Implementace procesní analýzy OR diagramů v metodě BORM

Oskar Maxa, MFF UK, 2015

Kontext

- oblast výzkumu: **modelování**, **analýza** a **simulace byznysových procesů** v metodě **BORM**
- procesy jsou reprezentovány pomocí **OR diagramu**
- přímá návaznost na dva články Podlouckého a Pergla z FIT ČVUT, jež:
  - shrnují dosavadní praxi a zavádějí konzistentní sémantiku **rozhodování** a **paralelismu** [1]
  - definují **prefixový automat** coby předstupeň formalizace OR diagramu a prostředek k jeho strukturální validaci [2]

Cíle práce

- vytvořit **formální model OR diagramu** na základě stávající implementace v aplikaci Open CABE vyvinuté na FIT ČVUT
- popsat **převod OR diagramu na prefixový automat,** a tím završit propojení stávající teorie a praxe
- navrhnout nový **simulační algoritmus** napravující významné sémantické nedostatky svých předchůdců
- algoritmus implementovat v rámci **webového portálu Oskar** pro správu a simulaci procesních diagramů

Kanonický OR diagram

Kanonický OR diagram je n-tice $(P, S, A, T, C, o, d, e^+, e^-)$, kde:

- P je množina **participantů**.
- S je množina **stavů**.
- A je množina **aktivit**.
- $T \subseteq \{(x, y) \mid x \neq y; x, y \in S \cup A\}$ je množina **přechodů**.
- $C \subseteq \{(x, y) \mid x \neq y; x, y \in A\}$ je množina **komunikací**.
- $o : S \cup A \to P$ je zobrazení přiřazující každému stavu a aktivitě participanta.
- $d : C \to \{true, false\}$ je zobrazení určující, zda je daná komunikace **přímá** nebo **nepřímá**.
- $e^+ : S \cup A \to E$ je zobrazení přiřazující každému stavu a aktivitě **vstupní podmínku** (pozitivní booleovský výraz nad přechody nebo $\top$).
- $e^- : S \cup A \to E$ je zobrazení přiřazující každému stavu a aktivitě **výstupní podmínku** (pozitivní booleovský výraz nad přechody nebo $\top$).
- $N = S \cup A$ nazýváme množinou **uzlů**.
- $H = T \cup C$ nazýváme množinou **hran**.

Dále jsou definovány následující množiny a zobrazení:

- $T_x^+ = \{t \in T \mid \exists y \in N : t = (y, x)\}$ je množina všech **příchozích přechodů** uzlu.
- $T_x^- = \{t \in T \mid \exists y \in N : t = (x, y)\}$ je množina všech **odchozích přechodů** uzlu.
- $deg^+ : N \to \mathbb{N}; deg^+(x) = |T_x^+|$ zveme **vstupní stupeň** uzlu.
- $deg^- : N \to \mathbb{N}; deg^-(x) = |T_x^-|$ zveme **výstupní stupeň** uzlu.
- $S_I = \{s \in S \mid deg^+(s) = 0\}$ je množina všech **počátečních stavů**.
- $S_F = \{s \in S \mid deg^-(s) = 0\}$ je množina všech **koncových stavů**.
- $C_x^+ = \{c \in C \mid \exists y \in A : t = (y, x)\}$ je množina všech **příchozích komunikací** aktivity.
- $C_x^- = \{c \in C \mid \exists y \in A : t = (x, y)\}$ je množina všech **odchozích komunikací** aktivity.

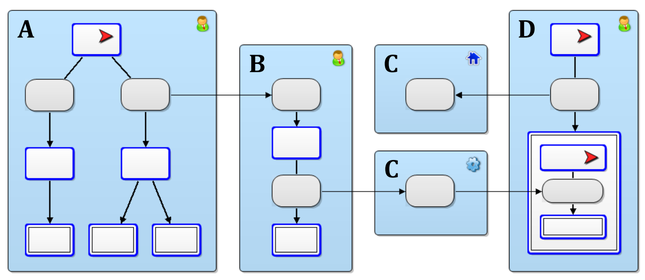
Platí následující omezení:

- $|P \cup S \cup A| = |P| + |S| + |A|$, čili množiny P, S a A jsou disjunktní.
- $|P| > 0$, čili diagram obsahuje alespoň jednoho participanta.
- $\forall p \in P : |\{x \in S_I \mid o(x) = p\}| = 1$, čili každý participant má právě jeden počáteční stav.
- $\forall p \in P : |\{x \in S_T \mid o(x) = p\}| = 1$, čili každý participant má právě jeden koncový stav.
- $\forall x \in A : deg^+(x) = deg^-(x) = 1$, čili každá aktivita má právě jeden příchozí a jeden odchozí přechod.
- (+ 8 dalších)

Převod na prefixový automat

1. Stavy OR diagramu -> stavy prefixového automatu.
2. Aktivity OR diagramu -> stavy prefixového automatu.
   Aktivity s přímými komunikacemi -> komunikační stavy prefixového automatu.
3. Nepřímá komunikace v OR diagramu -> přechod v prefixovém automatu.
4. Každé dva komunikační stavy spojené přímou komunikací sloučíme v jeden. Postup opakujeme, dokud se nezbavíme všech komunikací.
5. Přidáme speciální koncový stav, jehož vstupní podmínka je konjunkce přes koncové stavy všech participantů, kteří musejí proces dokončit.

Obecný OR diagram



Obecný OR diagram může oproti kanonickému obsahovat řadu dalších konstruktů. Jednoznačným postupem jej lze převést na kanonický.

A ... participant s více koncovými stavy
B ... spouštěný participant
C ... servisně orientovaný participant
D ... stav s vnořeným podprocesem

Simulační algoritmus

Navazuje na předchozí implementace:

- podpora pro širokou škálu konstruktů OR diagramu
- vysoká granularita simulačních kroků
- manuální i automatické (časované) krokování
- manuální zpětný chod simulace
- zobrazení diagramů v souladu s aplikací Open CABE

Vymezuje se vůči předchozím implementacím:

- sémanticky smyslupné chování ve spojovacích stavech
- omezení uživatelského rozhodování ve větvících stavech
- alternativní mód simulace bez iluze současnosti – manuální krokování v jednotlivých větvích procesu
- zobrazení navštívených a nenavštívených cest
- vizuální rozlišení přechodů, přímých a nepřímých komunikací

Algoritmus průběžně přiděluje **statusy** uzlům, hranám a participantům.
Ukázka pseudokódu (simulační krok pro stav *s*):

```
Krok(stav s):
Je-li Status(s) „připravený na podproces“:
Status(počáteční stav podprocesu) := „připravený započít přechod“
Status(s) := „čekající na podproces“
Je-li Status(s) „připravený započít přechody“:
Je-li výstupní podmínka stavu s konjunkce přes všechny odchozí
přechody:
    ZapočniPřechody(s, všechny odchozí přechody stavu s)
Jinak:
    Dej uživateli vybrat z povolených kombinací
    odchozích přechodů stavu s
    P+ := vybrané přechody
    P- := nevybrané přechody
    PropagujNenavštívenost(s, P-, {})
    ZapočniPřechody(s, P+)
Je-li Status(s) „připravený dokončit přechody“:
Pro všechny připravené odchozí přechody p stavu s:
    DokončiPřechod(Cíl(p))
```

Webový portál Oskar

Základní funkce

- práce s diagramy (zobrazení, simulace, komentáře)
- správa přístupu (uživatelé, skupiny, práva, zákazníci)

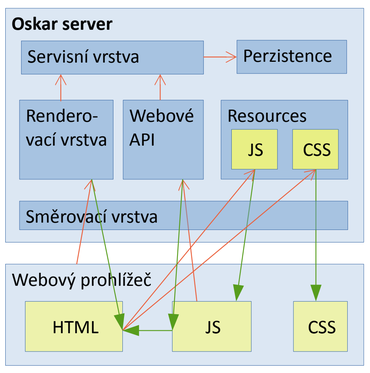
Základní vlastnosti

- multitenancy – obsluha více zákazníků, logické oddělení jejich dat
- podpora pro velké množství dat – stránkování, filtrování a řazení s podporou serveru
- bezpečnost – důsledná autentikace a autorizace, kvalitní šifrování
- jazyková podpora – kompletní lokalizace textu uživatelského rozhraní, rozšiřitelnost

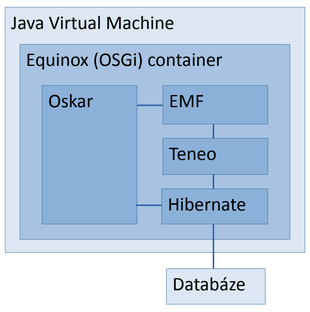
Hlavní použité technologie

- Clojure – programovací jazyk, dialekt LISPu na JVM
  - ClojureScript – varianta Clojure kompilovaná do JavaScriptu
- Eclipse – rodina technologií, rovněž stejnojmenné vývojové prostředí
  - Equinox – referenční implementace modulárního systému OSGi
  - EMF (Eclipse Modeling Framework) – definice datového modelu, generování kódu
- Hibernate, Teneo – objektovně-relační vrstva zajišťující převod datového modelu mezi Javou a SQL
- H2, MySQL – podporované relační databáze pro perzistenci dat aplikace
- Twitter Bootstrap – CSS framework určující značnou část podoby a funkčnosti klientské části aplikace
- Git – distribuovaný verzovací systém

Architektura



Perzistenční řešení



Reference

[1] Martin PODLOUCKÝ a Robert PERGL: Towards Formal Foundations for BORM ORD Validation and Simulation, ICEIS 2014, doi:10.5220/0004897603150322.

[2] Martin PODLOUCKÝ a Robert PERGL: The Prefix Machine – a Formal Foundation for the BORM OR Diagrams Validation and Simulation, EOMAS 2014, ISBN 978-3-662-44859-5.