Jedno rešenje implementacije asemblerskog uređivača na Java platformi uz pomoć Xtext okvira

Saša Kartalija, Ivan Letvenčuk, Jovica Grahovac, Jelena Kovačević, *Member, IEEE*

***Sadržaj* - U radu je opisana implementacija uređivača teksta (engl. *text editor*) za razvoj asemblerskog koda na DSP platformi, korišćenjem Xtext okvira za razvoj. Ovaj uređivač podržava specifične asemblerske instrukcije. Pored samog uređivača realizovan je i skup dodatnih alata koji u znatnoj meri olakšavaju korisniku korišćenje i razvoj asemblerskog jezika. Korisniku su na raspolaganju dodatna podešavanja samog uređivača, praćenje sadržaja uređivača, prepoznavanje istih instrukcija kao i automatski asistent koji na bazi predviđanja, korisniku nudi određene opcije pri razvoju jezika.**

Xtext pruža moderan API za opisivanje različitih aspekata DSL programskih jezika. Jezik razvijen u Xtext okviru je nezavisan od Eclipse razvojnog okruženja i može se koristiti u bilo kom Java okruženju.

***Ključne reči* — Xtext okvir, Eclipse, DSL, IDE, asembler uređivač, gramatika, Java**

I. Uvod

PROGRAMSKI jezici se prema širini oblasti primene mogu podeliti na dve osnovne kategorije:

1. Jezici opšte namene (engl. *General Purpose Language - GPL*), kao što su: JAVA, C i C++ [3].
2. Usko namenski jezici (engl. *Domain Specific Language - DSL*), kao što su: Xtext, SQL, HTML, Verilog, VHDL,

Jezici opšte namene se koriste za razvoj i održavanje širokog spektra algoritama na računarskim sistemima u čemu se i ogleda njihova prednost u odnosu na usko namenske jezike.

Ogromna primena ovih jezika je u izradi aplikacija za mobilne telefone, tablične računare, digitalne uređaje za reprodukciju video i audio signala, pa sve do kompleksnih web servera, baza podataka i korporativnih aplikacija.

Ovaj rad je delimično finansiran od Ministarstva prosvete i nauke Republike Srbije, projekat 32029.

Saša Kartalija, Fakultet Tehničkih Nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija (telefon: 381-64-2786187, e-mail: sasa.kartalija@rt-rk.com).

Ivan Letvenčuk, Fakultet Tehničkih Nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija (e-mail: ivan.letvencuk@rt-rk.com).

Jovica Grahovac, Fakultet Tehničkih Nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija (e-mail: jovica.grahovac@rt-rk.com).

Jelena Kovačević, Fakultet Tehničkih Nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija; (e-mail: jelena.kovačević@rt-rk.com)

I pored široke primene GPL jezika, određeni računarski sistemi i algoritmi zahtevaju drugačiji pristup razvoju programske podrške. DSL jezici, kao specijalizovani jezici za rešavanje specifičnih problema, obezbeđuju lakoću korišćenja, elegantnije, i rešenje koje u potpunosti ispunjava zahteve problema a istovremeno je jednostavno i razumljivo. SQL je najpoznatiji primer DSL jezika na kojem se ne neće uspeti napisati web aplikacija ali zato je za manipulaciju velikom količinom podataka nenadmašan.

Oblast primene jednog DSL jezika je praktično svedena na jedan specifičan problem. DSL jezik se od samog početka dizajnira i razvija za konkretan problem i upravo iz tog razloga ga nazivamo usko namenski jezik. Tako da za svaki DSL jezik koji je razvijen postoji tačno predodređena namena. Pošto je predodređen uskoj nameni od početka se definišu sintaksna i semantička pravila koja opisuju rešenje problema [4].

Da bi se problem uspešno rešio potrebno je učenje sintakse, koja može biti jako složena u nekim situacijama. Upravo to predstavlja, pored širine oblasti primene, još jedan od nedostataka ovog jezika. Glavni nedostatak ovog pristupa razvoju programske podrške je neprimenjivost na razvoj programske podrške pri rešavanju nekog drugog računarkog problema.

Zbog postojanja različitih ciljnih platformi od kojih su trenutno najpoznatije ARM (engl. *Advanced RISC Machine and Acorn RISC Machine*), MIPS (engl. *Microprocessor without Interlocked Pipeline Stages*)[2], javljaju se različiti tipovi sintakse asemblerskog jezika prilagođeni ciljnoj platformi. Upravo iz tog razloga za potrebe kompanije *Starkey[1]* ovim radom je razvijen uređivač teksta radi razvijanja nove sintakse asemblerskog jezika.

Da bi se od samog početka napisala specifična gramatika asemblerskog jezika, bilo je potrebno koristiti DSL jezik odnosno Xtext okvir u Eclipse razvojnom okruženju. Uređivač je deo Eclipse priključka koji omogućava:

1. Bojenje sintakse asemblerskog jezika (engl. *Syntax coloring*)
2. Prediktivni asistent (engl. *Code completion*)
3. Sadržaj uređivača sintakse (engl. *Outline view*)
4. Putanja kroz sadržaj uređivača (engl. *Source-code navigation*)
5. Indeksiranje (engl. *Indexing*)
6. Poređenje sadržaja (engl. *Compare view*)
7. Izmena imena u celokupnom sadržaju uređivača (engl. *Rename refactoring*)

Svaka od prethodno navedenih stavki je razvijena za ciljnu DSP platformu.

Rad je strukturiran na sledeći način: u drugom poglavlju su date teorijske osnove Eclipse razvojnog okruženja, kao i osnovni koncepti Xtext jezika, kao Eclipse priključka; u trećem poglavlju opisan je primer razvijenog jezika, kao i objašnjenje rešenja zadatka po koracima; četvrto poglavlje sadrži zaključak koji proizilazi iz ovog rada.

II. RAZVOJNO OKRUŽENJE

Integrisano razvojno okruženje poznato i kao IDE (engl. *Integrated Development Environment*) je programska podrška koja programerima pruža širok skup objekata za razvoj programskih aplikacija. Uobičajeno se sastoji od: uređivača izvornog koda, kompajlera (interpretera), alata za građenje koda i pronalaženje grešaka.

A. Eclipse

Eclipse je programsko razvojno okruženje koje pruža proširivu razvojnu platformu. Razvijeno je na Java programskom jeziku, a koristi se za razvoj alata, aplikacijskih okruženja kao i samostalnih aplikacija (engl. *Rich Client Platform – RCP).*

Eclipse SDK (engl. *Eclipse Software Development Kit*) je kombinacija nekoliko Eclipse projekata, uključujući Platform, Java Development Tools (JDT) i Plug-in Development Environment (PDE). Eclipse SDK je integrisano razvojno okruženje za razvoj Java aplikacija i izgradnju proizvoda baziranih na Eclipse platformi. Ova platforma, nezavisna od proizvođača, sastoji se od jezgra (engl. *core*) i raznih priključaka (engl. *plugin*). Jezgro pruža usluge za upravljanje priključcima, a priključci pružaju potpunu funkcionalnost.

PDE (engl. *Plugin Development Environment*) je okruženje za razvoj priključaka i predstavlja skup alata dizajniranih da pomažu Eclipse programeru u razvoju, ispitivanju, izgradnji izvršnih arhiva i isporuci (engl. *deployment*) Eclipse dodataka, fragmenata, mogućnosti (engl. *features*) i RCP aplikacija.

Eclipse platforma koristi model zajedničkog radnog prostora (engl. *Workbench*) u koje se integrišu svi alati koristeći dobro definisane tačke proširenja (engl. *extension points*). Na taj način platforma pruža korisniku jedinstveni izgled svih alata i pruža integrisano upravljanje svih resursa (projekata, direktorijuma, datoteka) kojima manipulišu ti alati.

B. Xtext okvir

U domenu razvoja jezika sa specifičnom sintaksom, namenjenom za konkretan problem, značajno mesto zauzima Xtext usko namenski jezik. Ovaj DSL jezik je nastao kao deo oAW3 (engl. *openArchitectureWare*) projekta. Domen Xtext DSL jezika je kreiranje DSL jezika tako da možemo reći da Xtext predstavlja meta-jezik [7].

Implementacija je zasnovana na kreiranju odgovarajućeg parsera i objektnog meta-modela na osnovu opisa zadatih na meta-jeziku pri čemu je Xtext meta-jezik.

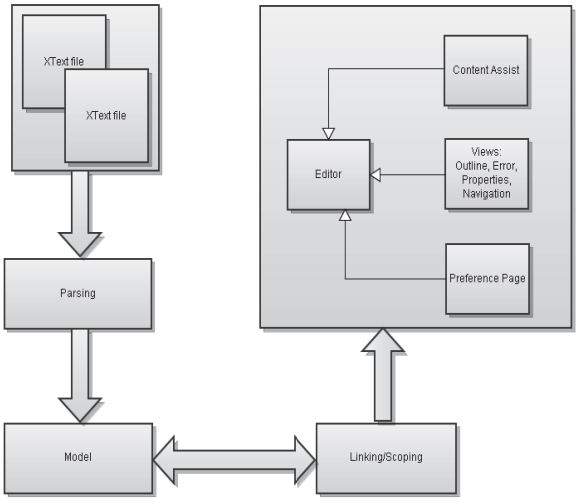
Rezultat parsiranja Xtext gramatike u ovom radu je asemblerski uređivač koji podržava asemblerske instrukcije. Ovaj pristup omogućava ugrađivanje složenih izraza. Kao rezultat rada dobija se celovito razvojno okruženje i potpuno integrisan Eclipse IDE. Bez obzira da li želimo da napravimo mali tekstualni usko namenski jezik ili pak jezik opšte namene [6].

Pošto su DSL jezici predodređeni za manju ciljnu oblast, u tom duhu je i u ovom radu upotrebljen Xtext okvir za razvoj uređivača čija je namena razvijanje specifičnog asemblerskog jezika.

C. Karakteristike Xtext-a

Xtext definiše sledeće:

1. Skup AST klasa predstavljenih kao EMF (engl. *Eclipse Modeling Framework*) meta-model,
2. Parser koji je u mogućnosti čitati tekstualnu sintaksu, nakon čega vraća generisani meta-model,
3. Eclipse uređivač koji nam omogućava predstavljanje pravila sintakse jezika u više boja, prediktivnog asistenta dopune programskog koda, konfigurisanu listu koju čine elementi sintakse, kao i proveru grešaka u sintaksi.



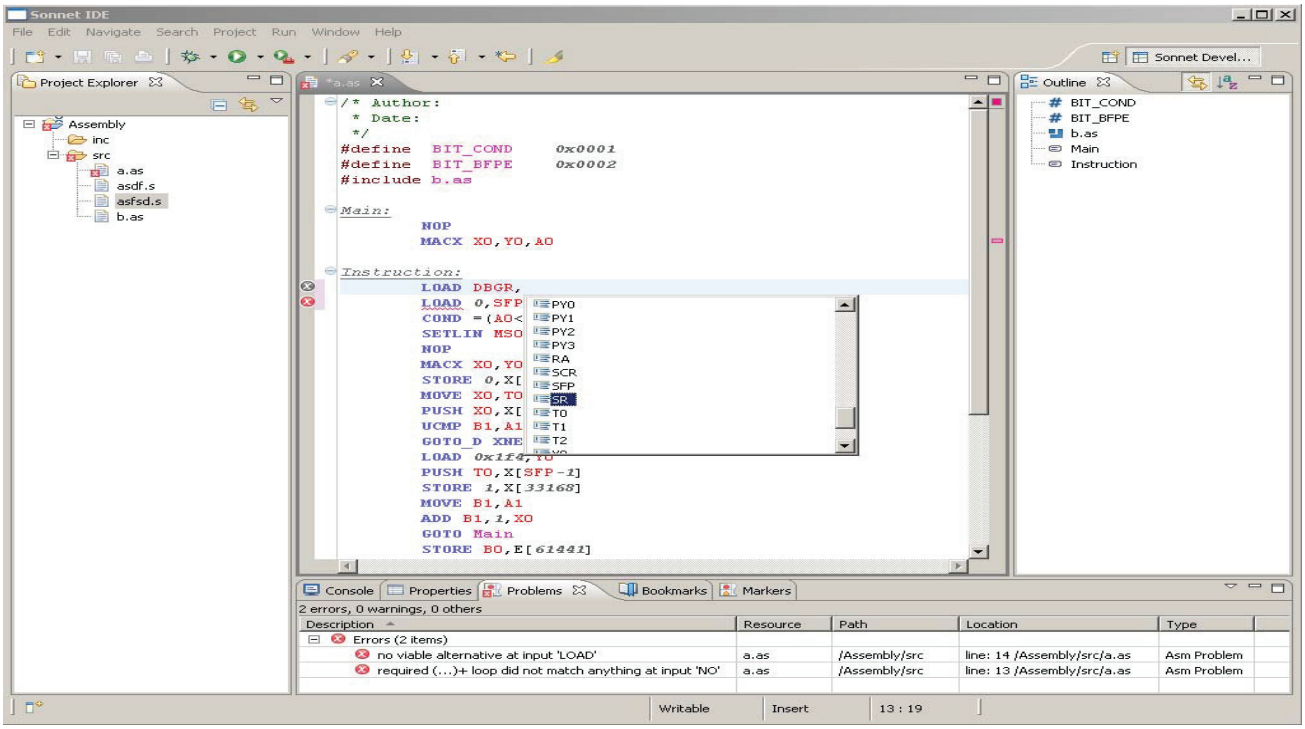
Sl. 1. Blok šema modela

Na slici 1. prikazana je blok šema modela koja predstavlja smer razvoja uređivača sa podrškom za razvoj asemblerskog jezika.

Nakon što je na pravi način sagledan problem pravi se model koji će dovesti do rešenja. Na početku se pišu pravila jezika odnosno u ovom slučaju specifična asemblerska gramatika. Od toga kako je definisana gramatika, u daljem procesu razvijanja nam zavisi funkcionalnost automatskog prediktivnog asistenta, sam sadržaj liste koju sačinjavaju elementi sintakse, kao i mogućnost verifikacije grešaka.

Nakon ispravno definisane gramatike vrši se parsiranje iste i kao rezultat dobija se model sa pravilima novog jezika. Međusobna veza unutrašnjih pravila zavisi od načina definicije istih. Pravila se pišu u formi stabla sa početkom na vrhu pa kroz propadanje i grananje dolazi se do najsitnijih detalja.

U sledećem koraku vrši se povezivanje kao i automatsko generisanje pojedinih delova modela. Sve ovo omogućava Xtext okvir, kao i samu izmenu svih generisanih klasa. Pravilnim definisanjem sintakse dobija se lista sa sadržajem asemblerskog uređivača, automatski prediktivni asistent, spisak grešaka koje je korisnik napravio, kao i stranica sa mogućim dodatnim podešavanjima koja su vezana za rezultujući uređivač.



Sl. 2. Asembler uređivač sa elementima

Na slici 2. prikazan je uređivač sa svim svojim dodatnim elementima poput liste sadržaja uređivača, liste napravljenih grešaka od strane onog ko razvija sintaksu, automatskog asistenta koji pomaže razvoj sintakse.

Ovom slikom su obuhvaćeni samo bitniji delovi uređivača.

III. XTEXT GRAMATIKA

U ovom primeru predstavljen je veoma mali deo procesa razvijanja uređivača sa podrškom za specifičnu asemblersku sintaksu. Priloženi primer je veoma jednostavan i za zadatak ima da pokaže na koji način se može definisati gramatika u Xtext okviru da bi se u asemblerskom uređivaču mogle koristiti labele (engl. *Label*) kao i celobrojne konstante (engl. *Integer_constant*).

```
grammar sonnet.ide.xtext.Asm with
org.eclipse.xtext.common.Terminals
generate asm "http://www.ide.sonnet/xtext/Asm"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

Model: (instr+=Instruction)*;
Instruction: Label | Dot | Preprocessor_commands;
Label: LabelWithPunkt | LabelWithEndl;
LabelWithPunkt: FirstOfLabel command+=Commands+ ;
FirstOfLabel: (name=ID | name1=INT) punkt=Punkt;
Punkt: punkt=(':' | '::');
. . .
Numbers: Integer_constant | Floating_point_constant;
Integer_constant: Binary | Octal | Decimal | Hexadecimal;
Decimal: INT | INTDECIMAL;
Hexadecimal: HEXA;
Binary: BINARY;
Octal: OCTAL;
. . .
terminal BINARY: '-'? '0b' ('0' | '1')+;
terminal OCTAL: '-'? '0' ('0'..'7')+;
terminal HEXA: '-'? '0x' ('a'..'f' | 'A'..'F' | '0'..'9')+;
terminal INT returns ecore::EInt: ('0'..'9')+;
terminal INTDECIMAL: '-'? ('0'..'9') ('0'..'9')*;
terminal ID: '^'? ('a'..'z' | 'A'..'Z' | '_') ('a'..'z' | 'A'..'Z' | '_' | '0'..'9' | '.')*;
```

Pravila - Osnovno parsiranje gramatike može biti odvojeno u sledeće faze:

1. Leksička analiza (engl. *Lexing*),
2. Parsiranje (engl. *Parsing*),
3. Povezivanje (engl. *Linking*),
4. Potvrđivanje (engl. *Validation*).

Xtext definiše određena semantička pravila na nivou gramatike. Ova pravila omogućavaju izvođenje semantičke analize nad stablom parsiranja samo na osnovu tekstualnog opisa jezika [5].

Neka od semantičkih pravila koja se navode na nivou Xtext gramatike su:

1. Dodele vrednosti – iskaz *(instr += Instruction)*;* definiše gramatičko pravilo prepoznavanja većeg broja iskaza koji su tipa Instruction. Ovaj iskaz istovremeno opisuje semantiku koja kaže da će se kreirati atribut instr koji će predstavljati listu objekata tipa Instruction a sve to zbog kvantifikatora zvezde (*)
2. Akcije koje se definišu unutar zagrade *'{'*. Na kraju iskaza se nalazi takođe zagrada *'}'* koja definiše kraj iskaza. Akcije u Xtext-u omogućavaju redefinisanje povratnog tipa
3. Reference predstavljene sa *instr = [Instruction];* predstavljaju gramatičko pravilo koje će u trenutku kada naiđe na ukazivač na Instruction prepoznati je. Rezultujući objekat će za vrednost ukazivača instr imati ukazivač na Instruction.

Xtext implementacija koja je deo EMF-a pored kreiranja parsera i odgovarajućeg meta - modela kreira i odgovarajući uređivač koji se u potpunosti integriše u Eclipse okruženje.

Prilikom parsiranja, u prvoj fazi niz karaktera odnosno tekstualni ulaz se deli na tokene (engl. *tokens*). Token je kratak deo ulaznog niza karaktera i sadrži jedan ili više karaktera. Token je pokazivač na neko drugo pravilo ili na neko od ugrađenih terminal pravila (ID, INT itd.) Terminal pravila su takođe tokeni. Postoji konvencija o imenovanju

terminal pravila velikim slovom. Xtext automatski pravi meta model iz definisane gramatike.

Pravila parsiranja u primeru su: *Model, Instruction, Label, LabelWithPunkt, FirstOfLabel, Punkt, Numbers, Integer_constant, Binary, Hexadecimal, Octal, Decimal,* dok su terminal pravila: *BINARY, OCTAL, HEXA, INT, INTDECIMAL, ID*. Telo pravila se nalazi iza dvotačke i sastoji se od definicije terminala i neterminala. Terminali se navode kao konstante karaktera (npr. ključna reč *'0b'* kod pravila *BINARY*) dok se neterminali kreiraju uvezivanjem na druga pravila parsiranja. Operatorima dodele je omogućeno definisanje vrednosti atributa instanci klasa odgovarajućeg meta modela. Takođe je moguća upotreba standardnih kvantifikatora:

1. ? - nula ili jedan broj pojavljivanja u sintaksi
2. * - nula ili više puta je moguće pojavljivanje u sintaksi
3. + - jedan ili više ponavljanja u sintaksi
4. kao i predikata: ! - ne i & - i

Kao što vidimo gramatika počinje sa određenim zaglavljem koje definiše neke karakteristike gramatike. Prvi deo zaglavlja predstavlja ime gramatike dok drugi deo definiše drugu postojeću gramatiku koja može da se iskoristi.

Prvo pravilo u primeru (Model) se naziva ulazno pravilo. Ono predstavlja skup svih instrukcija obuhvaćenih gramatikom kojom se realizuje podrška asemblerskog jezika. Pravilo koje omogućava podršku za pisanje imena (engl. *Label*) počinje ID ili INT pravilom (a-z, A-Z, 0-9, _) koje obuhvata navedene karaktere, cifre, kao i donju crtu. Nakon ovog pravila stavlja se dvotačka iza koje sledi bilo koja instrukcija iz seta instrukcija Commands.

Druga instrukcija je nekoliko celobrojnih konstanti:

1. Binarna (engl. *Binary*)
2. Decimalna (engl. *Decimal*)
3. Oktalna (engl. *Octal*)
4. Heksadecimalna (engl. *Hexadecimal*)

Svaka od ovih konstanti osim decimalnog formata zapisa počinje sa prefiksom na koji se nastavlja niz cifara. Pa tako za binarni format prefix je 0b, za oktalni 0 a za heksadecimalni 0x. Na prefikse se nastavlja niz cifara dok za heksadecimalni format pored niza cifara imamo i karaktere A-F, a-f.

Opisali smo DSL jezik sa nekoliko linija koda i kao rezultat dobili uređivač sa mogućnošću razvijanja asemblerskog jezika. U ovom radu objašnjena je podrška za samo nekoliko pravila koji su dati kao primer. Xtext alati generišu tekstualni uređivač, koji se koristi kao priključak za Eclipse.

IV. Zaključak

Usko namenski jezici igraju sve važniju ulogu u razvoju softverskih proizvoda. Razvoj jezika i pratećih alata je u prošlosti bio složen posao koji je zahtevao značajne resurse. Danas, razvojem okvira za razvoj DSL jezika, to više nije slučaj.

Jedan od najpopularnijih alata za razvoj tekstualnih DSL jezika je Xtext okvir za razvoj, koji omogućava generisanje okruženja za razvoj u vidu Eclipse priključaka na osnovu opisa na jeziku sličnom BNF-u. Xtext se razvija u sklopu EMP podprojekta Eclipse platforme.

Ovim radom potpuno je razvijen uređivač čija je namena razvoj asemblerskog jezika za specifičnu DSP platformu. Na osnovu specifikacije DSP platforme razvijeni jezik je predodređen i optimizovan za istu.

Da bi se stekla bolja slika o obimu uloženog truda u razvoj uređivača za ciljnu platformu, navodi se da je napisano oko hiljadu linija koda gramatike, iz kojih je Xtext okvir izgenerisao mnoštvo klasa i metoda (par desetina hiljada linija koda), zatim su proširene postojeće klase namenjene proveri grešaka, formatiranju sadržaja uređivača, bojenju sintakse, listi sadržaja uređivača itd. Ova proširenja za cilj su imala poboljšanje prvobitne funkcionalnosti (dobijene Xtext okvirom).

Dalja poboljšanja idu u smeru još bolje optimizacije jezika, kao i grafičkih komponenti uređivača, prvenstveno dela koji ukazuje na sadržaj samog uređivača, kao i automatskog prediktivnog asistenta kako bi mu se poboljšala osobina predikcije, da bi u svakom trenutku mogao čitati misli programera koji razvija jezik.

Literatura

[1] Starkey, Official web page Available: http://www.starkey.com, accessed 27.9.2012.

[2] Marjan M., Jan H., Antony M., *When and how to develop domain-specific languages,* ACM Computing Surveys (CSUR), Volume 37 Issue 4, New York, USA, December 2005.

[3] Volter. M., *DSLs for Product Lines: Approaches, Tools, Experiences*, Software Product Line Conference (SPLC), 2011 15th International Conference of the IEEE, Stuttgart, Germany, August 2011.

[4] Irazábal, J., *Supporting Modularization in Textual DSL Development*, Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the IEEE, Buenos Aires, Argentina, November 2010.

[5] Savic, D., Antovic, I., Vlajic, S., Stanojevic, V., Milic, M., *Language for Use Case Specification*, Software Engineering Workshop (SEW), 2011 34th IEEE, Limerick, June 2011.

[6] *Xtext – Language Development Made Easy*, www.eclipse.org/Xtext/documentation.html, accessed 25.9.2012.

[7] *oAW Xtext: A framework for textual DSLs,* www.voelter.de/data/workshops/EfftingeVoelterEclipseSummit, accessed 27.9.2012.

[8] Dejanović, I.; Milosavljević, G.; Perišić, B. & Tumbas, M., *A Domain-Specific Language for Defining Static Structure of Database Applications*. Computer Science and Information Systems, 2010, 7, 409-440

Abstract

In this paper, using the Xtext framework describes the implementation of an assembler editor for the development of assembly code. This editor supports specific assembler instructions. In addition to the editors realized the set of additional tools which substantially facilitate the development and to use assembler editor. User, are available to other settings of the editor, the editor content monitoring, recognizing the same instructions and an automatic assistant to the prediction based on the user provides certain features of the language development.

Xtext provides modern API for describing different aspects of DSL programming languages. Language developed in Xtext framework is independent of Eclipse and can be used in any Java environment.

ONE SOLUTION OF IMPLEMENTATION ASSEMBLER EDITOR ON THE JAVA PLATFORM USING THE XTEXT FRAMEWORK

Saša Kartalija, Ivan Letvenčuk, Jovica Grahovac, Jelena Kovačević