyms, exceptional cases to be handled, implicit information or confusions between schema and instance level. To tackle these problems, an adequate elicitation method together with a harmonized documentation method has to be chosen. Since this is an important decision to be made, we will provide a suggestion for both. After these choices have been made, the next step is to gather and record all the information according to the specified elicitation methods.

This normally leads to a large set of loosely arranged information pieces, collected in different kinds of documents, which have to be filtered in the next step of the elicitation process. Filtering corrects all the above listed problems identified in the collected set of information items. The last step integrates and classifies the collected knowledge, i.e. the loose pieces of filtered information are merged and ordered in a structured way according to the chosen documentation method. This concludes the elicitation phase.

Before describing the follow-up formalization phase, we tackle the problem of choosing an appropriate elicitation and documentation method. Often there exist several kinds of documents to be considered, such as working instructions, already existing process models, intra-net information or even theses about parts of the process. Also, exist a lot of methods for elicitation of requirements from the information providers such as interviews, monitoring, logging, rollplays, discussing, questionnaires, meetings, etc [6, 31]. Practical experience suggests to first elicit all adequate documents to get a good overview of the process before starting to consult information providers. Eliciting from information providers needs considerable effort such that a good previous knowledge about the process is desired. After having elicited documents we suggest to interview information providers focused on discussing scenarios. The interviews should be guided by the following framework: After a short round of introduction, the aim of the interview has to be explained to the information provider. This includes level of abstraction, borders, environment and, if already available, interfaces of the process to be modeled. We found it very helpful to shortly introduce the concept of scenarios before the actual interview, because information given by the information provider then was much better structured. Introducing already existing process models to illustrate the sort of the aspired model has similar positive effects. After that, firstly, single scenario instances or even real live examples should be elicited and documented as structured text. Together with the information provider, then a scenario schema is deduced from the scenario instances and documented in a precast scenario form. These forms include entries such as name, description, information provider, activities, events, ordering relation, variations and exceptional cases, pre- and post-conditions, goals and results, success factors and responsibilities. Having filled out such a scenario form, we ask for details about single important activities, events, involved systems, business objects or actors and document them in similar precast forms as well. Although our suggestion here is to focus on scenarios, sometimes it is helpful to discuss whole process fragments which can be done in a similar way. Process fragments are scenarios containing alternatives or loops. Some information providers experienced in modeling may find it easier to describe complex process structures directly in terms of such process fragments. Generally, within the interviews it is important to always mind completeness and clarity of each information recorded and to accomplish the interviews in an appropriate intensity (see e.g. [31]).

The precast forms are already part of our documentation method. We not only use the forms in the interviews, but each type of form defines a template for storing information. As far as possible we insert each gathered information to such a form and port the forms into a database. This yields one table in the database for each type of template. Information not fitting any template is stored in an additional table of the database as structured text together with general specifications such as information provider. Such an organization of information allows to generate different perspectives on the stored requirements through different database queries and search functionalities. It is also possible to automatically produce requirements documents following certain standards.

The main building blocks of a process model are activities, events and different kinds of objects. Therefore, to prepare the requirements for process modeling, we classify the information collected in the tables of the database in a similar way as described in the ARIS [21] methodology. Each information has to be classified into one out of three different views described in Figure 3 (the ARIS methodology suggests the views organization, data, function and one control view relating the first three). Every information about activities or events is stored in a process view. Information about objects is divided into a data and an organizational view. Resource and data objects as well as systems and applications are assigned to the data view and objects concerned with responsibilities and access rights are assigned to the organizational view. Each kind of template naturally belongs to one of these views. Scenarios, activities and events belong to the process view, business objects and systems to the data view and actors to the organizational view. Each information stored in the general purpose table explicitly has to be assigned to a view. Note that in our approach the process view is the dominant one and the relations between the views are naturally included within the process view (instead of a separate control view), e.g. by systems or roles associated to activities (as already given by the templates).

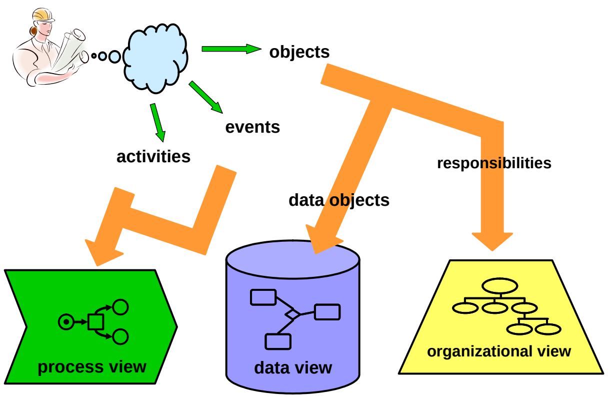


Fig. 3. Three different views to formalize data.

Additionally to the requirements database, we suggest to build a dictionary (similar to an approach described in [14]) to define a consistent language for activities, events and objects used in the documentation. To allow a quick matching of information gathered from different sources, each item in this dictionary is given with a short description (similar to a glossary) and a list of possible synonyms (similar to a thesaurus). In order to navigate through the dictionary we allow to add relations between items yielding a thesaurus-like representation of the domain specific vocabulary of the business process at hand. In particular, a hierarchical ordering of the items is useful to find notions used in the dictionary for certain objects. Similar as in the case of object oriented modeling, it is useful to refine the hierarchy by distinguishing an is-a, a part-of and a property-of relation if possible. Additionally, an association relation to generally link related items is important. Via a graphical representation it is possible to feedback such a dictionary to the information providers. A tool nicely visualizing the relations of the items, having intuitive hide/show functionalities and a search functionality is necessary here. Finally, it is important to link the dictionary to the process information stored in the database, e.g. by adding hyperlinks between the dictionary and respective notions occurring in the entries of the database.

4.2 Formalization

The next two steps depicted in Figure 2 are allocated in the formalization phase. This phase is necessary to get precise requirements of the intended process model. Using informal or semiformal models instead of formal ones for specifying requirements can lead to misunderstandings between author and recipient of a model.

There is a rich variety of different formalisms to choose from. The choice may depend on the documentation method, the target process modeling language or surrounding conditions in the enterprize. Generally, graphical modeling languages should be preferred. The structured pieces of information gathered in the elicitation phase can be translated in an appropriate formal representation. Each formal model of a requirement is called information artifact. The formal models should be linked with the respective documented information.

Our suggestion is to formalize scenarios similar to instance EPCs [32, 22], but allow both events and activities (functions) which not necessarily strictly alternate. That means, it is possible to specify any ordering between activities and events such that any kind of scenario specification can be regarded. In particular, it is possible to consider partial orders of activities called runs or partial orders of events called lifelines. Process fragments (if elicited) can be formalized similarly by adding alternatives to the scenarios. To express process fragments it is also possible to already use the target process modeling language or to simply specify a respective set of scenarios for each fragment. The other way round, having a highly related set of scenarios, it may be helpful to directly fuse them to one process fragment, if that is easy, in order to account for their strong connection. Additionally, we use formalization concepts for pre- and postconditions, relations between activities or events, invariants or behavioral restrictions (which might be derived from elicited business rules) and triggers in the process view. For the data view we use ER-diagrams and related concepts within the UML, and for the organizational view organigrams or group and role concepts are applied.

4.3 Validation

After some information artifacts have been prepared, their validation is started. This is done in three steps. Before we take a closer look at them, we discuss some basics about validation. Validation of the formalized requirements is a necessary phase in our approach because it is easier to check the requirement artifacts than checking a whole process model. When the validation phase takes place at an early stage of the whole modeling procedure, mistakes recognized at such a stage can be clarified with less effort. As a consequence, validation is a task that also takes place in the first two phases of elicitation and formalization, e.g. the preparative high-level model, the elicitation plan and the filled out precast forms have to be validated (using the same methods applied in our actual validation phase). But the main validation phase of our approach, which we discuss in this subsection, is accomplished using the information artifacts after the phases of elicitation and formalization. This is because at this stage the requirements have to be correct and complete to allow a reasonable further processing in the integration phase. Therefore, a validation-quality-goal which has to be passed by every single information is applied.

The first step of validation, called analysis, solely deals with the documented information not involving any information provider. In this analysis step the unambiguous formal representation of the information artifacts enables to check for inconsistencies (e.g. some precondition never appears as a postcondition), conflicts (only occurs when there are more than one information provider for specific information) and similar problems in the requirements. Concerning conflicts, it is important to identify, analyze and solve them in this early stadium. They are documented because it might be that the same controversial subject reappears later on. Concerning inconsistencies, even automated or semi-automated analysis methods are applied for analyzing formal artifacts, e.g. matching preconditions and postconditions, checking every artifact if it is formalized with a correct syntax or analyzing patterns. Besides examining the formal representations, we also investigate the applied precast forms. These have to be checked to contain no empty fields and that the content of the fields has the right form. Altogether, in this first step of validation we discover problems and lack of clarity within the information artifacts. But the problems and unclear parts can only be resolved with additional information from the information providers. That is the reason why the analysis takes place before the other two steps of validation, namely validation w.r.t. correctness and validation w.r.t. completeness (see Figure 2). Within these two steps, one returns to the information providers, now knowing further points to discuss.

In the step of validation w.r.t. correctness, besides trying to resolve conflicts, inconsistencies and similar problems, it is checked in detail whether the artifacts faithfully model the intended requirements. The main goal of this step is to eliminate mistakes coming from misunderstandings during the elicitation phase and mistakes coming from a faulty transfer from informal requirements to formal artifacts during the formalization phase. Therefore, the information artifacts are discussed with the corresponding information provider, i.e. it is asked if they express exactly what the provider meant. For the discussion, we lean on standard validation techniques from software engineering [6] such as inspections, reviews or walkthroughs. Due to their concreteness and clearness, our main modeling concept of scenarios is very well suited for such discussions. If necessary, the artifacts can be discussed with different information providers (having different perspectives on one topic) or even external specialists, so that the final information artifacts can really be regarded as correct. This part of the validation w.r.t. correctness is performed in collaborative meetings or one-on-one interviews. As assisting techniques for this step of validation we first use perspective based reading (the information provider has to concentrate on a special point of view or role while reading an artifact) to reveal problems [6]. Moreover, we apply automatic approaches for validating formal information artifacts w.r.t. correctness. It is very useful to simulate the given scenarios or to test them towards performance. Even, first prototypes or process fragments are synthesized from the scenarios (and additional artifacts). Automatically analyzing them as well as feedbacking them to information providers often leads to new insights whether the respective input scenarios are correct or not.

The last step in the validation phase is the validation w.r.t. completeness. This step should not be seen independently from the former validation step. That means, often both validation steps take place together, e.g. within the same discussions with information providers, and often the same validation techniques, e.g. discussion techniques, are applied. But, nevertheless, we have distinguished this part of validation from validation w.r.t. correctness, because there is a different focus, namely to test whether the gathered information are not yet complete. We use the following validation techniques specially tailored to find missing information: First, examining existing scenarios, process fragments or prototypes as well as unfolding process fragments or prototypes yields hints or inspirations towards additional scenarios. Second, matching equal states of different scenarios or finding structural dependencies between different scenarios indicate that other combinations of parts of the scenario should be considered. Third, it is important to check if every single context aspect is taken care of, e.g. if all interfaces, stakeholders and objects of the environment are covered by and fit to the specified scenarios. In addition to these three techniques, one also has to simply ask the information providers if they can suggest further providers which might contribute additional information.

4.4 Information Management

Parallel to the five other phases depicted in Figure 1, there is an information management phase, providing the necessary infrastructure for storing, relating and updating all the documents and data, as well as monitoring the progress of the steps described above. Regarding the infrastructure, an appropriate tool management, data management and file system has to be established. Concerning the monitoring of the progress of the activities (note that this task is sometimes also considered a part of validation [6]) a simple to-do list is suggested. In this list every row represents one special information and every column represents the activities to be performed for one single information. The rows can be taken directly from the elicitation plan. The columns should not only contain the main activities elicitation, formalization, validation, integration and verification, but also more precise steps like “make appointment”, “prepare appointment”, “filter information”, and “classify information” are very useful. In particular for the validation phase, a detailed consideration of the progress, even regarding a subdivision of the different applied validation techniques, is necessary. Using such a to-do list concept we suggest to not only keep quantity aspects in mind, since this could mislead to just superficially perform the activities such that they can be marked in the list as done. A respective list has a quality meaning as well. So, when marking a task as done one should check the quality of its execution and of its results. For certain tasks, it is even helpful to supplement the main to-do list by additional detailed checklists [6].

4.5 Integration and Verification

As depicted in Figure 1, the next phases of our modeling approach are the integration of the gathered requirements into a formal process model, e.g. an EPC or a Petri net, and verification of the model w.r.t. the requirements. For the integration, we suggest a semi-automated approach. Given the formal requirements as an input, automatic synthesis methods can suggest model components to a user and automatic test methods can on the fly check the correctness of each component added by hand. That means, the process is designed by a modeling expert, based on the requirements, assisted by a program proposing components to be added and constantly checking for inconsistencies between the designed process model and the requirements. Such a method is faster and less error prone than modeling without algorithmic support. There are also advantages of a semi-automated approach [10] compared to a fully automated approach. Firstly, being involved into the modeling process increases the understanding of the model. This is very important when the model should be extended or revised. Secondly, during the modeling process ambiguous situations can explicitly be solved by the user, and he is explicitly confronted with problems occurring when translating the requirements into the process modeling language.

In the verification phase reliable formal requirements are a basis to apply verification methods which check whether the process model correctly reflects the specified requirements such as testing methods to check the executability of specified scenarios, unfolding methods to check wether the process model has additional non-specified behavior and model checking methods for the verification of formalized business rules. Besides, there are specification independent correctness criteria for process models, such as the absence of deadlocks, certain soundness criteria or structural properties, which also have to be checked by formal methods in the verification phase.

5 Conclusion

The described approach of emphasizing the early design phases of business process modeling together with a consequent documentation and formalization of the information gathered has many benefits. Most important is that as explained throughout the paper the approach heavily supports the generation of valid models in an informal environment. Furthermore, the elicited requirements are documented in such a way that it is possible to get a detailed understanding of them at any time, e.g. if the business process has to be changed or expanded one can build on the existing requirements. Especially, in the case that the business process model is for statutory warranties, its requirements have to be traceable. Moreover, the formal approach enables reliable validation of the information collected and verification and testing of the constructed process model. Formal requirements also support the application of formal methods such as synthesis in the integration phase.

By now, within the AUDI project, we followed our modeling approach for example business processes and found the results most promising. We also developed and applied prototype tools supporting the early phases in our process modeling approach. From the experiences of our project we in particular discovered that in practice there are several difficulties that have to be regarded. There are legal constraints, e.g. it can happen that information providers are not allowed to be stored together with the given information, and organizational constraints, e.g. information providers are often not available. Most important is the factor of time and cost such that a tradeoff between the invested effort made in the early phases of process modeling and the related cost has to be found. Still, as learned from software engineering for important process models we think that emphasizing early design phases always pays off.

The integration and verification phases of our process modeling approach still have to be elaborated. This is the focus of our further theoretical research. In particular, we plan to support the integration phase by adjusting methods known from Petri net synthesis and to develop verification methods by adapting the theories of testing executability of scenarios and calculating unfoldings of Petri nets (algorithms in all these areas are implemented in our toolset VipTool [24, 25], see http://viptool.ku-eichstaett.de). Concerning further practical work, we plan a comprehensive evaluation of the modeling approach going beyond our current industrial project.

References

1. Mayr, H.C., Kop, C., Esberger, D.: Business Process Modeling and Requirements Modeling. In: ICDS 2007, IEEE (2007) 8–14
2. Oestereich, B.: Objektorientierte Geschäftsprozess-Modellierung und modellgetriebene Softwareentwicklung. HMD - Praxis Wirtschaftsinformatik **241** (2005)
3. Aalst, W., Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
4. Weske, M.: Business Process Management – Concepts, Languages and Architectures. Springer (2007)
5. Harel, D., Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer (2003)
6. Pohl, K.: Requirements Engineering. dpunkt (2008)
7. Faulk, S.: Software Requirements: A Tutorial. In: Software Engineering, IEEE (1995) 82–101
8. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)
9. Glinz, M.: Improving the Quality of Requirements with Scenarios. In: Second World Congress on Software Quality, Yokohama (2000) 55–60
10. Amyot, D., Eberlein, A.: An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development. Telecommunication Systems **24**(1) (2003) 61–94
11. Moody, D.L.: Cognitive load effects on end user understanding of conceptual models: An experimental analysis. In: ADBIS, LNCS 3255, Springer (2004) 129–143
12. Frederiks, P.J.M., van der Weide, T.P.: Information modeling: The process and the required competencies of its participants. Data Knowl. Eng. **58**(1) (2006) 4–20
13. Fliedl, G., Kop, C., Mayr, H.C.: From Textual Scenarios to a Conceptual Schema. Data Knowl. Eng. **55**(1) (2005) 20–37
14. Mayr, H.C., Kop, C.: A User Centered Approach to Requirements Modeling. In: Modellierung 2002, LNI 12, GI (2002) 75–86
15. Liang, H., Dingel, J., Diskin, Z.: A Comparative Survey of Scenario-Based to State-Based Model Synthesis Approaches. In: SCESM 2006, ACM (2006) 5–12

16. Salbrechter, A., Mayr, H.C., Kop, C.: Mapping Pre-Designed Business Process Models to UML. In: IASTED Conf. on Software Engineering and Applications 2004, IASTED/ACTA Press (2004) 400–405
17. Holten, R., Striemer, R., Weske, M.: Vergleich von Anstzen zur Entwicklung von Workflow-Anwendungen. In: Software Management 97. (1997) 258–274
18. Weske, M., Goesmann, T., Holten, R., Striemer, R.: A Reference Model for Workflow Application Development Processes. In: WACC, ACM (1999) 1–10
19. Castela, N., Tribolet, J.M., Guerra, A., Lopes, E.R.: Survey, Analysis and Validation of Information for Business Process Modeling. In: ICEIS, Ciudad Real (2002) 803–806
20. Scheer, A.W.: Architecture of Integrated Information Systems: Foundations of Enterprise-Modeling. Springer (1992)
21. Scheer, A.W., Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. In: BPM 2000, LNCS 1806, Springer (2000) 376–389
22. Dongen, B., Aalst, W.: Multi-Phase Process Mining: Aggregating Instance Graphs into EPC's and Petri Nets. In: 2nd Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management, Petri Nets 2005, Miami (2005) 35–58
23. Desel, J.: From Human Knowledge to Process Models. In: UNISCON 2008, LNBIP 5, Springer (2008) 84–95
24. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of Petri Nets from Scenarios with VipTool. In: Petri Nets 2008, LNCS 5062, Springer (2008) 388–398
25. Desel, J., Juhás, G., Lorenz, R., Neumair, C.: Modelling and Validation with Viptool. In: BPM 2003, LNCS 2678, Springer (2003) 380–389
26. Bergenthum, R., Desel, J., Mauser, S., Lorenz, R.: Construction of Process Models from Example Runs. In: ToPNoC, Springer (to appear in 2009)
27. van Hee, K.M., Sidorova, N., Somers, L.J., Voorhoeve, M.: Consistency in model integration. Data Knowl. Eng. **56**(1) (2006) 4–22
28. Mendling, J., Simon, C.: Business process design by view integration. In: BPM Workshops, LNCS 4103, Springer (2006) 55–64
29. Recker, J.: Process Modeling in the 21st Century. BPTrends **3**(5) (2006) 1–6
30. Bernhard, J., Jodin, D., Hömberg, K., Kuhnt, S., Schürmann, C., Wenzel, S.: Vorgehensmodell zur Informationsgewinnung Prozessschritte und Methodennutzung, Technical Report 06008. Sonderforschungsbereich 559, Modellierung großer Netze in der Logistik, Universität Dortmund (2007)
31. Hömberg, K., Jodin, D., Leppin, M.: Methoden der Informations- und Datenerhebung, Technical Report 04002. Sonderforschungsbereich 559, Modellierung großer Netze in der Logistik, Universität Dortmund (2004)
32. Scheer: IDS Scheer: ARIS Process Performance Manager. http://www.ids-scheer.com.

Realtime Detection and Coloring of Matching Operator Nodes in Workflow Nets

Andreas Eckleder
Nero Development & Services GmbH, Karlsbad, Germany
aeckleder@nero.com
Thomas Freytag
Cooperative State University Karlsruhe, Germany
freytag@dhbw-karlsruhe.de
Jan Mendling
Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de
Hajo A. Reijers
Eindhoven University of Technology, The Netherlands
h.a.reijers@tue.nl

Abstract

This work describes the implementation of an algorithm to identify and colorize matching split/join-operator pairs in workflow net based process models within the open source software WoPeD [1]. The concept was suggested as a powerful means to enhance the understandability of process graphs in [2]. The implemented detection and coloring method works in realtime, i. e. process designers get immediate feedback on actual or intended editing activities.

1 Introduction

The understandability of process graphs is a key requirement for successful visual process modelling results. In [2, 3] it was investigated how the understandability of workflow nets can be supported by several methods. One of them is to assign colors to matching pairs of control flow operators (splits and joins). The approach makes use of the fact that colors are recognized and associated with a specific semantics faster than other elements of visualization.

For a given pair of nodes in a workflow net, the number of independent paths leading from the one node to the other can be calculated with the max-flow/min-cut algorithm of Ford and Fulkerson [4]. This approach is able to determine all P/T and T/P handles of a given workflow net and therefore suitable to prove whether wellhandledness applies or not. In particular the techniques for finding matching operator nodes in a workflow net can also be applied to find mismatching operator nodes and thus can help to perform structural analysis, e. g. to check the existence or the violation of well-handledness.

In the following sections, the algorithm for performing the required check will be introduced along with its formal prerequisites. Afterwards, an implementation in the open source product WoPeD [1] will be sketched and demonstrated. Finally, a conclusion will be given with ideas to enhance the proposed techniques.

2 Approach

Our definition of a pair of matching nodes is a generalization of the concept of PT/TP-handles as used to define well-handledness in [5, 6].

In a well-handled WF-net PN, two nodes x and y are called *matching operator nodes* iff

- x is an AND-split and y an AND-join or x is an XOR-split or a place, and y an XOR-join or a place
- there is a pair of elementary paths C1 and C2 leading from x to y such that: $\alpha(C_1) \cap \alpha(C_2) = \{x, y\} \Rightarrow C_1 \neq C_2$.

The Ford and Fulkerson algorithm can be used to verify that there are indeed at least two elementary paths leading from a given node x to another node y. This can be done in analogy to the approach described in [5] to detect PT and TP handles. However, to detect all matching operator nodes of a given workflow net, all pairs of nodes $\{n_1, n_2\} \in (A_s \times A_j) \cup (X_s \times X_j) \cup (X_s \times S) \cup (S \times X_j) \cup (S \times S)$ where $A_{s/j}$ stands for the nodes of type AND-split/join respectively and $X_{s/j}$ stands for the nodes of type XOR-split/join respectively, must be checked. As only nodes with at least two elements in their postset can serve as a split and only nodes with at least two elements in their preset can serve as a join, we limit our selection of pairs to the combinations of $\{n_1, n_2\}$ where $|n_1 \cdot| > 1$ and $|\cdot n_2| > 1$. Whenever the max-flow / min-cut algorithm is reporting a maximum flow > 1 for any given pair of nodes, that pair is marked as a matching operator node. Once all matching operator pairs of a given workflow net are detected, their graphical representation can be colorized in a suitable way in order to stress the semantical relation between them. Figure 1 shows a simple example of the coloring algorithm applied to a single AND-split/join handle.

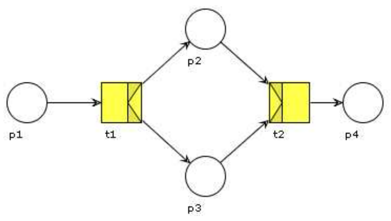


Figure 1: Simple coloring example

If multiple distinct handles exist in the same net, each matching operator node pair is assigned an individual color (see figure 2). When assigning colors to operator node pairs, it must be considered that assigning a node to a given matching operator pair is not mutually exclusive, so a given node can be part of more than one match.

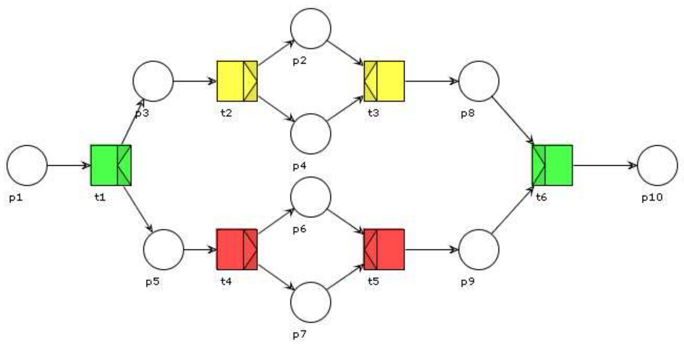


Figure 2: Example of distinct matching pairs marked in individual colors

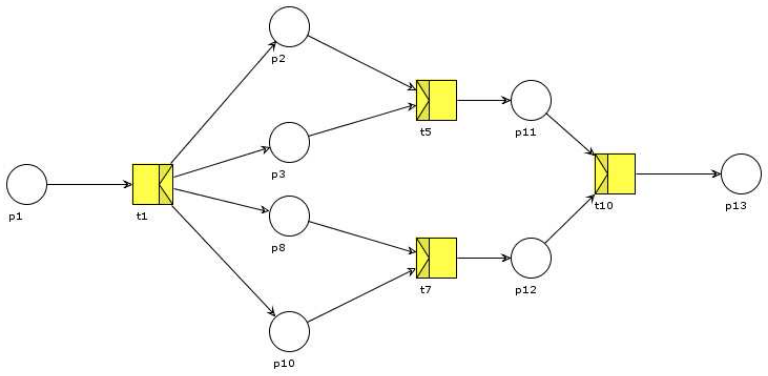


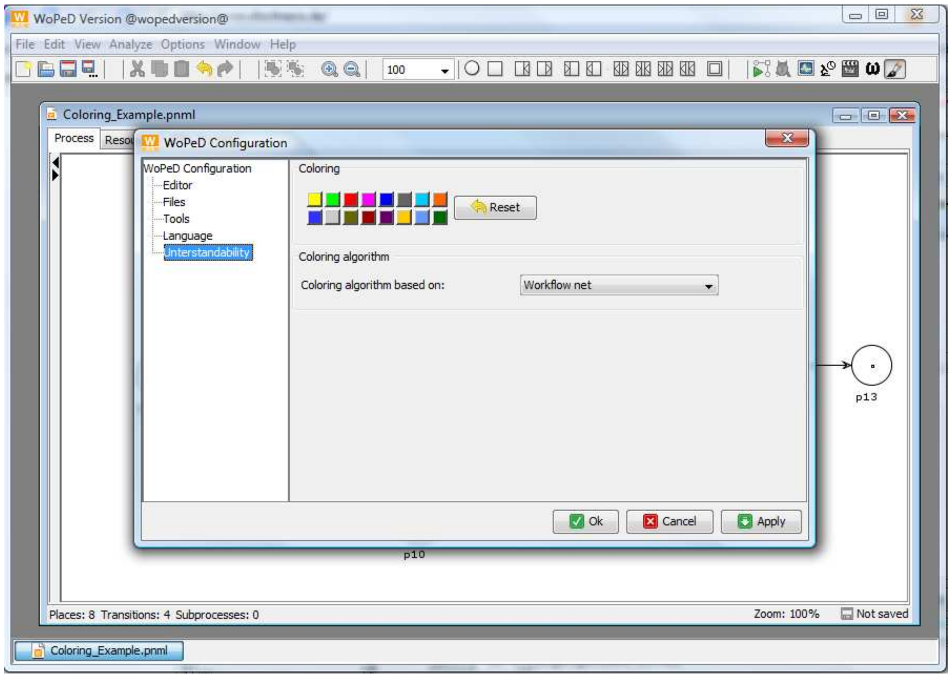
Figure 3: Handle clustering example

Since only one single color can be assigned to each node at a time, a way must be found to determine a common color for operator nodes that are part of more than one matching pair. This is done by building node clusters from the list of matching operator node pairs, where a given cluster contains all nodes of all pairs sharing at least one common node. Figure 3 shows an application example of this clustering algorithm, resulting in the same color being used for multiple matching operator handles *{t1,t7}*, *{t1,t5}* and *{t1,t10}*.

3 Implementation

WoPeD is an open source, Java-based graphical editor for workflow nets supporting the well-established "van der Aalst" notation[6]. The tool is maintained via Sourceforge, a common platform for the distributed development of free software projects. Several publications have accompanied the emerging development of WoPeD [7, 8, 9]. In the newest release which is obtainable on the

WoPeD website [1], coloring can be enabled or disabled simply by an assigned toggle button on the toolbar. When coloring is switched on, each cluster of matching operators is assigned one of the colors from a selection palette. The palette itself can be created within a settings dialog (see figure 4) and filled with arbitrary color values.



Coloring algorithm based on: Workflow net

Figure 4: A settings dialog allows the configuration of optical appearance

There is a special neutral color (usually white) that is used for all nodes that have not been identified as members of any pair or cluster by the algorithm. Their graphical representation matches that of standard nodes when the coloring feature is disabled.

In “enabled” mode, the workflow net graph is constantly monitored for user-inflicted changes. If a relevant change is detected, the coloring algorithm is executed, producing a possibly new set of node clusters. Each cluster receives an individual color from the palette until all colors are in use. If this happens, colors must be re-used or the palette must be extended. Finally, the visual representation of the workflow net is updated using the new colors. To enhance the visual feedback of correct modelling, only node pairs that do not violate the rules of well-handledness are considered for coloring.

The coloring algorithm has been implemented on top of a simplified representation of the transformed workflow net. This transformed representation is derived from the original graph $G = (S, T, F)$ by inserting a first node n' and a second node n'' for each node $n \in S \cup T$, and then creating an arc connecting *n'* to *n"*. Once all nodes are processed this way, an arc connecting *x"* to *y'* is inserted for all arcs $(x, y) \in F$ of the original net. The implementation of the Ford and Fulkerson algorithm is derived from the one introduced in [10], with the modification to select nodes based on "breadth-first" search. The algorithm runs at polynomial time.

Building the node clusters whose individual nodes are sharing one color has been implemented by using a simple, iterative algorithm as follows:

1. Let A be a list of sets of nodes, each set consisting of one of the matching node pairs detected
2. While $a \cap b \neq \emptyset$ for any $a, b \in A$ with $a \neq b$, set $A = A \setminus \{a, b\} + \{(a \cup b)\}$

The exemplary workflow net shown in figure 3 shows a total of three operator node pairs, each with more than one distinct node paths leading from one to another. Each pair is added to an initial list of node sets:

1. *{t1, t7}*
2. *{t1, t5}*
3. *{t1, t10}*

In the first iteration, *{t1, t7}* and *{t1, t5}* are combined to *{t1, t5, t7}*. The second and last iteration combines *{t1, t5, t7}* and *{t1, t10}* to *{t1, t5, t7, t10}*. All nodes belonging to the same set of nodes are drawn with the same color.

4 Conclusion

One shortcoming of our approach is the fact that the number of colors a human can clearly distinguish from each other is fairly limited. A possible solution for this could involve the assignment of special patterns in addition to plain palette colors (e. g. hatched, striped or plaid). Such patterns could be used to extend the amount of distinguishable handle clusters for complex workflow nets with more existing clusters than palette color entries.

The coloring algorithm has been implemented in a sufficiently generic way as to allow its application to the generalized problem of detecting PT/TP-handles and thus control-flow errors in workflow nets. Our implementation therefore also replaces the structural workflow net analysis functionality of WoPeD, allowing PT/TP-handle detection without falling back to external tools as Woflan.

References

[1] WoPeD website: www.woped.org, accessed on Aug 2, 2009.

[2] M. D. Lara. Proano: Visual layout for drawing understandable Process Models. Master's thesis, Technische Universiteit Eindhoven, 2008.

[3] J. Mendling, H.A. Reijers, and Jorge Cardoso. What Makes Process Models Understandable? In G. Alonso, P. Dadam and M. Rosemann, editors, Proceedings of the 5th International Conference Business Process Management (BPM 2007), Lecture Notes in Computer Science 4714, 48-63. Springer Verlag, Berlin, 2007.

[4] L. R. Ford and D. R. Fulkerson: Maximal flow through a network, Canad. J. Math. 8 (1956), 399-404.

[5] H. M. W. Verbeek: Verification of WF nets. PhD dissertation, Technische Universiteit Eindhoven, 2004.

[6] W. M. P. van der Aalst and K. van Hee: Workflow Management: Models, Methods, and Systems, 2002.

[7] T. Freytag and S. I. Landes: PWFtool - a Petri net workflow modelling environment. AWPN 2003 - Research Report, Catholic University of Eichstaett, 2003.

[8] C. Flender and T. Freytag: Visualizing the Soundness of Workflow Nets. AWPN 2006 - Research Report, University of Hamburg, 2006.

[9] A. Eckleder and T. Freytag: WoPeD 2.0 goes BPEL 2.0. AWPN 2008 - Research Report, University of Rostock, 2008.

[10] T. Ihringer: Diskrete Mathematik: Eine Einführung in Theorie und Anwendungen, 2002.

Autorenverzeichnis