Albert Ludwigs University - Freiburg

Bachelor Thesis

Inter-Instance Constraints

Author:
Regina König

Supervisor:
Adrian Lange

A thesis submitted in fulfilment of the requirements
for the degree of Bachelor of Science

in the

Research Group Name
Department or School Name

June 2015

Declaration of Authorship

I, Regina KÖNIG, declare that this thesis titled, 'Inter-Instance Constraints' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Companies spend millions of dollars on firewalls, encryption and secure access devices, and it's money wasted, because none of these measures address the weakest link in the security chain."

Kevin Mitnick

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

Abstract

Faculty Name
Department or School Name

Bachelor of Science

Inter-Instance Constraints

by Regina KÖNIG

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Contents

List of Figures

List of Tables

Abbreviations

BPMN **B**usiness **P**rocess **M**odell (and) **N**otation

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

Chapter 1

Einleitung

1. Satz:

1.1 Motivation

TODO: Motivation, Beispiel, Konventionen in Arbeit, Gliederung der Arbeit
Um internen Betrug in Unternehmen zu verhindern, wird bei vielen Authorisierungsmodellen das separation of duty Prinzip beachtet. Dieses besagt, dass es bestimmte Aufgaben gibt, die nicht von ein un der selben Person erledigt werden können. Dazu gibt es verschiedene Ansätze, die ermögliche, Einschränkungen auf die Authorisierung zu definieren, die je nach Art bereits in der Entwurfphase des Workflows oder dynamisch während der Laufzeit erzwungen wird, indem sich die Informationen dynamisch aktualisieren (Hier ein paas Quellen angeben). Alelrdings erlauben die aktuellen Modelle nur Einschränkungen innerhalb eines Workflows, was aber nicht ausreicheng ist.
Betrachten wir folgendes Beispiel:

Um eine Zahlung zu tätigen, muss die Zahlung zuerst beantragt werden. Dieser Antrag muss zweimal geprüft werden. Sollte die Prüfung positiv verlaufen, wird der Antrag bewilligt und die Zahlung wird getätigt.

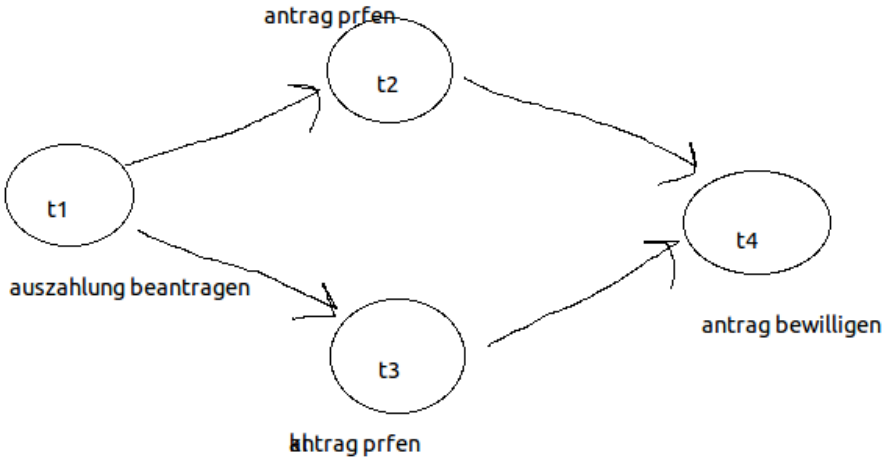

FIGURE 1.1: Beispiel Workflow

Es ist allerdings unerwünscht, dass t3 und t4 von der selben Person ausgeführt wird, die die Zahlung beantragt. Ferner sollten auch keine Verwandten dieser Person daran arbeiten, da die Bewilligung aus reiner Gefälligkeit erfolgen könnte.

Diese Einschränkungen, die sich jeweils auf einen Arbeitsablauf beziehen, sind aber nicht ausreichend, um Betrug zu verhindern. Es könnte sich eine Gruppe bilden, die sich in mehreren Instanzen des Arbeitsablaufs gegenseitig die Auszahlungen bewilligen. Eine zusätzliche Einschränkung wäre somit, dass eine Gruppe nur 5mal an t1 und t2 zusammen arbeiten darf.

1.2 Ziel der Arbeit

In dieser Arbeit wird eine Grammatik entwickelt, die es erlaubt, solche Einschränkungen zu definieren. Zusätzlich wird ein Programm vorgestellt, das MXML Logs einliest und sie auf die definierten Einschränkungen hin überprüft.

1.3 Aufbau der Arbeit

Folgende Konventionen werden hier eingehalten: kursive Begriffe bezeichnen Fachbegriffe, Blockschrift kennzeichnet Pseudocode,...

In Kapitel 2 werden wichtige Begriffe erläutert. In Kapitel 3 widmen wir uns der Herleitung von Einschränkungen und der Definition einer antsprechenden Grammatik. Diese wird in Kapitel 4 in ein Programm integriert, welches Event Logs auf die Verletzung von Einschränkungen prüft. Dazu wird der Algorithmus vorgestellt. Kapitel 5 und 6 schließen mit einer Untersuchung der Korrektheit und einem Ausblick in weitere Forschung.

Chapter 2

Theoretische Grundlagen

2.1 Erklärung der Begriffe

In diesem Kapitel werden wichtige Begriffe vorgestellt, die im Verlauf der weiteren Arbeit von Bedeutung sind.

2.1.1 Workflow

Bild zu Workflow UML Ein Workflow ist eine definierte Abfolge von abgeschlossenen Arbeitsaufgaben, den Tasks.

TODO: suche nach einer offiziellen Definition von workflows

2.1.2 Tasks

Tasks sind Aufgaben, die in sich abgeschlossen sind. Sie besitzen 3 verschiedene Zustände: Assigned, Started und Completed welche durch Events getriggert werden. Das MXML Modell unterstützt 13 verschiedene Events, die sich in ihrer Bennenung bei den einzelnen Programmen unterscheiden können. Events können auch feiner gegliedert sein, bzw es kann auch sein, dass nur eine Teilmenge davon verwendet wird. In dieser Arbeit gehen wir davon aus, dass es folgende Events gibt: started, assigned, completed, ...

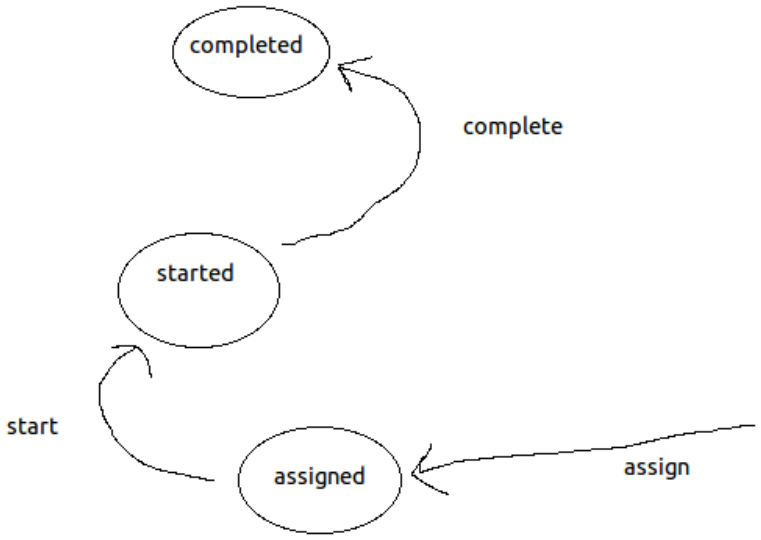

FIGURE 2.1: Tasks und Events

2.1.3 Authorisierung

2.1.4 Rollenmodell

2.1.5 Zeitmodell

2.1.6 Event Logs

EventLogs sind Abbildungen von Workflows. Ein EventLog kann durchaus unvollständig sein bzw Fehler beinhalten.

Das in dieser Thesis verwendete Format für Event Logs ist der Standard MXML

caseID	Task	User	Role	Timestamp	EventType
0	Approach check	'Mark'	'Admin'	1999-12-13T12:22:15	start
0	Pay check	'Theo'	'Azubi'	1999-12-13T12:22:16	start
1	Approach check	'Lucy'	'Azubi'	1999-12-13T12:22:17	start
1	Pay check	'Mark'	'Admin'	1999-12-13T12:22:18	abort
0	Revoke check	'Theo'	'Clerk'	1999-12-13T12:22:19	start

Das ist nur ein Auszug und kein vollständiger Log. Es können auch weitere Daten vorhanden sein, die hier nicht dargestellt werden.

TABLE 2.1: Beispiel Log Einträge.

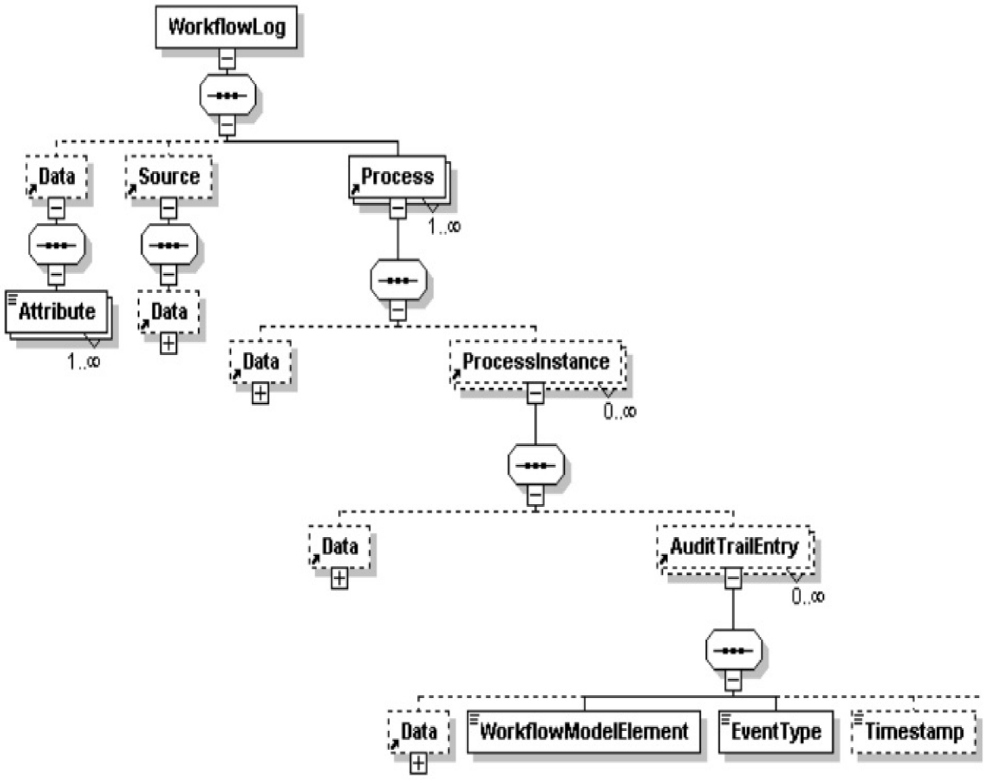

FIGURE 2.2: MXML Modell

2.1.7 Definition Constraints

2.1.8 Schutzziele

Die von Accorsi nehmen. Später darauf eingehen, was eingehalten wurde. Bzw schon bei Definition von Constraints darauf eingehen.

Chapter 3

Verwandte Arbeit

3.1 Related work

3.1.1 Geschichte

Chinese Wall Modell Bewer/ Nash
Business Rules und Auditierung ?? Um die Zuverlässigkeit von Abläufen zu gewährleisten, wird schon lange Auditierung betrieben. Ursprünglich ging es darum, den korrekten Ablauf nachzuweisen. Man kann damit aber auch die zulässige Ausführung von Ereignissen prüfen.

3.1.2 Heute

Übergang zu Inter/instance Constraints Heute: verschiedene Forschungszweige: TBAC, Inter-Instance Constraints Allgemein eher den Inhalt auswerten, statt eine Liste von verwandten Arbeiten anzugeben Arten Unterscheiden, wie man Constraints spezifizieren kann Als logische Prädikate, GL4 Welche Constraint-Richtungen werden behandelt? ZB nur Entailment, Cardinality.

3.1.3 warner inter-Instance

Darauf bezieht sich die Arbeit.
In dieser Arbeit wird der theoretische Ansatz von ... um inter instance constraints

erweitert und ist der Ausgangspunkt meiner Arbeit. Allerdings gehen sie davon aus, dass die Constraints für Authorisierungszwecke gedacht sind, und unterscheiden deswegen zwischen statischer Analyse zur Entwurfszeit und dynamischer Analyse während des Ablaufs. Bei der dynamischen Analyse werden die Informationen nach jeder Aktion aktualisiert und unter Umständen die Liste von verbotenen Nutzern / Rollen angepasst. Da ich Auditierung benutze, wird statically checked weggelassen Erklären, was ich anders mache

3.1.4 leitner instance-spanning

Hat ein Framework entwickelt.

3.1.5 tan consistency

Es geht eher um die Prüfung auf Konsistenz, es werden aber globale Constraints erwähnt.

3.1.6 ProM

3.1.7 SCIFF

Sciff wurde von Marco Alberti, Federico Chesani und Marco Gavanelli im Rahmen des SOCS Projekts entwickelt. Es arbeitet ebenfalls mit Prolog, kann aber keine Inter Instance Constraints.

Chapter 4

Material und Methoden

4.1 Theorie

4.1.1 Ein praktisches Beispiel mit vielen Einschränkungen

Bild zum Workflow + Erklärung

schriftliche Auflistung der Einschränkungen

-Intra-Instance

-Inter-Instance

–User U darf Task1 nicht mehr als 5 mal im Monat ausführen

–User U darf bei Task1 nicht mehr als 100000 Euro auszahlen

–U1 und U2 dürfen nicht mehr als 3mal an T1 und T2 zusammen arbeiten

-Inter-Process

–Ein User darf nicht mehr als 100 Tasks pro Tag erledigen

4.1.2 Arten von Constraints - Herleitung

zeitliche Beschränkungen –absolute Einschränkungen

–relative Einschränkungen

Akkumulationen

Separation of Duty

Binding of Duty Constraints

Cardinality Constraints

Workflow Soundness

–Bild dazu

4.1.3 Vorüberlegungen zur Grammatik

Woran erkennt man den Kontext der Constraints?

Welches Rollenmodell wird benutzt, und welche Auswirkungen hat es auf die Grammatik?

4.1.4 Formale Definition

Zuerst Herleitung, was wir genau an Aussagen brauchen

Workflows

Zeitstempel

Constanten

Variablen

Literale

Regeln (Constraints sind Schlussfolgerungen, die sich aus Vorbedingungen ergeben.

Fakten

Status (aus Event Logs gelesen)

Ableitung,...

Zu Beispiel gefundene Einschränkungen definieren

Kann ein Task mehrere Instanzen haben?

Hierarchisches Rollenmodell

Umgang mit Negation

4.1.5 Grammatik in BNF

Konstanten sind als 'String' , Variablen ohne

Welche Zeichen darf man wo verwenden?

Verfügbare Literale

Syntax

Fakten: SET extern—workflow

Regeln: IF (..—..) (AND (..—..)*)

THEN (..—..)

WHERE t.name = t2.name AND ...

4.1.6 Erklärungen zur Grammatik

Wenn sich etwas auf jede Instanz einzeln beziehen muss, muss Task1.workflow.instance = Task2.workflow.instance

Eignet sich für hierarchisches und auch für normales Rollenmodell. Hängt nur davon ab, ob die Hierarchie als Fakt gesetzt wurde.

User und ihre Rollenzuweisungen müssen nicht explizit angegeben werden. Nur wenn man das für die Vergleiche braucht.

Beziehung zwischen critical task pair und collaborateur.

4.1.7 konkretes Beispiel

Die zuvor gefundenen Beispiele wären mit der neuen Grammtik:

(C1): IF NUMBER OF USER executed TASK IS N AND N ¿ 5 THEN USER cannot execute TASK

Chapter 5

Praxis

5.1 Implementierung

5.1.1 Algorithmus

Zuerst aus Logs Status-Prädikate auslesen.
Für jedes critical task pair entsprechende collaborateurs setzen.
Schleife über Regeln
–Die resultierenden Heads bestimmen und schauen, ob es für cannot do ein executed und für must do kein executed gibt-¿ dann wurde es verletzt. Um das herauszufinden, wird die Regel in executed $->$ blabla, oder not(executed) $->$ blabla übersetzt.

Einfaches Beispiel: Rule: executed(ui, t1)-¿ cannot do(ui,t2) und status: executed('tom', t1). Daraus wird abgeleitet: cannot do('tom', t2). Da es nicht in der DB ist, ist alles OK

5.1.2 Aufbau mein Programm

Das Programm liest mittels SEWOL MXML logs ein und übersetzt sie in Status Prädikate. Die Constraints werden ebenfalls eingelesen und in Regelprädikate übersetzt. Der Parser für die Grammatik wurde mit ANTLR erzeugt. Die eigentliche Untersuchung kann entweder durch ein Skript oder innerhalb des Javacodes gestartet werden.

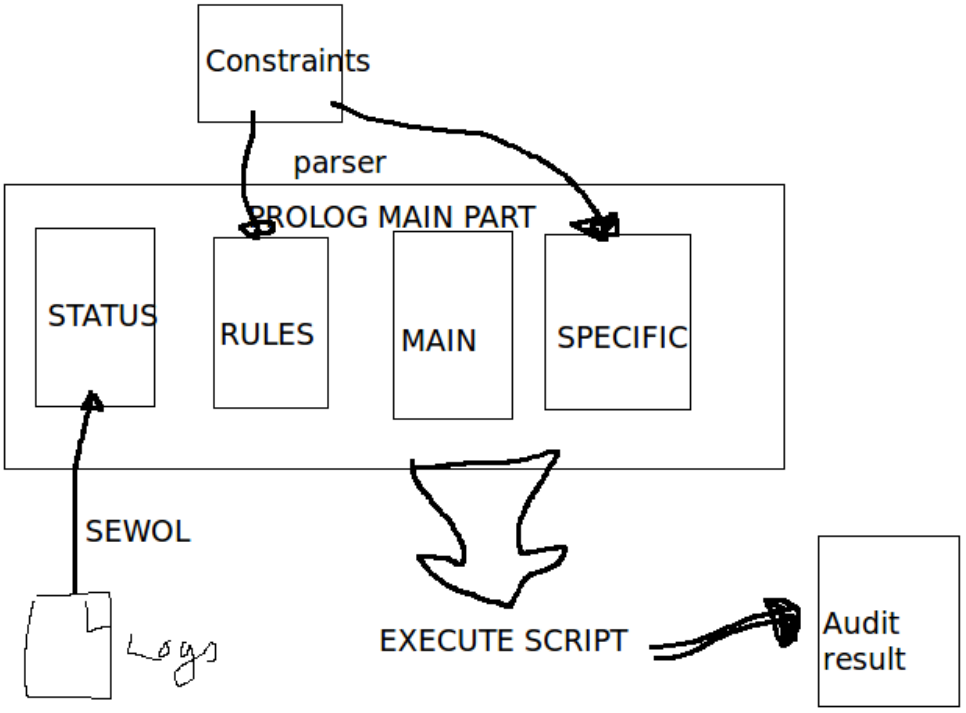

FIGURE 5.1: Aufbau mein Programm

Chapter 6

Ergebnisse und Diskussion

6.1 Auswertung und Aussicht

6.1.1 Mein Beispiel mit ein paar Logs untersuchen

Erklärung, woher diese Logs kommen und wie sie erzeugt wurden.

Zeigen, wie der Output aussieht.

Beweismethoden??

Wann gibt es false Negatives, false Positives?

Gibt es Fehler? Wieviele?

6.1.2 Aussicht

Wie kann man die restlichen Constraints implementieren

Compliance Checking:

Cannot do und Must execute vergleichen.

Optimierung??

Appendix A

Weitere Beispiele für die Benutzung der Grammatik

zusätzliche Beispiele zu Constraints und deren Darstellung

Intra Instance

Task1 und Task2 dürfen nicht vom selben User ausgeführt werden

$->$ IF USER executed 'Task1' THEN USER cannot execute 'Task2'

Inter Instance

Inter Process

Appendix B

Argumente und Konfiguration

Input Dateien für die Constraints: Wo sind sie? Wie kann man mehrere einlesen? (Dadurch sollen die Regeln wiederverwendbar werden)

Kann man vielleicht eine textdatei anlegen, wo die Inputfiles drinstehen? Input Dateien MXML - wo sind sie? Wird der ganze Ordner eingelesen oder gibt man ein bestimmtes File an?

Output Logger - Konsole oder File. Einstellung Level

Unterscheiden zwischen Parser Logger und Visitor Logger?

Konfigurationsdatei für die vorkommenden Events

Output Ordner

Parameter	Argument	Default	Beschreibung
–constraintfolder	abs oder rel Pfad zum Ordner	rulefiles	..
–logfolder	abs oder rel Pfad zum Ordner	logfiles	..
–outputfolder	abs oder rel Pfad zum Ordner	prologfiles	..

Tabelle mit allen möglichen Parametern

TABLE B.1: Kommandozeilenparameter

Appendix C

User Manual

Voraussetzung sind Java 7

SWI-Prolog 6

Per Shell Skript: ./run.sh

Appendix D

Fehlermeldungen

Write your Appendix content here.

Bibliography

[1] Eclipse BPMN2 Modeler User Guide. URL https://www.eclipse.org/bpmn2-modeler/documentation/BPMN2ModelerUserGuide-1.0.1.pdf.

[2] JBoss jBPM Team. jBPM v6.0 Documentation. URL http://docs.jboss.org/jbpm/v6.0/userguide/index.html.