Nadgradnja obstoječega domensko specifičnega modelirnega jezika za merilno tehniko

Tomaž Kos[1], Marjan Mernik[2], Tomaž Kosar[2]

[1] *Dewesoft d.o.o., Gabersko 11a, 1420 Trbovlje*
E-pošta: tomaz.kos@dewesoft.com
[2] *Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, Koroška cesta 46, 2000 Maribor*
E-pošta: marjan.mernik@um.si, tomaz.kosar@um.si

Extending an existing domain-specific modeling language for measurement domain

Abstract. *As applications change over time, so do programming languages. Since the Sekvencer language, which is used in measurement system Dewesoft, has been in use since 2010, in practice it has become clear that users want new functionalities and mechanisms. In this article, we present a prototype of the new programming language RT-Sekvencer. The language is adapted to create real-time measurement procedures that are performed on hard real time systems.*

1 Uvod

Kljub napredku programskih jezikov in naprednih platform je večina programske opreme razvita na nizki ravni abstrakcije glede na koncepte, ki jih uporabljajo programerji aplikacij. Ta semantični prepad rešujejo domensko specifični jeziki (angl. domain-specific languages, DSL) [1]. Ti zajemajo bistvene značilnosti problemskega prostora na način, ki je ločen od podrobnosti specifičnega prostora za reševanje [2]. Ti jeziki, če jih primerjamo s splošno namenskimi jeziki (angl. general-purpose languages, GPL), podpirajo višjo raven abstrakcije in so bližje domeni problema kot domeni implementacije [3]. V ospredje postavljajo domeno abstrakcije, ki omogoča domenskim strokovnjakom programiranje direktno z domenskimi koncepti. Koristi uporabe DSL-jev so povečana fleksibilnost, produktivnost, zanesljivost in uporabnost in so bile pokazane tudi v empiričnih študijah [4].

Prav tako poznamo tudi domensko specifične modelirne jezike (angl. doman-specific modeling languages, DSML) [5], ki se v marsičem prekrivajo z DSL-ji. Ti jeziki specificirajo programe z modeli in uporabljajo vizualne koncepte, ki še dodatno dvignejo nivo abstrakcije. Abstraktna sintaksa jezika je običajno definirana z metamodelom, ki opisuje koncepte jezika, povezave med njimi in strukturna pravila, ki omejujejo elemente modela in njihove kombinacije.

V osnovi so domensko specifični modelirni jeziki zelo primerni za uporabo na specifičnih področjih, kot je na primer načrtovanje kompleksnih merilnih sistemov. Merilni sistemi Dewesoft, skupaj s programskim produktom Dewesoft[1], se uporabljajo za zajem, obdelavo in kasneje tudi analizo izmerjenih meritev in pri tem uporabljajo domensko specifičen modelirni jezik Sekvencer [5, 6] za konstrukcijo merilnih postopkov. Z omenjenim jezikom lahko končni uporabnik, domenski strokovnjak, na enostaven način brez strokovnega računalniškega znanja kreira poljuben merilni postopek, ki se vrši na merilnem sistemu.

Ker je jezik Sekvencer v uporabi že od leta 2010, je praksa pokazala, da si posamezni uporabniki želijo novih funkcionalnosti in mehanizmov. V tem članku tako predstavljamo prototip novega programskega jezika, ki smo ga poimenovali RT-Sekvencer. Prav tako bo jezik prirejen za kreiranje realnočasovnih postopkov, ki se izvajajo na krmilnikih Dewesoft, kot je IOLite-LX [7].

Članek je strukturiran v pet poglavij. V drugem poglavju predstavimo merilni in krmilni del sistema Dewesoft. Prav tako v tem poglavju opišemo paket Dewesoft skupaj s trenutnim programskim jezikom Sekvencer. Sledi tretje poglavje, kjer so predstavljene motivacija in zahteve. V četrtem poglavju sledi predstavitev prototipa: arhitektura, način prevajanja in generiranje kode. Zadnje poglavje povzame vsebino članka.

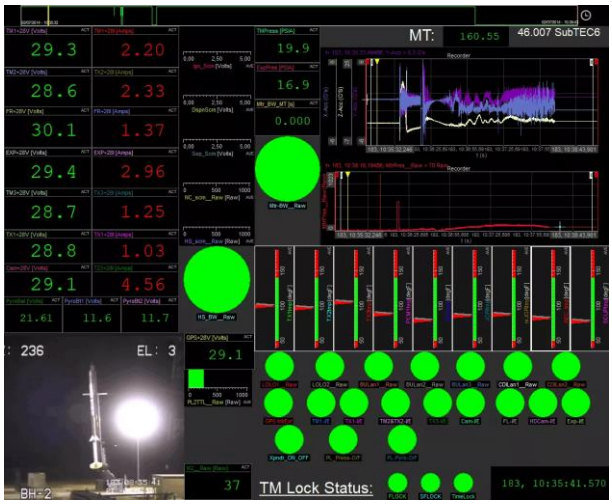
2 Obstoječe razvojno okolje

V zadnjem desetletju je razvoj osebnih računalnikov s seboj prinesel možnost za razvoj namenskih namiznih aplikacij. Ena od teh je programski paket Dewesoft in DSML programski jezik Sekvencer, ki je v komercialni uporabi že več kot deset let in je na kratko predstavljen v naslednjih podpoglavjih.

2.1 Merilni programski paket Dewesoft

Programski paket Dewesoft (trenutna verzija X 2021.4) je produkt istoimenskega podjetja Dewesoft d.o.o. Programski paket v osnovni ideji in izvedbi predstavlja nadomestek klasičnim merilnim instrumentom (npr. logični analizator, funkcijski generator) in ob enostavni uporabi ponuja vse prednosti, ki jih omogoča merilni računalnik pri zajemanju in na koncu tudi analizi podatkov. Programski paket je prisoten in se uporablja v vseh panogah industrije, od avtomobilske do vesoljske industrije, v času testiranja novih produktov in komponent. S programskim paketom lahko zajamemo podatke iz različnih senzorjev, kot so analogni in digitalni senzorji, video, GPS, CAN, XCP, PCM itd. Poleg že prej omenjenih lastnosti orodje omogoča tudi vizualizacijo, analizo in tudi izvoz v različne formate za procesiranje, kot je prikazano na sliki 1.

[1] https://www.dewesoft.com



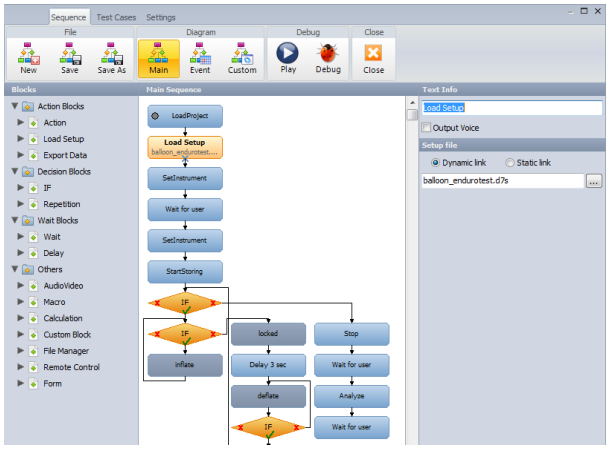
Slika 1: Primer merilnega sistema Dewesoft

2.2 Programski jezik Sekvencer

Jezik Sekvencer je vključen v programski paket Dewesoft X. V industriji ga uporabljajo za definiranje merilnih postopkov za namen testiranja delovanja produktov.

Vizualne gradnike premikamo po metodi »primi in spusti« in jim določamo pripadajoče lastnosti. Programski gradniki jezika predstavljajo akcije, ki se izvajajo. Akcije se izvajajo od začetnega gradnika (označen s krogcem) in se nadaljujejo na naslednjem gradniku, kamor kaže puščica. Gradniki lahko vsebujejo tudi lokalne in globalne spremenljivke, ki so namenjene za hranjenje trenutne vrednosti. Ker lahko ob velikem številu gradnikov vizualno programiranje postane neobvladljivo, modelirni jezik omogoča tudi vgnezdene modele (angl. custom blocks), s pomočjo katerih lahko združimo gradnike v celoto in se nanje sklicujemo v preostalem delu programa.

Na sliki 2 je prikazano razvojno orodje. Na levi strani slike vidimo gradnike domenskega jezika. Na sredini je delovna površina, kjer končni uporabniki programirajo proces. Na desni strani zaslona pa najdemo lastnosti, ki jih lahko uporabnik izbira za izbrani gradnik iz generiranega modela.



Slika 2: Razvojno okolje za Sekvencer

3 Motivacija in zahteve

Motivacija za nov jezik RT-Sekvencer je bila izdelava programov, ki tečejo na realnočasovnih sistemih. Glede na zahteve odziva lahko realnočasovne sisteme delimo na mehke (angl. soft real time systems) in trde (angl. hard real time systems). Prvi so tisti, katerih odzivnost v nekem določenem času ni tako pomembna in ne bo kritično vplivala na delovanje sistema. Med te sisteme štejemo obstoječe orodje Sekvencer. V drugo skupino štejemo trde krmilnike v realnem času (angl. hard real time controllers, hard RTC). Pri teh je odzivnost ključnega pomena in lahko neodzivnost v določenem času povzroči kritično napako (npr. lahko povzroči poškodbe, izgubo življenj, znatno finančno izgubo itd.). Bistvene značilnosti takih sistemov so zajamčeni in predvidljivi odzivni časi, izjemna zanesljivost in varnost, nizek reakcijski čas, frekvenca zanke stabilnosti. Ker se v nekaterih primerih Dewesoft uporablja za kritične aplikacije, je ključnega pomena, da je RT-Sekvencer zasnovan kot trdi realnočasovni sistem.

Bistveni motiv pri izgradnji orodja je bil tudi povezava dveh vrst sistemov uporabe. Na eni strani so to sistemi za zajem podatkov (angl. Data Acquisition, DAQ). Ti sistemi omogočajo hitrejši zajem podatkov, časovno usklajenost in običajno boljšo natančnost vhodnih podatkov. Na drugi strani pa so to realnočasovni sistemi, ki temeljijo na čim manjši latenci povratne zanke. V našem primeru bi želeli imeti sistem z večjo vzorčevalno frekvenco (npr. 1 MHz) in manjšim reakcijskim časom (npr. 1 kHz), kar v praksi pomeni, da bi v kontrolnem ciklu zajeli več podatkov hkrati (npr. 1000 podatkov na sekundo).

Ob upoštevanju zgornjih dejstev smo pri izgradnji DSML-ja za kreiranje realnočasovnih postopkov podali naslednje zahteve:

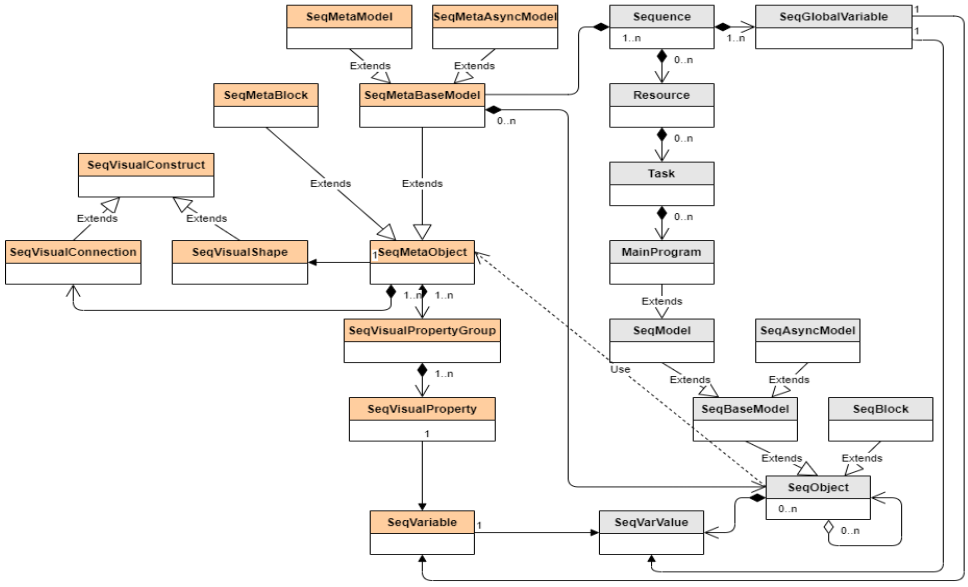
- kreiranje programov mora biti čim bolj preprosto,
- kompleksnost programov mora biti čim manjša,
- povečati se mora kvaliteta programov,
- enostavnejša detekcija napak programov,
- produktivnost programiranja se mora izboljšati v primerjavi s PLC-programiranjem,
- za kreiranje merilnega postopka ni potrebno imeti programerskega znanja iz GPL-jev,
- programiranje se prestavi pod okrilje domenskih strokovnjakov.

4 Prototip jezika RT-Sekvencer

Jezik RT-Sekvencer je bil primarno razvit z namenom programiranja realnočasovnih sistemov. S tem namenom je bilo zasnovano celotno orodje in vsa pripadajoča infrastruktura, ki jo bomo spoznali v nadaljevanju.

4.1 Modelirni jezik

Modelirni jezik je bil načrtovan na osnovi predhodnega jezika Sekvencer, kjer se je naredila analiza domene.



Slika 3: Metamodel jezika RT-Sekvencer

Koncepti in terminologija jezika so bili opredeljeni na podlagi izkušenj in potreb uporabnikov z različnih področij uporabe. Ker se bo jezik uporabljal tudi za modeliranje realnočasovnih programov, smo morali dodati tudi nove konstrukte. Jezik smo zasnovali tako, da lahko konstrukte dodajamo poljubno na enostaven način. Uporabnik jezika ima možnost programirati s predpripravljenimi konstrukti. Poleg že omenjenih lastnosti je bil jezik nadgrajen tudi z lastnostjo dodajanja lastnih konstruktov, ki se načrtujejo in logično programirajo znotraj jezika C++ (za napredne uporabnike). S tem konceptom smo lahko dosegli odprtost, generalizacijo in lahko nadgradljivost glede na potrebe domene.

Pri realnočasovnih procesih se je pokazala potreba po večprocesnem in večnitnem programiranju. V jezik je dodan mehanizem za kreiranje in kasneje izvajanje več procesov, ki pa so lahko različnega tipa:

- *glavni proces*, ki se začne ob zagonu programa,
- *periodični proces*, ki se začne izvajati ob določeni časovni periodi,
- *sproženi proces*, ki se začne izvajati ob določenem zunanjem dogodku.

Poleg novih procesov so bili dodani tudi konstrukti za večnitno programiranje znotraj procesa.

V osnovi so DSML-jeziki razviti na osnovi metamodela, ki predstavlja abstraktno notacijo jezika. Pri definiciji jezika smo definirali metamodel z razrednim diagramom UML, kot je prikazano na sliki 3. Definirali smo razrede, atribute in povezave med njimi. Za boljšo predstavo metamodela na sliki 3 smo odstranili atribute razredov in zapisali samo osnovne razrede (npr. iz razreda SeqVariable so izpeljani različni tipi spremenljivk, kot je SeqStringVariable). Diagram je razdeljen na dva dela. Na desni strani diagrama jezika (razredi, obarvani s sivo barvo) je metamodel programa. Ta del je na eni strani definiran hierarhično glede na zasnovo programa in arhitekture krmilnika. Po drugi strani pa je v tem zapisu definirana logika programa. Struktura je definirana in zasnovana tako, da je osnovni razred Sequence. Ta razred ima lahko različne vire (razred Resource), ki predstavljajo dejanski krmilnik oz. procesno enoto izvajanja programa. Znotraj vira so definirana različna opravila (razred Task). Opravilo je enota, ki vsebuje informacije o tem, kako in kdaj se neki program izvede. Nato hierarhično sledi logika opravila (razred MainProgram), ki je izpeljana iz razreda SeqModela in se nanaša na pravila, ki so zapisana v njegovem metamodelu (razred SeqBaseModel).

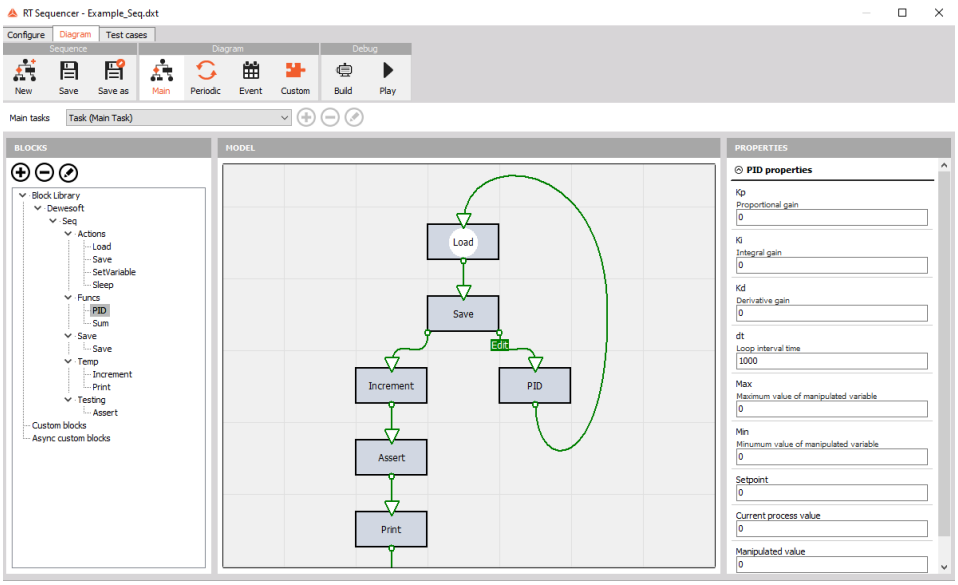
Na levi strani slike 3 razredi, obarvani z oranžno barvo, predstavljajo konstrukte in pripadajoče razrede, ki omogočajo poljubno definicijo modelirnih blokov. Predstavljajo sintakso našega programa, napisanega na desni strani diagrama. Osnovni razred je SeqMetaObject, iz katerega sta izpeljana razreda SeqMetaBlock in SeqMetaBaseModel. Prvi predstavlja konstrukt, s pomočjo katerega se lahko modelira program. Drugi pa predstavlja skupek konstruktov, ki se lahko znotraj modela izvajajo (razred SeqObject). Ti konstrukti se lahko izvajajo sinhrono (razred SeqMetaModel) in asinhrono s pomočjo nitenja (razred SeqMetaAsyncModel).

Vsak razred SeqMetaObject vsebuje različne skupine vizualnih blokov (razred SeqVisualPropertyGroup in SeqVisualProperty) in so lahko različnih podatkovnih tipov (razred SeqVariable). Ti podatkovni tipi so lahko vhodne, izhodne in konstantne spremenljivke.

Vizualna notacija jezika in konstruktov je določena s pomočjo likov (razred SeqVisualShape) in povezav (razred SeqVisualConnection). Vsaka oblika je določena in sestavljena iz različne barve, velikosti in oblike, medtem ko je povezava fiksno določena in ima obliko črte s puščico. Vsaka oblika pripada točno enemu konstruktu (razred SeqMetaObject). Vsak konstrukt predstavlja akcijo, ki se bo sekvenčno izvedla. Kot je razvidno iz metamodela, ima lahko vsak konstrukt N vhodnih in M izhodnih povezav.

4.2 Modelirno okolje

Modelirno okolje vključuje štiri osnovne funkcionalnosti orodja, ki so nujne za kasnejše generiranje končnega programa:



Slika 4: Modelirno orodje

- konfigurator sistema,
- izdelava uporabniških blokov,
- modelirno orodje in
- procesiranje/vizualizacija.

Prvi del okolja predstavlja konfigurator sistema, kjer vizualno nastavimo posamezne komponente sistema. Znotraj tega orodja nastavimo senzoriko (npr. tip meritve, skalirne faktorje), določijo se opravila, ki jih naknadno modeliramo, kreiramo uporabljene spremenljivke in splošne nastavitve samih krmilnikov.

Pri kreiranju novih blokov si pomagamo z orodjem za izdelavo uporabniških blokov. Znotraj orodja lahko uporabnik spreminja izvajalno kodo ali njene lastnosti. Orodje je razdeljeno na 4 skupine, ki se med seboj povezujejo. V prvi skupini lahko spreminjamo osnovne lastnosti, kot so ime, avtor, datum, verzija, in določimo, koliko vhodnih in izhodnih povezav ima posamezen blok. V drugi skupini vizualno načrtujemo blok in določimo njegovo obliko. Tretja skupina so lastnosti povezav, kjer lahko določimo, katere lastnosti posamezen blok vsebuje. Kreirane lastnosti lahko kasneje spremenimo med programiranjem. Ti parametri bodo vplivali na delovanje samega bloka in tudi programa v celoti. Četrta in zadnja skupina je sama logika izvajanja bloka ali njegova izvajalna koda. To orodje je namenjeno uporabnikom, ki jim programiranje v programskem jeziku C++ ne dela preglavic.

Modelirno orodje je namenjeno kreiranju samega programa. Kot je prikazano na sliki 4, uporabnik s predpripravljenimi bloki (leva stran) kreira postopek delovanja programa. Na sredini je delovna površina, kamor polagamo bloke in gradimo program. V primerjavi s prejšnjo različico so bile dodane tudi osnovne modelirne akcije (povečaj/pomanjšaj, kopiraj, prilepi itd.). Pri vsakem bloku lahko na desni strani orodja spremenimo njegove lastnosti. Ko je program zgrajen, se lahko znotraj orodja prevede v izvajalno kodo, ki se kasneje prenese na realnočasovni sistem.

V zgrajenem programu orodje omogoča tudi procesiranje in vizualizacijo statusnih kanalov, ki se pojavljajo znotraj realnočasovnega sistema. Ti statusi so lahko spremenljivke, ki se definirajo znotraj orodja, ali predpripravljeni statusi, ki sporočajo pravilno delovanje sistema (uporaba procesorja, pomnilnika, časovna izmera trajanja izvajalnega cikla, trenutno izvajalni blok itd.).

5 Zaključek

Zaradi želje po novih funkcionalnostih je programska oprema podvržena nenehnim spremembam. Enako velja tudi za domensko specifične jezike. V članku je opisan motiv za razvoj prototipa novega programskega jezika, imenovanega RT-Sekvencer. Obstoječi programski jezik Sekvencer smo nadgradili s podporo za realnočasovne sisteme in ga tako približali domenskim strokovnjakom. Poleg motivacije za razvoj so izpostavljene še njegove prednosti, kot so programiranje krmilne zanke, uporaba naprednih funkcij produkta Dewesoft in povezovanje dveh področij: DAQ in RTC-jev. Prav tako so predstavljeni arhitektura, načrtovanje in razvoj modelirnega jezika in okolja.

Literatura

[1] M. Mernik, J. Heering, A. Sloane: When and How to Develop Domain-Specific Languages. ACM Computing Surveys 2005, 37, 316–344, 2005.

[2] A. de la Vega, D. García-Saiz, M. Zorrilla, P. Sánchez: Lavoisier: A DSL for increasing the level of abstraction of data selection and formatting in data mining. Journal of Computer Languages, 60, 100987, 2020.

[3] T. Kosar, P.E. Martínez López, P.A. Barrientos, M. Mernik: A preliminary study on various implementation approaches of domain-specific language. Information and Software Technology, 50, 390 – 405, 2008.

[4] T. Kosar, M. Mernik, J. Carver: Program comprehension of domain-specific and general-purpose languages: Comparison using a family of experiments. Empirical Software Engineering, 17, 276–304, 2012.

[5] T. Kos, T. Kosar, M. Mernik: Development of data acquisition systems by using a domain-specific modeling language. Computers in industry, 63(3), 181-192, 2012.

[6] T. Kos, T. Kosar, M. Mernik: A tool support for model-driven development: An industrial case study from a measurement domain. Applied Sciences, 9(21), 4553, 2019.

[7] Dewesoft IOLITE LX, https://dewesoft.com/products/daq-and-control-systems/iolite-lx