Softverski alat za automatsku vizuelizaciju UML dijagrama aktivnosti

Aleksandar Malešević
Network Manager d.o.o.
Banja Luka, Bosna i Hercegovina
amalesevic@networkmanager.ba

Dražen Brđanin, Slavko Marić
Elektrotehnički fakultet Univerziteta u Banjoj Luci
Banja Luka, Bosna i Hercegovina
bdrazen@etfbl.net, ms@etfbl.net

Sadržaj—**U ovom radu prikazan je jedan pristup za automatsku vizuelizaciju UML dijagrama aktivnosti i opisana implementacija softverskog alata, koji kao polazni osnov uzima XMI-zasnovanu reprezentaciju programski generisanog dijagrama aktivnosti i automatski generiše njegovu UMLDI-zasnovanu vizuelizaciju.**

Ključne riječi – UML; dijagram aktivnosti; XMI; UMLDI; vizuelizacija; Eclipse-Topcased

I. UVOD

Automatska vizuelizacija UML dijagrama aktivnosti [1], koji predstavlja često korišćenu notaciju za modelovanje toka aktivnosti u softverskim i poslovnim sistemima, nije dovoljno istražena, niti je adekvatno podržana u softverskim alatima za modelovanje. Postoji samo nekoliko radova posvećenih automatskoj vizuelizaciji dijagrama aktivnosti [2], odnosno modela poslovnih procesa [3]. Samo manji broj postojećih alata za modelovanje ima skromne mogućnosti automatske vizuelizacije programski generisanih dijagrama aktivnosti, dok većina uopšte i ne posjeduje tu mogućnost, što je posebno karakteristično za platforme otvorenog koda.

Postoji nekoliko tipičnih situacija u kojima je potrebna automatska vizuelizacija programski generisanih dijagrama aktivnosti, a najčešće kada se dijagram aktivnosti generiše na osnovu tekstualne specifikacije funkcionalnih sistemskih zahtjeva (npr. [4]) ili na osnovu ekvivalentnog modela reprezentovanog drugačijom notacijom (npr. BPMN u [5]).

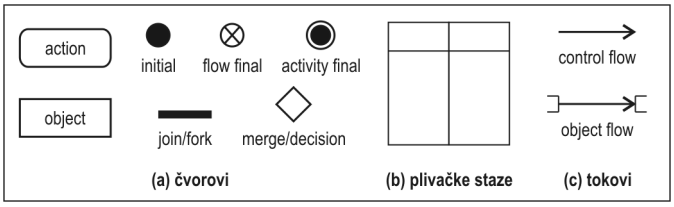
U ovom radu prikazan je jedan pristup za automatsku vizuelizaciju programski generisanog dijagrama aktivnosti i opisan softverski alat koji je implementiran kao *plug-in* u Eclipse-Topcased [6] razvojnom okruženju. Implementirani alat kao polazni osnov uzima XMI-zasnovanu [7] reprezentaciju dijagrama aktivnosti i automatski generiše njegovu vizuelizaciju zasnovanu na UMLDI specifikaciji [8].

Rad je organizovan na sljedeći način. Nakon uvodnog dijela, u drugom dijelu dat je pregled notacije UML dijagrama aktivnosti, koja je trenutno podržana u implementiranom alatu za automatsku vizuelizaciju. U trećem dijelu opisana je XMI-zasnovana serijalizacija, a u četvrtom dijelu UMLDI-zasnovana vizuelizacija programski generisanog dijagrama aktivnosti. Implementacija softverskog alata je prikazana u petom, a primjer automatske vizuelizacije u šestom dijelu. Na kraju su dati zaključci i pravci daljih istraživanja.

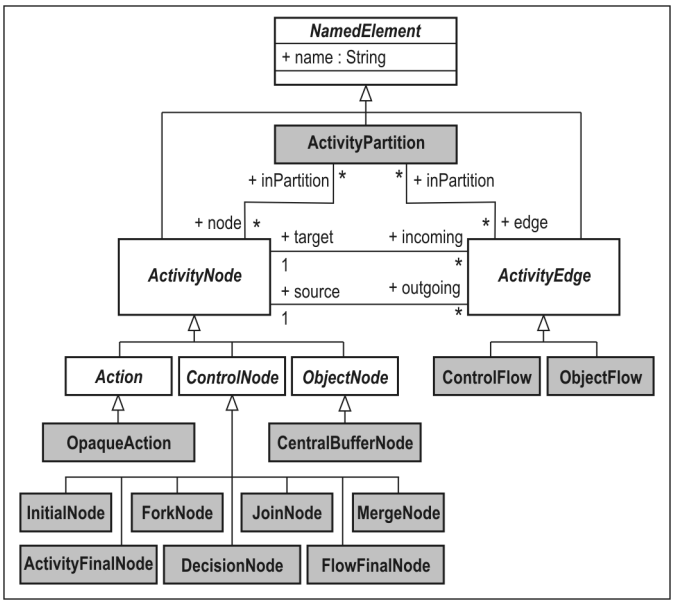
II. UML DIJAGRAM AKTIVNOSTI

Dijagram aktivnosti [1] pripada grupi UML dijagrama namijenjenih za modelovanje dinamičkih aspekata poslovnih i softverskih sistema. Iako je dijagram aktivnosti bio podržan i u inicijalnim specifikacijama (UML 1.x), originalna notacija je značajno sintaksno obogaćena i semantički modifikovana u UML 2.x, što je značajno proširilo mogućnosti njegove primjene. Prvobitno je semantika bila zasnovana na konceptu konačnih automata (i sama notacija tretirana je kao specijalni slučaj UML dijagrama stanja), a kasnije na *tokenima*.

Grafička notacija UML dijagrama aktivnosti, čija je automatska vizuelizacija trenutno podržana u implementiranom alatu, prikazana je na sl. 1, a njen metamodel na sl. 2.



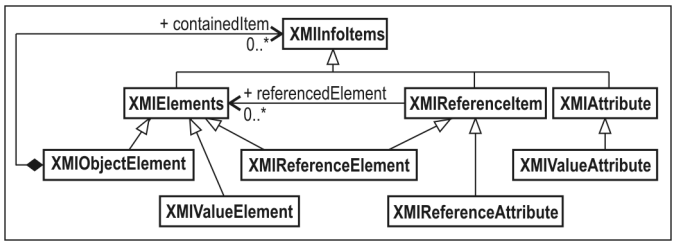
Slika 1. Notacija dijagrama aktivnosti koja je podržana u alatu.



Slika 2. UML metamodel [1] podržane notacije.

III. XMI-ZASNOVANA SERIJALIZACIJA

XMI (*XML Metadata Interchange*) [7] je industrijski (OMG) standard za XML-zasnovanu razmjenu metapodataka, koji se tipično koristi za serijalizaciju UML-zasnovanih modela. Metamodel XMI-zasnovane serijalizacije UML modela prikazan je na sl. 3.



Slika 3. Metamodel XMI-zasnovane serijalizacije [7].

Svaka akcija, objekat i kontrolni čvor u dijagramu aktivnosti reprezentuje se XMI *node* elementom, čiji *xmi:type* atribut ima odgovarajuću vrijednost (*uml:CentralBufferNode*, *uml:ObjectFlow, uml:OpaqueAction*, itd.). Osim standardnih atributa (*xmi:id* i *name*), svaki čvor može još da sadrži i identifikatore ulaznih (*incoming*) i izlaznih (*outgoing*) tokova, kao i druge atribute koji karakterišu dati čvor. Atribut *inPartition* sadrži identifikator plivačke staze kojoj dati čvor pripada. Primjer serijalizacije akcije, dat je na sl. 4.

```
<node xmi:type="uml:OpaqueAction"
      xmi:id="_x_NX8AC" name="Obrada"
      outgoing="_1KH0wAC" incoming="_0hfHcAC"
      inPartition="_rQe2IgC" />
```

Slika 4. Primjer XMI reprezentacije akcije.

Svaki tok u dijagramu aktivnosti reprezentuje se XMI *edge* elementom, čiji *xmi:type* atribut definiše vrstu toka (*uml:ControlFlow*, *uml:ObjectFlow*). Pored identifikatora i naziva, svaki tok još ima atribute *source* i *target* koji sadrže identifikatore čvorova koje dati tok povezuje. Svaki tok ima i *guard* atribut, čija vrijednost određuje mogućnost obilaska datog toka) te *weight* atribut, čija vrijednost specifikuje minimalan broj tokena koji mora istovremeno da prođe datim tokom. Primjer serijalizacije kontrolnog toka dat je na sl. 5.

```
<edge xmi:type="uml:ControlFlow"
      xmi:id="_0hfHcAC" name="Kontrolni tok"
      source="_t8voEAC" target="_x_NX8AC">
      <guard xmi:type="uml:LiteralBoolean"
             xmi:id="_0hfHcQC" value="true"/>
      <weight xmi:type="uml:LiteralInteger"
              xmi:id="_0hfHcgC" value="1"/>
</edge>
```

Slika 5. Primjer XMI reprezentacije kontrolnog toka.

Svaka plivačka staza u dijagramu aktivnosti reprezentuje se XMI *group* elementom. Pored tipičnih *xmi:id* i *name* atributa, svaka grupa može da ima i *node* atribut koji sadrži identifikatore svih čvorova koji pripadaju odnosnoj plivačkoj stazi. Primjer serijalizacije plivačke staze dat je na sl. 6.

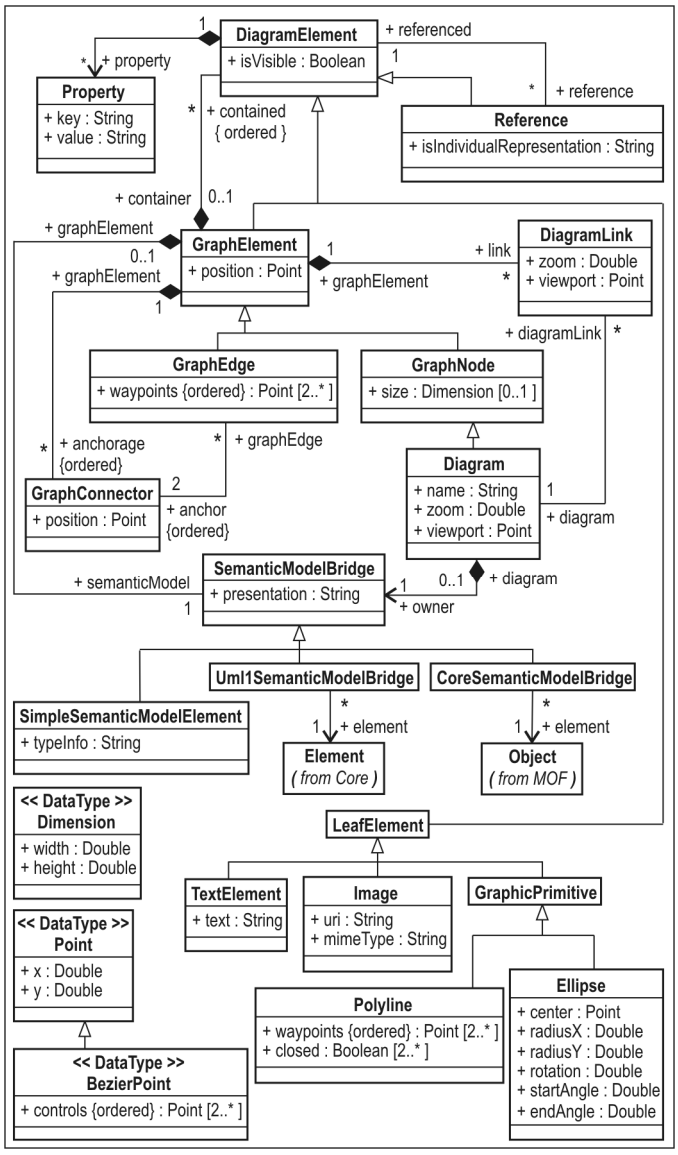
```
<group xmi:type="uml:ActivityPartition"
       xmi:id="_rQe2IgC" name="Bibliotekar"
       node="_t8voEAC _vIwvQAC _x_NX8AC"/>
```

Slika 6. Primjer XMI reprezentacije plivačke staze.

IV. UMLDI-ZASNOVANA VIZUELIZACIJA

Iako je široko prihvaćena u softverskim alatima za modelovanje, XMI-zasnovana reprezentacija UML modela ne ispunjava sve zahtjeve u pogledu prenosivosti modela, jer omogućava samo razmjenu informacija o sadržaju modela (elementi i njihove međusobne veze), ali ne i o njihovom rasporedu u odgovarajućoj grafičkoj reprezentaciji modela. Zbog toga prenos UML dijagrama, kreiranog nekim softverskim alatom, tipično rezultuje gubitkom vizuelizacije datog modela u drugom alatu. Ovaj nedostatak uticao je na stvaranje UMLDI (*UML Diagram Interchange*) specifikacije [8].

UMLDI metamodel (sl. 7) je slabo spregnut sa UML metamodelom (zavisi samo od dvije metaklase visokog nivoa: *Core::Element*, *Elements::Element*) i omogućava serijalizaciju dijagrama, tj. grafičke reprezentacije modela, nezavisno od sadržaja modela. Fajl koji sadrži serijalizovanu reprezentaciju dijagrama (*.umldi*), spregnut je sa fajlom koji sadrži serijalizovanu reprezentaciju sadržaja UML modela (*.uml*) samo vezama prema odnosnim elementima modela.



Slika 7. UMLDI metamodel [8].

UMLDI vizuelizacija zasniva se na grafovskoj interpretaciji UML dijagrama. Svaki vidljivi element dijagrama reprezentuje se instancom *GraphNode* ili *GraphEdge* klase.

Vizuelizacija čvorova i plivačkih staza reprezentuje se *di:GraphNode* elementima. Podelement *anchorage*, između ostalog, sadrži identifikatore grafičkih reprezentacija svih tokova povezanih sa datim čvorom, dok podelement *semanticModel* sadrži referencu prema odnosnom konceptu u UML modelu. Primjer *GraphNode* elementa dat je na sl. 8.

```
<contained xsi:type="di:GraphNode"
    xmi:id="_t8JLIAC"
    position="279,88" size="30,30">
  <anchorage xmi:id="_0hV9gAC"
    graphEdge="_0hV9ggC"/>
  <semanticModel
    xsi:type="di:EMFSemanticModelBridge"
    xmi:id="_t8JLIQC" presentation="default">
    <element href="Primjer.uml#_t8voEAC"/>
  </semanticModel>
</contained>
```

Slika 8. Primjer *GraphNode* elementa.

Vizuelizacija tokova reprezentuje se *di:GraphEdge* elementima. Atribut *anchor* sadrži identifikatore grafičkih reprezentacija čvorova povezanih datim tokom. Pored podelementa *semanticModel*, koji sadrži referencu prema odnosnom toku u UML modelu, postoji još nekoliko podelemenata koji opisuju grafičku reprezentaciju i poziciju toka u dijagramu. Primjer *GraphEdge* elementa dat je na sl. 9.

```
<contained xsi:type="di:GraphEdge"
     xmi:id="_0hV9ggC"
     anchor="_0hV9gAC _0hV9gQC">
  <semanticModel
     xsi:type="di:EMFSemanticModelBridge"
     xmi:id="_0hV9gwC" presentation="default">
    <element href="Primjer.uml#_0hfHcAC"/>
  </semanticModel>
  <contained xsi:type="di:EdgeObjectOffset"
             xmi:id="_0hV9hAC"
             id="stereotypeEdgeObject"/>
  ...
</contained>
```

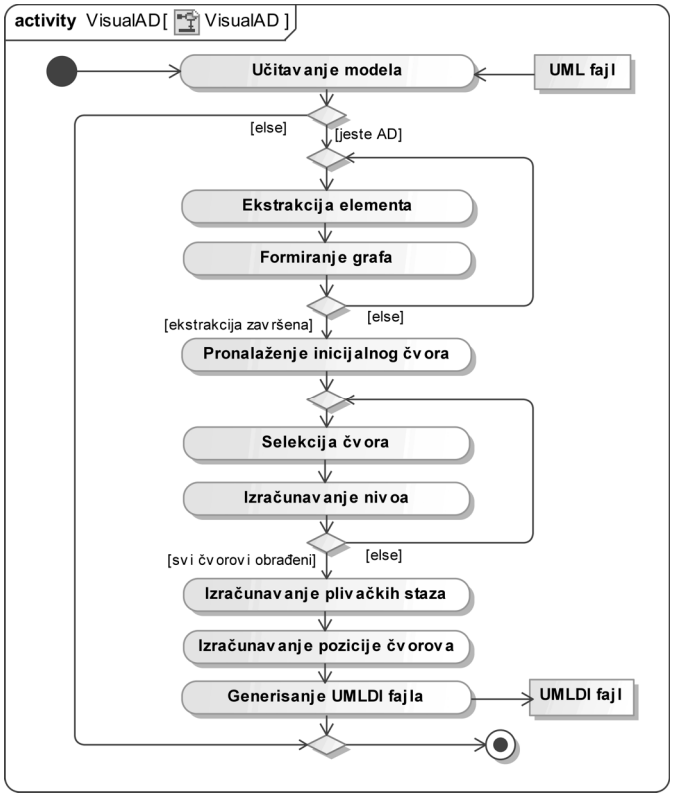
Slika 9. Primjer *GraphEdge* elementa.

V. IMPLEMENTACIJA

Implementirani alat procesira ulaznu, XMI-zasnovanu reprezentaciju dijagrama aktivnosti (*.uml* fajl) i automatski generiše UMLDI-zasnovanu vizuelizaciju (*.umldi* fajl). Algoritam rada implementiranog alata predstavljen je dijagramom aktivnosti na sl. 10.

Ulazni fajl pretprocesira se DOM-zasnovanim parserom uz formiranje odgovarajućeg usmjerenog grafa, korišćenjem *jGraphT* biblioteke [9].

Jedan od najvažnijih koraka u procesu vizuelizacije jeste identifikacija nivoa čvorova u dijagramu. Nivoi se koriste za vertikalno pozicioniranje čvorova unutar plivačkih staza. Inicijalni čvor pripada nultom nivou i referenca je za određivanje nivoa ostalih čvorova. Nivoi ostalih čvorova se predstavljaju pozitivnim cjelobrojnim vrijednostima. Algoritam za određivanje nivoa prikazan je na sl. 11.



Slika 10. Algoritam implementirane funkcionalnosti.

```
cvorovi_bez_incoming = empty;
for each cvor ci ∈ graf
  if (broj_ulaznih_tokova(ci) == 0)
    ci.nivo = 0;
    cvorovi_bez_incoming.add(ci);
  end if
end for
cvorovi_bez_nivoa = empty;
for each cvor ci ∈ graf
  if (for all parent(ci).nivo != -1)
    if (ci.inPartition != (for all parent(ci))
      ci.nivo = max(parent(ci).nivo);
    else
      ci.nivo = max(parent(ci).nivo)+1;
    end if
  else
    cvorovi_bez_nivoa.add(ci);
  end if
end for
while (cvorovi_bez_nivoa != empty)
  for each ci ∈ cvorovi_bez_nivoa
    if (for all parent(ci).nivo != -1)
      if (ci.inPartition != (for all parent(ci))
        ci.nivo = max(parent(ci).nivo);
      else
        ci.nivo = max(parent(ci).nivo)+1;
      end if
      cvorovi_bez_nivoa.remove(ci);
    end if
  end for
end while
for each ci ∈ cvorovi_bez_incoming
  ci.nivo = max(parent(ci).nivo)-1;
end for
```

Slika 11. Algoritam za određivanje nivoa čvorova u dijagramu.

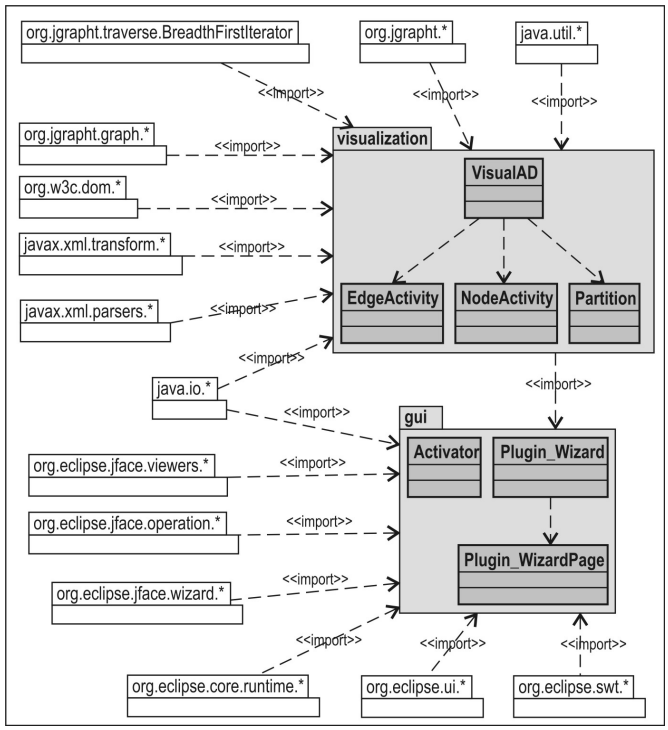
Nakon određivanja nivoa, za svaku plivačku stazu izračunavaju se pozicija i veličina (širina, visina), kao na sl. 12. Zatim se određuje pozicija svakog čvora. Prvo se izračunava relativna horizontalna pozicija tako da svi čvorovi na istom nivou budu uniformno raspoređeni po širini plivačke staze. Apsolutna pozicija svakog čvora izračunava se na osnovu pozicije plivačke staze, relativne horizontalne pozicije čvora i nivoa na kojem se čvor nalazi. Apsolutna vertikalna pozicija čvora dobija se kao proizvod nivoa čvora i podrazumijevane visine nivoa uz odgovarajući inicijalni pomjeraj, dok se apsolutna horizontalna pozicija dobija sabiranjem pozicije plivačke staze i relativne horizontalne pozicije čvora.

Na kraju se generiše *.umldi* fajl u skladu sa UMLDI specifikacijom.

```
for each partition pi
  pi.max_cvorova = max(ukupno_cvorova_na_nivou);
end for
for each partition pi
  pi.sirina = W_INIT + (pi.max_cvorova-1)*W_CVOR;
  pi.visina = H_INIT + (max_nivo * pos_y_STEP);
end for
for each partition pi
  pi.pozicija = pi-1.pozicija+pi-1.sirina+OFFSET;
end for
```

Slika 12. Algoritam za određivanje veličine i pozicije plivačkih staza.

Prethodno opisana funkcionalnost implementirana je kao Eclipse-Topcased [8] *plug-in* pod nazivom **VisualAD**, čija je arhitektura prikazana na sl. 13. Paket **gui** sadrži klase koje omogućavaju integraciju u Topcased platformu i implementiraju korisnički interfejs, dok paket **visualization** sadrži implementaciju procesa vizuelizacije.

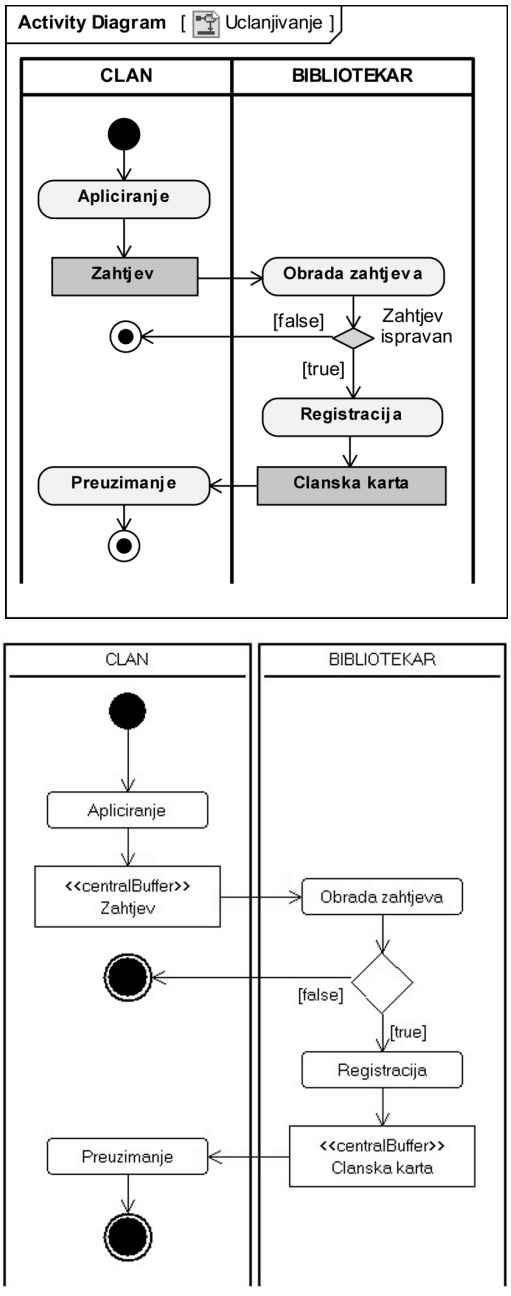


Slika 13. Arhitektura implementiranog VisualAD *plug-ina*.

VI. PRIMJER AUTOMATSKE VIZUELIZACIJE

Implementirani alat ima mogućnost automatske vizuelizacije XMI-zasnovane specifikacije dijagrama aktivnosti programski generisane u Topcased okruženju, ali i XMI-zasnovane specifikacije uvezene iz neke druge platforme za modelovanje.

U cilju ilustracije, na sl. 14 (gore) prikazan je manuelno kreiran dijagram aktivnosti u jednom komercijalnom alatu za modelovanje, a na sl. 14 (dolje) rezultat automatske vizuelizacije XMI-zasnovane specifikacije odnosnog UML modela, koja je uvezena u Topcased okruženje.



Slika 14. Manuelno kreiran dijagram aktivnosti u komercijalnom alatu (gore) i automatski generisani dijagram pomoću VisualAD alata u Eclipse-Topcased okruženju (dolje).

VII. ZAKLJUČAK

U ovom radu prikazan je jedan pristup za automatsku vizuelizaciju programski generisanog UML dijagrama aktivnosti i opisan softverski alat kojim je implementiran proces vizuelizacije. Implementirani alat, pod nazivom **VisualAD**, kao polazni osnov uzima XMI-zasnovanu reprezentaciju UML dijagrama i automatski generiše vizuelizaciju u skladu sa UMLDI specifikacijom. Proces vizuelizacije zasniva se na reprezentaciji dijagrama aktivnosti odgovarajućim usmjerenim grafom i segmentaciji grafa po vertikalnim nivoima. Alat je implementiran kao *plug-in* u Eclipse-Topcased platformi.

Pored mogućnosti automatske vizuelizacije programski generisanog UML dijagrama aktivnosti, implementirani alat ima i mogućnost automatske vizuelizacije dijagrama aktivnosti koji je kreiran u nekom drugom alatu za modelovanje, a čija je XMI-zasnovana specifikacija potom uvezena u Topcased okruženje. Na taj način se eliminišu ograničenja XMI specifikacije u pogledu prenosivosti vizuelizacije modela između različitih softverskih platformi, kao što je ilustrovano u ovom radu.

Nastavak istraživanja biće fokusiran na: (i) potpuno pokrivanje notacije dijagrama aktivnosti, i (ii) unapređenje vizuelizacije, odnosno otklanjanje nedostataka u pogledu presjecanja tokova, optimalno rutiranje tokova, eliminisanje preklapanja, itd.

LITERATURA

[1] Object Management Group (OMG), Unified Modeling Language (UML): Superstructure, v2.3. OMG, 2010.

[2] M. Siebenhaller, and M. Kaufmann, "Drawing activity diagrams", Univ. of Tübingen, Germany, Tech. Rep. WSI-2006-02, 2006.

[3] P. Effinger, M. Siebenhaller, and M. Kaufmann, "An Interactive Layout Tool for BPMN", Proc. of CEC '09, 2009, pp. 399-406.

[4] T. Yue, L. Briand, and Y. Labiche, "An Automated Approach to Transform Use Cases into Activity Diagrams", in ECMFA 2010, LNCS, vol. 6138, T. Kühne et al. (Eds.). Springer, 2010, pp. 337–353.

[5] A. Rodriguez, I. Garcia-Rodriguez de Guzman, E. Fernandez-Medina, and M. Piattini, "Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach", Information and Software Technology, vol. 52, pp. 945-971, 2010.

[6] The Topcased website. [Online]. Available: http://www.topcased.org/

[7] Object Management Group (OMG), MOF2XMI Mapping Specification, v. 2.4.1, OMG, 2011.

[8] Object Management Group (OMG), UMLDI, v. 1.0, OMG, 2006.

[9] The jGraphT website. [Online]. Available: http://jgrapht.org/

ABSTRACT

This paper presents an approach to automatic visualization of UML activity diagram, and provide the implementation of corresponding software tool which takes the XMI-based representation of a programmatically generated activity diagram and automatically generates its layout in accordance with UMLDI specification in Eclipse-Topcased environment.

SOFTWARE TOOL FOR AUTOMATIC VISUALIZATION OF UML ACTIVITY DIAGRAM

Aleksandar Malešević, Dražen Brđanin, Slavko Marić