Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE24Markovic

INTEGRACIJA NEWWAVE WORKFLOW ENGINE-A I OPENPONK ALATA

NEWWAVE WORKFLOW ENGINE AND OPENPONK TOOL INTEGRATION

Milica Marković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljena integracija NewWave workflow engine i OpenPonk alata. Model je nacrtan u OpenPonk editoru za meta-modelovanje. Plugin je zadužen za parsiranje modela u jeziku specifičnom za domen i mapiranje na NewWave elemente.*

Ključne reči: *Workflow engine, Pharo, NewWave, OpenPonk*

Abstract – *This paper presents integration of NewWave workflow engine and OpenPonk tool. The model is provided using OpenPonk tool for metamodeling. Plugin is in charge for reading domain specific language of model and for mapping in NewWave elements.*

Keywords: *Workflow engine, Pharo, NewWave, OpenPonk*

1. UVOD

Razvojem tehnologije, pre same izrade poslovnog procesa, neophodno je modelovanje elemenata poslovnog procesa. Modelovanje poslovnih procesa - *Business Process Modeling,* (BPM) je aktivnost u kojoj se predstavljaju (specificiraju) poslovni procesi nekog preduzeća na način da se mogu analizirati, poboljšavati i automatizovati. Odnose se na postojeće ili buduće (poboljšane) poslovne procese. Modeli tipično definišu:

- ko su korisnici (spoljni akteri),
- šta su ulazi i izlazi,
- koje su aktivnosti,
- način odvijanja poslova (*workflow* - tok izvršavanja),
- ko ih obavlja (unutrašnji akteri) [1].

Kao proizvod ove ideje, nastala je specifikacija takozvanog Workflow Engine. Standardizacijom i implementacijom Workflow engine-a, dobija se prednost u mogućnosti jednostavnog modelovanja aktivnosti poslovnih procesa. Workflow engine je softverska aplikacija koja upravlja poslovnim procesima [2]. Takođe, upravlja i nadgleda stanje aktivnosti u poslovnom toku. Akcije u samom toku mogu biti bilo šta, od čuvanja forme za apliciranje u *Document management system* do slanja kružnog mejla korisnicima. Workflow engine-i olakšavaju upravljanje tokom informacija, zadacima i događajima.
Pharo, objektno-orijentisan programski jezik, poseduje veliki potencijal za implementaciju i podršku mnogim poslovno-orijentisanim aplikacijama [3].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

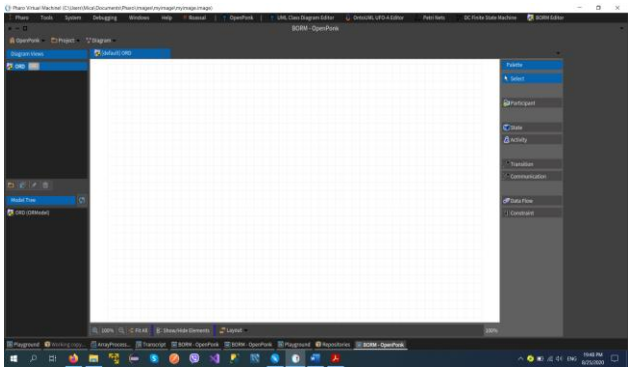
Pharo obezbeđuje jednostavnu, ali moćnu sintaksu i koncentrisan je na jednostavnosti i trenutnoj povratnoj informaciji.
Mnogi projekti zasnovani na Pharo programskom jeziku podržavaju pristup zasnovan na modelima (*Model-Driven approach)* i dinamički pristup svim fazama razvoja softvera: specifikaciji, dizajnu i implementaciji. U nedostatku široko prihvaćene podrške za modelovanje i implementaciju toka rada, nastao je NewWave workflow engine, sa idejom da uvede podršku za workflow rad u Pharo-u i omogući jendostavno upravljanje procesima [4].
OpenPonk je besplatna, otvorenog koda platforma za konceptualno modelovanje dijagrama, jezika specifičnih za domen i algoritama operacija nad modelima i dijagramima, kao što su transformacije i validacije modela, automatski raspored elemenata modela [5]. OpenPonk je implementiran u Pharo okruženju i razvijen pod MIT licencom. Razvijeno je okruženje koje omogućava grafičku vizualizaciju, interakciju grafičkih objekata, perzistenciju, raspored i grafički korisnički interfejs. OpenPonk platforma nudi dizajnerima modela alat koji rešava modelovanje notacije, specifične algoritme, transformacije modela, simulacije, itd. [6].
NewWave je workflow engine otvorenog koda napisan u Pharo-u. NewWave omogućava korisnicima da specificiraju kompoziciju workflow-a. To je omogućeno kroz procese i aktivnosti kao različiti tipovi zadataka i redosled tih aktivnosti. Svaki tok je vizualizovan korišćenjem Roassal biblioteke. NewWave primenjuje TaskIt biblioteku koja omogućava korišćenje zadataka u Pharo, sa apstrakcijom da izvršava i sinhronizuje konkurentne zadatke [7].
Cilj ovog rada je da pokaže kako na osnovu specifikacije meta-modela u alatu OpenPonk koji korisnik sam napiše, pozvati NewWave workflow engine da izvrši preces nacrtan u editor.

2. SPECIFIKACIJA SISTEMA

Motivacija za izradu ovog rada je da se na osnovu modela procesa specificiranog u alatu OpenPonk izvrši NewWave proces sa svojim elementima BPMN notacije.

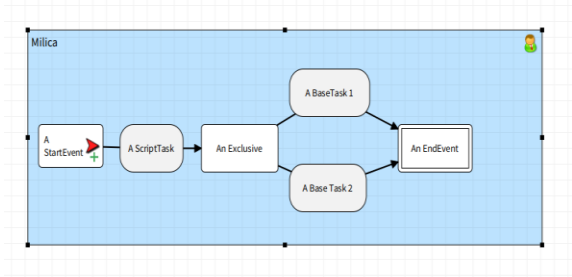
Iako korisnik kreira modele primarno kroz interfejs za dijagrame, postoje još načina za kreiranje modela: importovanje iz drugih alata (preko formata za razmenu, kao što je XMI), transformacija iz drugačije prezentacije (re-inženjerstvo programskog koda), ručno kreiranje potrebnih klasa (preko prorgamabilnog API-a), korišćenjem jezika specifičnih za domen, itd. Izbor odgovarajućeg načina zavisan je od konteksta, stoga će mnogi alati ponuditi različite opcije korisnicima. Kako bi ovo bilo omogućeno korisnicima, u OpenPonk je uveden

jezik specifičan za domen za meta-modele – BORM. Na slici 1 prikazan je BORM editor.



Slika 1. BORM okruženje

BORM model ima vizuelnu notaciju, a moguće je i koristiti tekst za opis modela procesa. Za potrebe izmena modela preko jezika specifičnih za domen, u OpenPonk je uveden DSL editor. Ovaj editor je direktno povezan sa trenutno otvorenim Workbench editorom i korisnik može vršiti izmene u modelu kroz ovaj editor; kada se tekst sačuva u editoru, model se promeni i kada se editor osveži na osnovu promena u modelu, promeni se i tekstualna prezentacija modela. Na slici 2 prikazan je primer modela procesa nacrtan u BORM OpenPonk Activity radnoj površini.



Slika 2. Model NewWave procesa

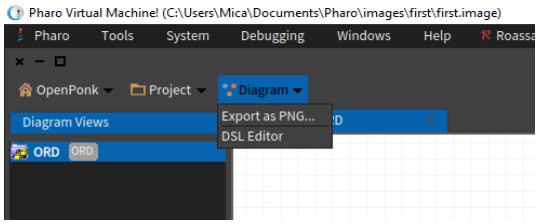
U primeru iz ovog poglavlja, proces započinje StartEvent događajem, koji je u BORM modelu označen sa *initial state* (znak plus sa slike 2). Nakon StartEvent-a sledi jedan ScriptTask u kome je definisana skripta koju on izvršava, i u ovom konkretnom primeru to je „10 atRandom“. Nakon ovog zadatka, sledi An Exclusive gateway-grananje, nakon koga slede dva BaseTask-a, BaseTask 1 i BaseTask 2. U Exclusive gateway, definisani su uslovi – *condition*, koje zadatak pre grananja mora da ispuni kako bi se nastavio proces u jednom od dva BaseTask-a. U ovom konkretnom primeru je da je za izvršavanje prvog BaseTask neophodno da broj generisan u ScriptTask-u bude veći ili jednak 5, a uslov za BaseTask 2 je da generisani broj bude manji od 5. Nakon ova dva BaseTask-a, sledi An EndEvent kojim se završava proces. EndEvent ima oznaku *final*, po čemu se kao i StartEvent-*initial*, razlikuje od ostalih zadataka.

3. IMPLEMENTACIJA REŠENJA

Kada je proces kreiran, potrebno je otvoriti *Domain Specific Language* (DSL) editor procesa u meniju OpenPonk-a, kao što je prikazano na slici 3).

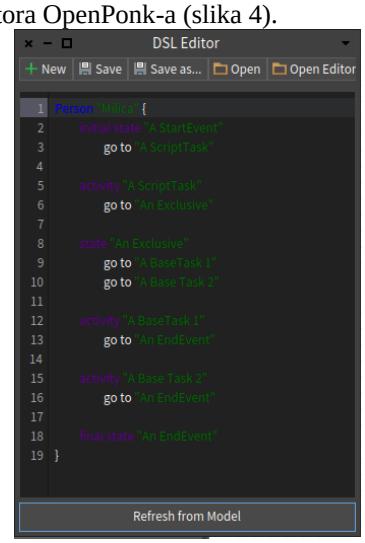
3.1 Kreiranje modela

Kada su nacrtani elementi procesa u Workbench editoru OpenPonk-a, potrebno je otvoriti DSL editor kako bi se napravio tekstualni model procesa, koji je potreban za dalji rad.



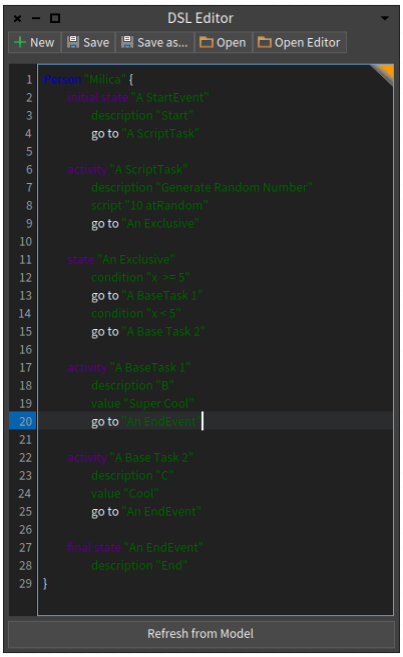
Slika 3. DSL editor

Kada se DSL editor otvori, neophodno je da se klikne na dugme *Refresh from model* na dnu prozora editora da bi se učitao model napisan u jeziku specifičnom za domen BORM editora OpenPonk-a (slika 4).



Slika 4. DSL editor sa modelom procesa

Pošto u modelu definisanom na ovaj način nema dovoljno podataka specifičnih za elemente procesa, potrebno je dodatno definisati atribute koji su usko vezani za elemente procesa. Da bi se engine pokrenuo, za minimalno izvršavanje potrebno je kreirati *Start* i *End Node* i sekvencu koja ih povezuje. Na slici 5 prikazan je tekstualni model procesa sa dodatnim obeležjima.



Slika 5. Model NewWave procesa

3.2 Obrada modela procesa

Nakon što je fajl sačuvan, potrebno ga je učitati, prepoznati elemente procesa i napraviti tok izvršavanja NewWave procesa. Za učitavanje fajla zadužena je klasa *LoadingFile* i njena metoda *loadDslFile*, koja za argument prima string putanju do mesta gde se nalazi fajl, prikazani na listingu 1.

```
loadDslFile: path

        | anArray |
        anArray   :=   path   asFileReference
readStreamDo: [ :in |
                Array streamContents: [ :out |
                [ in atEnd ] whileFalse: [ out
nextPut: in nextLine ] ] ].
        ^ anArray.
```

Listing 1. Metoda za učitavanje fajla

Zatim, potrebno je razgraničiti model po elementima koji se u njemu nalaze. Da bi se to uradilo, definisana je klasa klasa i njena metoda *numberOfElementsInDsl* za određivanje broja elemenata u procesu na osnovu broja praznih linija, prikazana na listingu 2.

```
numberOfElementsInDsl: stringArray

          | numberOfElements |
          numberOfElements := 0.
          2 to: (stringArray size-1) do: [ :i
|
                    (stringArray at: i) = ''
                              ifTrue: [
numberOfElements := numberOfElements + 1. ]
].
          numberOfElements := numberOfElements
+ 1.
        ^numberOfElements.
```

Listing 2. Metoda za računanje broja elemenata

Na osnovu dobijenog broja elemenata u modelu procesa, jedan primer implementacije je metoda koja kreira dvodimenzionalni niz elemenata čiji redovi će biti rezervisani za jedan element iz procesa, a kolone za obeležja tih elemenata, npr. za *description*, *script*, *value*, *condition*, *go to*, itd.

Metoda *splitArray* klase *ModelElement* prikazana je na listingu 3.

```
splitArray: stringArray numberOflementsDsl:
aNumberOflementsDsl
          | arrayOfArrays j k red |
          arrayOfArrays := Array2D new:
aNumberOflementsDsl.
          j := 1.   "row"
          k := 1.   "column"
          2 to: stringArray size - 1 do: [ :i
|
                    row := stringArray at: i.
                    row = ''
                              ifFalse: [
arrayOfArrays at: j at: k put: row.
                                        k := k
+ 1. ]
                              ifTrue: [ j := j
+ 1.
                                        k := 1.
] ].
        ^arrayOfArrays.
```

Listing 3. Metoda za računanje broja elemenata

Kada je model podeljen po redovim i kolonama matrice, sledeći korak bi bio dobijanje niza elemenata u kome bi bili nazivi elemenata iz modela. Na listingu 4 prikazana je metoda klase ArrayProcessing *getNamesOfElements*, čiji su parametri dvodimenzionalni niz dobijen u prethodnom koraku i broj elemenata u procesu dobijen u koraku ranije.

```
getNamesOfElements: array2d numberOfElements:
aNumber

          | arrayProcessed processElementName
processElementNameTokeni processElement |
          arrayProcessed := Array new:
aNumber.
          1 to: aNumber do: [ :i |
                    processElement := array2d
at: i at: 1.
                    processElementNameTokeni
:= processElement findTokens: '"'.
                    processElementName :=
processElementNameTokeni at: 2 .
                    arrayProcessed at:i put:
processElementName . ].
        ^arrayProcessed.
```

Listing 4. Metoda za dobijanje naziva elemenata

Kada je niz podeljen po redovima i kolonama, određen broj elemenata u procesu i dobijeni nazivi elemenata procesa, mogu se pozvati instance klasa NewWave procesa i napraviti tok procesa-workflow. Na listingu 5 prikazan je deo metode *processArray* klase *ArrayProcessing*. Rezultujući niz NewWave elemenata realizovan je kao OrderedDictionary kolekcija.

```
processArray: array2d numberOfElements:
aNumber array: anArray
          | k r se sc ee exclusive bt1 bt2
receiver1 message1 result1 condition1
condition2 keyword1 argument1 receiver2
result2 receiver3 keyword2 argument2 result3
resultArray me numberOfElementsInExclusive
bt3 receiver4 condition3 keyword3 argument3
result4 |
          k := 1.
          r := 1.
          resultArray := OrderedDictionary
new.
          me:=ModelElement new.
        numberOfElementsInExclusive:=me
numberOfElementsInExclusive: array2d
aNumber: aNumber .
```

Listing 5. Deo metode za određivanje NW elemenata

Na listingu 6 prikazano je prepoznavanje *NWStartEvent* na osnovu naziva elementa iz tekstualnog DSL fajla.

```
1 to: aNumber do: [ :e |
(anArray at: e) = 'A StartEvent'
ifTrue: [ se := NWStartEvent new.
          se description: (((array2d at: r at:
k + 1) findTokens: '"') at: 2).
          resultArray at: 1 put: se.
          k := 1.
        r:=r+1].
```

Listing 6. Prepoznavanje NWStartEvent elementa

Na osnovu naziva elementa procesa, poziva se instance klasa NewWave procesa. Atributi NewWave elementa se dobijaju iz tačno definisanih pozicija u dvodimenzionalnom nizu. Pravljenje sekvence elemenata je implementirano kroz kolekciju *OrderedDictionary* i tako što se

trenutni element u procesu obrade dodaje kao sledeći element prethodnom elementu u rezultujućoj sekvenci. Ukoliko su prisutna grananja, u primeru procesa iz ovog rada to je *Exclusive gateway*, potrebno je izračunati broj elemenata nakon grananja i u tim elementima ispitati koji su sledeći i prethodni elementi u procesu.
U *ScriptTask*-u, blok koda u script atributu obrađen je preko obrazaca za unarne i *keyword messages*-poruke. U grananju je definisana provera uslova, a sami uslovi u task-ovima. Kada su prepoznati NewWave elementi i napravljena sekvenca elemenata, NewWave proces je spreman za izvršavanje. Važan element za izvršavanje procesa je *StartEvent*.

3.3 Izvršavanje procesa

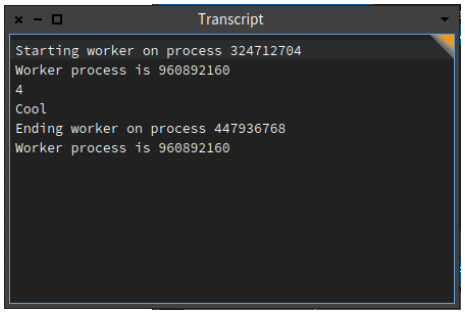
Za samo izvršavanje NewWave procesa definisana je klasa *ProcessFlow* i metoda klase *makeFlow* koja ima parametar niz NewWave elemenata koji je dobijen izvršavanjem metode *processArray* klase *ArrayProcessing*, prikazano na listingu 7.

```
makeFlow: anArray

          | process engine |
          process := NWProcess
                    id: '1'
                    name: 'process1'
                    initialFlowElement:
(anArray at:1).
          engine := WaveEngine new.
          engine addProcess: process name:
'process1'.
          engine startProcess: 'process1'.
        ^anArray.
```

Listing 7. Metoda za izvršavanje procesa

ProcessHandler je zadužen za procese i kreira se kada je proces prosleđen engine-u – korak: *engine addProcess:*. WaveExecutor je sam izvršilac procesa i neophodan mu je element od koga će započeti izvršavanje procesa. U primeru u radu *initialNode* je StartEvent. Jedan proces može imati više izvršilaca, jedan je glavni, a drugi se kreiraju po potrebi. Rezultat izvršavanja procesa prikazan je u Transcript prozoru Pharo okruženja, prikazano na slici 6.



Slika 6. Rezultat izvršavanja NewWave procesa

4. ZAKLJUČAK

U ovom radu detaljno je opisan proces integracije alata OpenPonk i NewWave procesa, kao i svi problemi i njihova rešenja nastala tokom implementacije. Prvobitno je diskutovano na temu specifikacije modela procesa u BORM editoru OpenPonk-a, a zatim i dodavanje dodatnih atributa modelu radi lakšeg prepoznavanja NewWave elemenata. Pored toga, predstavljen je način izvršavanja NewWave procesa na osnovu prepoznatih NewWave elemenata. Opisani postupak donosi povezivanje NewWave workflow engine-a koji je implementiran od strane tima profesora i asistenata sa Fakulteta tehničkih nauka u Novom Sadu i OpenPonk alata implementiranog od strane i Centra za konceptualno modelovanje i implementaciju sa Češkog tehničkog univerziteta iz Praga.

Nedostatak implementacije parsera sa sobom donosi dodatne obaveze u radu sa modelom procesa. Potrebno je voditi računa o memorisanju tekstualnog modela procesa iz DSL editora BORM dijagrama. Drugi nedostatak ovog pristupa rada jeste veliki broj ispitivanja i određivanja elemenata u procesu. Implementacija rešenja na ovaj način nije najbrža niti optimalna, ali je predložen jedan pristup implementaciji u ovom radu.

5. LITERATURA

[1] „Modelovanje poslovnih procesa“, http://mf.unibl.org/upload/documents/Dokumenti/Predmeti/Informacioni%20sistemi/3%20-%20Modelovanje%20poslovnih%20procesa%20(DFD%2C%20IDEF0%2C%20UML).pdf, pristupljeno: 8. jun 2020.

[2] „Workflow Engine“, https://en.wikipedia.org/wiki/Workflow_engine, pristupljeno: 8. juna 2020.

[3] „Pharo“, https://pharo.org/, pristupljeno: 23. jun 2020.

[4] „NewWave“, https://github.com/skaplar/NewWave, pristupljeno: 23. jun 2020.

[5] „OpenPonk“, https://openponk.org/, pristupljeno: 23. jun 2020.

[6] Peter Uhnak, Robert Pergl, „The OpenPonk modeling platform“, IWST’16, Prague, Czech Republic.

[7] Sebastijan Kablar, Miroslav Zarić, Gordana Milosavljević, „NewWave Workflow Engine“, IWST’19, Cologne, Germany, avgust 2019.

Biografija:

Milica Marković rođena je 21. septembra 1994. godine u Novom Sadu. Osnovnu školu „Ivo Lola Ribar” završila je 2009. godine. Gimnaziju „Jovan Jovanović Zmaj” u Novom Sadu završila je 2013. godine. Iste godine upisala je Univerzitet odbrane, Vojnu akademiju, smer Vojnoelektronsko inženjerstvo, modul Informacioni sistemi. 2017. godine završava pomenute osnovne akademske studije i iste godine upisuje master akademske studije na Fakultetu tehničkih nauka u Novom Sadu.