MATLAB KALBOS OBJEKTINIO PROGRAMAVIMO GALIMYBIŲ LYGINAMOJI ANALIZĖ

Aurimas Šimkus, Sigita Turskienė
Šiaulių universitetas

Įvadas

Programavimo standartų kompanijos TIOBE 2012 m. spalio mėn. atliktų tyrimų duomenimis, objektinis programavimas (OP) jau penkerius metus yra gana populiari programavimo paradigma [11].

Daug šilumos pernešimo ir kitų fizikinių reiškinių modeliavimo programų baigtinių elementų metodu (BEM) sukurta procedūrine programavimo paradigma [1, 8, 9, 10, 12]. Iš [4, 13] seka, kad šiuo metu vis plačiau nagrinėjamos OP realizavimo galimybės minėtiems uždaviniams spręsti. Nagrinėjamos ir kompiuterinių matematikos sistemų (KMS) *Matlab, Maple, Mathematica* [2, 3, 7] programavimo kalbų OP paradigmos realizavimo galimybės. Informacijos apie OP galimybes KMS programavimo kalbomis yra nedaug, nemaža jos dalis pateikiama KMS gamintojų tinklalapiuose. Realių didelių programų ar tyrimų sistemų kol kas nedaug sukurta. Todėl šiuo metu sunkoka spręsti apie OP paradigmos palaikymo kokybę minėtose KMS.

Tyrimo tikslas – atlikti OP galimybių *Matlab* programavimo kalba lyginamąją analizę, sukuriant šilumos pernešimo strype uždavinio sprendimo BEM programą.

Tyrimo uždaviniai: ištirti OP paradigmos realizavimo galimybes *Matlab* kalba, jas palyginti su *C++, C#* ir *Java* programavimo kalbų galimybėmis; sukurti objektinę programą šilumos pernešimo strype uždaviniui spręsti BEM; atlikti programos testavimą.

Tyrimo metodai: srities modeliavimas, funkcinė analizė, testavimas.

1. Matlab programavimo kalbos objektinio programavimo galimybių analizė

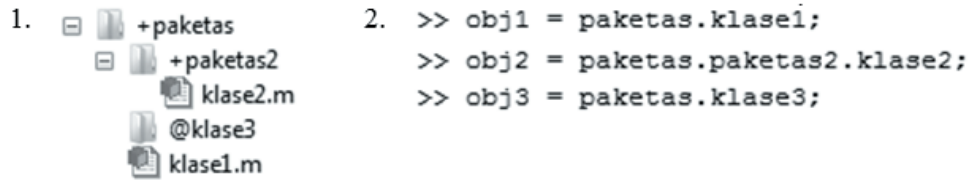
Objektiškai programuoti *Matlab* kalba galima trimis būdais:

- naudoti integruotąsias *Matlab* klases,
- kurti vartotojo klases ir objektus,
- naudoti *Java* arba *.NET Framework* klases [6].

Matlab kalba, kaip ir *C#* ar *Java* programavimo kalbos, yra realizuota objektine struktūra [7]. Skirtingai nei *C++* kalboje, *Matlab* kalboje visos duomenų struktūros realizuotos objektiškai.

1.1. Klasės aprašo sintaksės analizė

Iš [5, 7] seka, kad *Matlab* kalbos klasės sintaksė skiriasi nuo lyginamųjų programavimo kalbų sintaksės. Skirtingai nei palyginamosiose programavimo kalbose, *Matlab* kiekviena klasė kuriama atskirame faile. Jeigu klasė yra didelė ir sudėtinga, tai kuriamas klasės katalogas. Kataloge privalo būti to paties pavadinimo pagrindinis klasės failas, kuriame nurodomi klasės metodų ir kitų dalių aprašai. Metodai talpinami failuose greta klasės pagrindinio failo. Projekte, kuriame yra daug klasių, kai kurios klasės gali būti tarpusavyje susijusios, jos *Matlab* kalboje dedamos į paketus (1 pav.).



1 pav. Paketų ir išskaidytųjų klasių naudojimo pavyzdys: 1 – paketų struktūra, 2 – objektų kūrimas

Klasės antraštė susideda iš šių dalių: atributų aprašo, klasės pavadinimo ir bazinių klasių, skiriamų „&“ simboliu. Kaip ir *C++* kalboje, *Matlab* kalbos klasė gali paveldėti kelias klases, *C#* ir *Java* kalbose – tik vieną. Kreipinys į bazinės klasės metodus užrašomas taip:

- *metodas@bazinė_klasė* (*Matlab*),
- *bazinė_klasė::metodas* (*C++),*
- *bazinė_klase.metodas* (*Java, C#).*

Dar vienas skirtumas nuo *C++* kalbos yra klasės paveldimumo inkapsuliacija. *C++* kalboje galima nurodyti paveldėjimo inkapsuliaciją, pvz., *„class A : **private** class B“*. *Matlab* klasių paveldėjimas, kaip ir *C# ar Java* kalbose, yra viešas. Galima tik uždrausti klasę paveldėti arba paslėpti jos turinį. *Matlab* kalbos klasėje galima nurodyti šias klasės sudedamąsias dalis [7]:

- duomenis ir metodus, kuriuose skiriasi nagrinėjamų kalbų atributų naudojimas,
- išvardijimus, kurie skirtingai nei C++ kalboje, *Matlab* kalboje apibrėžiami tik klasėje,
- įvykius, kurie numatyti tik kaip klasės dalis (panašiai kaip *C# ar Java*). *C++* kalboje įvykiai, kaip atskiras darinys, nenumatytas.

1.2. Duomenų ir metodų apibrėžties galimybės

Iš [5, 7] seka, kad skirtingai nei *C++* kalboje, *Matlab* kalbos kiekvienas klasės duomuo, kaip ir *C#* bei *Java* kalbose, gali turėti standartinę reikšmę. Duomenys ir kitos *Matlab* duomenų struktūros yra betipiai.

Skirtingai nei nagrinėjamose kalbose, praktiškai kiekviena *Matlab* klasė turi konstruktorių. Jame iniciali-

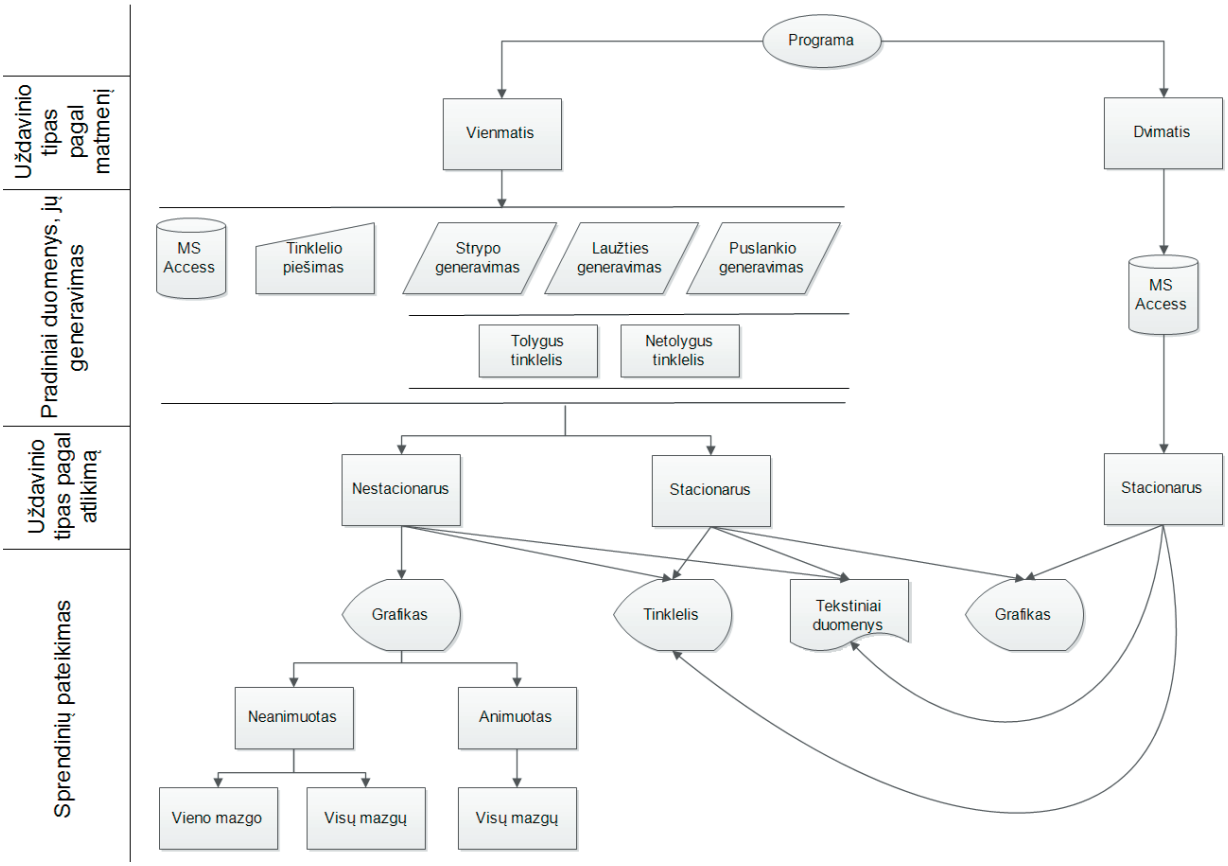
zuojamas vietinis klasės objektas (lyginamosiose kalbose – *this*), kuriuo susiejami klasės duomenys ir metodai. Metodai aprašomi kaip ir *Matlab* kalbos funkcijos. Objektų metodai nurodomi po objekto pavadinimo, pvz., *turtas.pridėti()*. Skirtingai nei *C++, C# ir Java* kalbose, metodo kreipinyje gali nebūti skliaustelių, jei metodas neturi parametrų.

Matlab kalboje, kaip ir nagrinėjamose kalbose, galima realizuoti metodų užklotį. Tačiau negalima vienoje klasėje sukurti dviejų vienodų vardų metodų. Galimas standartinių *Matlab* kalbos operatorių, pvz., sudėties, atimties užklojimas. Tada užklojančioji funkcija turi standartinį pavadinimą (*plus*, *minus*) bei numatytą parametrų skaičių.

Kaip ir *C++* kalboje, pasiekiamumą galima nurodyti duomenų ir metodų grupėms, tačiau *Matlab* kalba galima šias grupes dar skirstyti ir pagal kitus atributus. Pavyzdžiui, vienas duomenų blokas gali būti pasiekiamas iš išorės (*Access = public*), o duomenys – statiniai (atributas *Static*), o kitas – taip pat pasiekiamas, bet duomenys – dinaminiai. Palyginamosiose kalbose statiškumas nurodomas prie kiekvieno duomens ar metodo atskirai. *Matlab* kalboje duomenų pasiekiamumas gali būti nurodomas atskirai: tik gauti (*GetAccess*) arba tik nustatyti (*SetAccess*).

2. Šilumos pernešimo uždavinio sprendimas BEM

Kad išsamiau būtų išnagrinėtos Matlab kalbos OP galimybės, buvo sprendžiamas šilumos pernešimo strype uždavinys BEM. Uždavinio sprendimo BEM schema pateikta [10]. Objektinės programos kodas *MATLAB* kalba suskaidytas į atskirus paketus pagal klasių ir programos dalių paskirtį: *DB* – klasių paketas darbui su duomenų baze, *Gnr* – baigtinių elementų pradinių duomenų generavimas, *CalcM* – temperatūros pasiskirstymo strype skaičiavimas, *Grafika* – baigtinių elementų tinklelio ir skaičiavimo rezultatų vizualizavimas. Programos funkcinių dalių diagrama pateikta 2 pav.

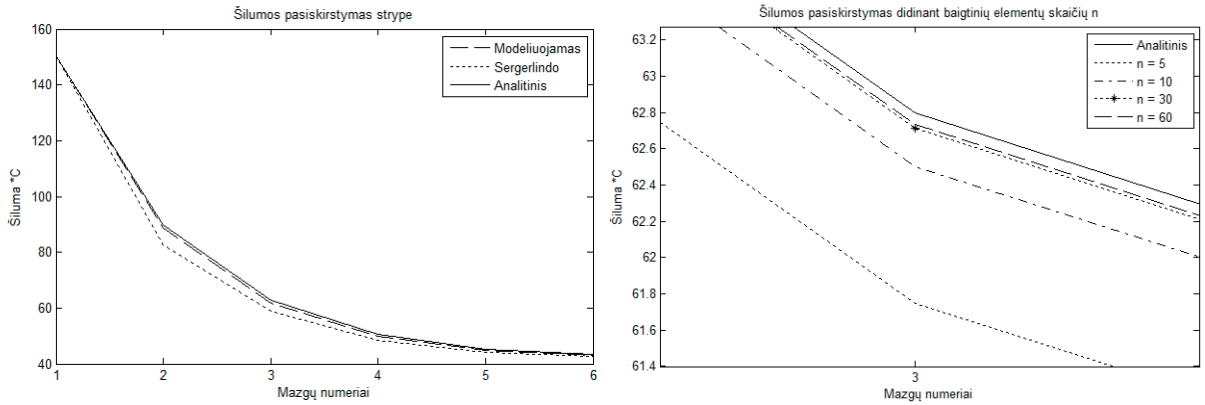


2 pav. Programos funkcionalumo diagrama

3. Skaičiavimų rezultatai

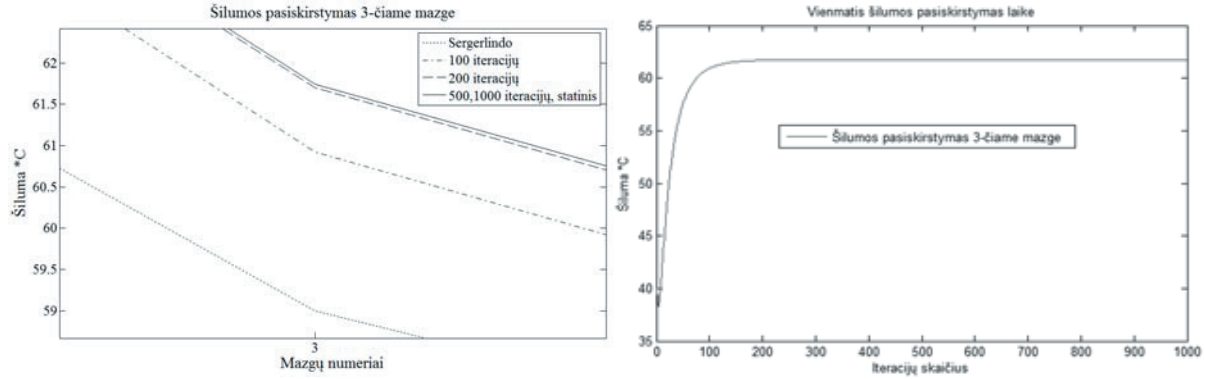
Sukurtos programos rezultatų teisingumas nustatytas nagrinėjant stacionarų ir nestacionarų temperatūros laukus strype [10]. Programos duomenys saugomi duomenų bazėje, nes yra lengviau atpažinti struktūrines duomenų dalis, be to, didėjant programos sudėtingumui, daugėja failų.

Gautos mazginės temperatūros reikšmės lyginamos su analitiniais rezultatais [10]. Be to, tyrinėtas temperatūros reikšmių kitimas, didinant baigtinių elementų skaičių (3 pav.).



3 pav. Stacionaraus temperatūros uždavinio sprendinio konvergavimas

Iš 3 pav. seka, kad gauti vienmačio uždavinio sprendiniai konverguoja, yra tikslesni už [10] pateikiamus sprendinius, o didinant baigtinių elementų skaičių, sprendinys artėja prie analitinio sprendinio.



4 pav. Nestacionaraus temperatūros uždavinio sprendinio konvergavimas

Nestacionaraus šilumos pernešimo uždavinio sprendinys palygintas su [10] atliekant 100–1000 iteracijų (1 iteracija ~1 min. intervalu) pagal Galiorkino algoritmą [9, 10]. Iš 4 pav. seka, kad stacionarus sprendinys pasiekiamas maždaug per 200 iteracijų. Atsižvelgiant į tai, kad stacionaraus uždavinio sprendinys yra artimas analitiniam (4 pav.), matyti, kad ir per 100 iteracijų gautas sprendinys yra tikslesnis už [10] pateikiamą sprendinį.

Pagrindinės problemos ir jų sprendimai:

1. *Programos failų struktūrizavimas*. Nagrinėjamą programą sudaro 38 failai. *Matlab* sistema programą vykdo iš katalogo, kuris tuo metu atidarytas. Nustatyta, kad norint vykdyti kitame, o ne atidarytame, kataloge esančią programos dalį, katalogo pirmasis simbolis turi būti „+“, pvz., *+kitas*, o kreiptis į tame kataloge esantį programos kodo failą reikia taip: *kitas.funkcija*.

2. *Baigtinių elementų tinklelio vizualizavimas*. Vienas iš problemos sprendimo būdų buvo *Matlab* sistemoje integruotų *Mesh* ir *Grid* funkcijų naudojimas. Kilo problema dėl specifinių duomenų pateikimo šioms funkcijoms. Todėl sukurtas algoritmas generuoti ir vizualizuoti baigtinių elementų tinklelį, sudarytą iš tiesių ir mazgų.

Išvados

1. *Matlab* programavimo kalbos analizės ir objektinės programos realizacijos metu nustatyta, kad *Matlab* sistema turi gerai išplėtotą klasių kūrimo ir naudojimo įrankį. Vartotojai gali naudotis *Matlab* arba išorinių platformų klasėmis arba kurti savo klases.
2. Iš palyginamųjų programavimo kalbų galimybių analizės seka, kad *Matlab* klasių kūrimo galimybės yra panašios tiek į *C++,* tiek į *C#* ir *Java* kalbų klasių kūrimo galimybes – yra šių trijų kalbų mišinys. Be to, turi ir skirtumų.
3. Objektinės programos realizacijos metu nustatyta, kad dideli projektai *Matlab* sistemoje turėtų būti struktūrizuojami – klasių failai skirstomi į paketus, o jų turinys skaidomas į atskiruose failuose realizuojamus metodus.
4. Iš skaitinių rezultatų analizės seka, kad sukurtos šilumos pernešimo uždavinio sprendimo objektinės programos sprendiniai konverguoja, yra tikslesni už [10].
5. *Matlab* objektiškai orientuota programavimo kalba yra tinkama didelėms ir kompleksinėms taikomosioms programoms kurti.

Literatūra

1. Barauskas R., Belevičius R., Kačianauskas R., 2004, *Baigtinių elementų metodo pagrindai*. Vilnius: Technika.
2. Baumann G., Mnuk M., 2006, Elements–an object-oriented approach to industrial software development. *The Matematica Journal*. Vol. 10. No. 1. P. 161–186.
3. Chibisov D., Ganzha V. G., Zenger C., 2003, Object oriented finite element calculations using Maple. *Selcuk Journal of Applied mathematics*. Vol. 4. No. 1. P. 58–86.
4. Commend S., Zimmermann T., 2001, Object-Oriented Nonlinear Finite Element Programming: a Primer. *Advances in engineering software*. Vol. 32. Nr. 8. P. 611–628.
5. *Comparing Object-Oriented Features of Delphi, C++, C# and Java.* [interaktyvus] [žiūrėta 2012-12-12]. Prieiga per internetą:<http://www.derangedcoder.net/programming/general/comparingObjectOrientedFeatures.html>.
6. *Matlab. ExternalInterfaces.* [interaktyvus] [žiūrėta 2012-12-12]. Prieiga per internetą: <http://www.mathworks.se/help/techdoc/matlab_external/bp_kqh7.html>.
7. *Matlab. Object-Oriented Programming.* [interaktyvus] [žiūrėta 2012-12-12]. Prieiga per internetą: <http://www.mathworks.se/help/techdoc/ref/brk7uzk.html>.
8. Rodriguez N. J., Gaytan J. L. S., Herrera J. F. B., Guerrero F. A. R., 2005, A finite element method for heat transfer with convection effects embedded into the Capacitance. *WIT Transactions on Modelling and Simulation*. Vol. 41. P. 81–91.
9. Seetharamu K. N., Nithiarasu P., Lewis, R. W, 2004, *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. Chichester: John Wiley & Sons Ltd..
10. Sergerlind L. J., 1984, *Applied Finite Element Analysis (2nd ed.).*New York: John Wiley & Sons Ltd.
11. *TIOBE Programming Community Index for October 2012.* [interaktyvus] [žiūrėta 2012-12-12]. Prieiga per internetą: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.
12. Turskienė S., 2012, The finite element models of plates junction zones in heat transfer problems. *Mechanika'2012: 17th International conference,* April 12–13. Kaunas, Lithuania / Kaunas University of Technology, Lithuanian Academy of Science, IFTOMM National Committee. ISSN 1822-2951. P. 316-321. Kaunas: Technologija.
13. Zabaras N., Srikanth A., 1999, An object-oriented programming approach tote Lagrangian FEM analysis of large in elastic deformations and metal-forming processes. *International journal for numerical methods in engineering*. Vol. 45. P. 399–445.

COMPARATIVE ANALYSIS OF POSSIBILITIES FOR OBJECT-ORIENTED PROGRAMMING IN *MATLAB* LANGUAGE

Aurimas Šimkus, Sigita Turskienė

Summary

Statistically, object-oriented programming is the most popular programming paradigm nowadays. This paradigm is widely used in systems tackling real world issues. It offers a convenient way to solve such issues by using computer mathematics systems. The paper analyses possibilities for object-oriented programming in Matlab language. They are compared with capabilities offered by C++, C# and Java programming languages. A complex object-oriented program in Matlab language has been created. This program enables solving a heat transfer problem by using a finite element method. Program testing results are also provided.

Keywords: object-oriented programming, computer mathematics system Matlab, heat transfer problem, finite element method.

MATLAB KALBOS OBJEKTINIO PROGRAMAVIMO GALIMYBIŲ LYGINAMOJI ANALIZĖ

Aurimas Šimkus, Sigita Turskienė

Santrauka

Šiuo metu objektinis programavimas yra statistiškai populiariausia programavimo paradigma. Ji plačiai naudojama realaus pasaulio problemas sprendžiančiose sistemose, kurias patogu spręsti kompiuterinėmis matematikos sistemomis. Darbe analizuojamos *Matlab* kalbos objektinio programavimo galimybės, kurios lyginamos su C++, C# bei Java kalbų galimybėmis. *Matlab* kalba sukurta kompleksinė objektinė programa, kuri įgalina spręsti šilumos perdavimo uždavinį baigtinių elementų metodu. Pateikti programos testavimo rezultatai.

Prasminiai žodžiai: objektinis programavimas, kompiuterinė matematikos sistema Matlab, šilumos pernešimo uždavinys, baigtinis elementų metodas.

Įteikta 2013-04-15