Liet. matem. rink. **45**, *spec. nr.*, 2005, 113–118

Programinio komponento ontologijos formalioji specifikacija

L. BAGUŠYTĖ, Audronė LUPEIKIENĖ (MII)
el. paštas: audronel@ktl.mii.lt

1. Įvadas

Komponentinis programų ir informacinių sistemų kūrimas grindžiamas praktine patirtimi, paradigma neturi reikiamų teorinių rezultatų. Bazinės šios paradigmos sąvokos dažnai nėra vienareikšmiškai apibrėžtos [3]. Tai galima pastebėti netgi vienos komponentinės technologijos ribose. Komponento ontologija reikalinga tam, kad būtų nustatyta, kokius terminus vartoti ir ką tie terminai reiškia. Norint, kad terminai būtų vienareikšmiškai interpretuojami, ir kad juos galėtų naudoti kompiuterizuota sistema, reikia formalios ontologijos. Tai leistų kūrėjams ir generatyviesiems artefaktams bendrai naudotis sukauptomis žiniomis ir automatizuoti sistemų kūrimo procesą. Tačiau sukurtų komponento ontologijų nėra, jų sudarymas neretai pakeičiamas metamodelio, apimančio ontologijos ir specifikacijos elementus, sukūrimu [7], [9].

Programinio komponento ontologijos sukūrimo problemas sąlygoja nepakankami teoriniai rezultatai ontologijų ir jų specifikavimo srityje. Pavyzdžiui, T. Gruberio [4] pateikta ontologijos apibrėžtis – išreikštiniu būdu nusakyta konceptualizacijos specifikacija – priklauso nuo terminų „specifikacija“ ir „konceptualizacija“ interpretavimo. Darbe [6], analizuojant naudojamas termino „ontologija“ interpretacijas, daroma išvada, kad ontologija turėtų būti suprantama dviem prasmėmis: kaip ontologinė teorija (loginė teorija, kurios aksiomos turi būti teisingos visuose galimuose pasauliuose, nagrinėjamuose naudojant tam tikrą konceptualizaciją) ir kaip konceptualizacijos sinonimas. Dauguma ontologijų specifikacijų turi tuos pačius jas sudarančius elementus [4], [8]. Tačiau, kaip pažymi darbo [4] autorius, ne visi šie elementai būna visose ontologijų specifikacijose, skiriasi specifikacijų detalumas ir išsamumas.

Šiame straipsnyje nagrinėjamos programinio komponento ontologijos specifikavimo ir jos formalizavimo problemos. Darbe pagrindžiamas naudojamos ontologijos interpretacijos pasirinkimas, aptariamas ontologijos specifikacijos turinys ir šios specifikacijos sudarymas. Pateikiama programinio komponento mereologinė teorija – komponento formaliosios specifikacijos fragmentas. Tyrimai atliekami verslo, informacinių ir programų sistemų bendros infrastruktūros kūrimo kontekste [2], t.y. atsižvelgiama į tai, kad gautus rezultatus būtų galima panaudoti ir informacinių sistemų komponentams kurti.

2. Ontologijos formalioji specifikacija ir jos sudarymas

Pagal [8] ontologija specifikuojama penketu

$$(C, R, H^c, rel, A^o),$$

čia C yra konceptų identifikatorių aibė, R ryšių identifikatorių aibė ir C∩R=Ø; H^c yra konceptų hierarchija arba taksonomija. $H^c(C_1, C_2)$ reiškia, kad C_1 yra C_2 subkonceptas, o funkcija $relR \to C \times C$, nusako konceptų netaksonominius ryšius. A^o yra aksiomų rinkinys.

Darbe [4] pastebima, kad dauguma specifikuojamų ontologijų turi šiuos penkis pagrindinius elementus, tačiau ne visi išvardintieji elementai privalo būti ontologijoje. Be to, gali skirtis ontologijos specifikacijos detalumas.

Specifikuojant komponentų ontologiją naudosime šiuos konceptų identifikatorius, t.y. terminus, – komponentas, interfeisas, loginis komponentas, portas (sąveikos taškas), identifikatorius, vidinis artefaktas (pvz., modulis). Ryšių identifikatorių pavyzdžiai: jungia, slepia, realizuoja, apima, agreguoja, deleguoja, naudoja, teikia. Interfeisas ir komponentas yra loginio komponento dalys; identifikatorius, sąveikos taškas ir vidiniai artefaktai yra tikrosios komponento dalys, komponentas yra sudėtinio komponento dalis. Šiais terminais išreiškiama darbe naudojama nagrinėjamos dalykinės srities samprata.

Prieš nagrinėjant konceptų taksonomiją, netaksonominius ryšius ir aksiomas, turi būti pasirinkta ontologijos interpretacija. Darbe [6], analizuojant naudojamas termino „ontologija“ interpretacijas, daroma išvada, kad ontologija turėtų būti suprantama dviem prasmėmis: kaip ontologinė teorija ir kaip konceptualizacijos specifikacija. Taigi, pirmuoju atveju, ontologija – tai loginė teorija, skirta išreikšti ontologines žinias. Ontologines žinias atitinka aksiomos, galiojančios visuose galimuose pasauliuose, nagrinėjamuose naudojant tam tikrą konceptualizaciją. Nuo konceptualizacijos pasirinkimo priklauso, kaip struktūrizuojama dalykinė sritis, kitaip sakant, kokia yra naudojama kategorijų sistema. Sudarant tam tikros dalykinės srities, pavyzdžiui, programinio komponento, ontologinę teoriją, turi būti peržiūrėta aksiomų aibė. Tuo atveju ontologinė teorija papildoma dalykinės srities aksiomomis ir įvardinama kaip dalykinė teorija.

Kai ontologija interpretuojama kaip konceptualizacijos specifikacija, yra kuriamas lingvistinis artefaktas. (Prisiminkime, kad darbo [6] autoriai, sakydami, jog konceptualizacijos specifikacijos tiksli prasmė priklauso nuo terminų „specifikacija“ ir „konceptualizacija“ supratimo, išanalizavo visas galimas šių terminų prasmes.) Tai reiškia, kad šios interpretacijos atveju yra sudaromas žodynas, kuriame pateikiami terminai, žymintys esybes ir ryšius tarp jų. Be to, turi būti apibrėžiama gramatika, kad žodyno terminai būtų paaiškinami prasmingais sakiniais. Taigi, ontologija yra kontroliuojamas žodynas, kuriam užrašyti turi būti pasirinkta tinkama kalba. Tačiau šia kalba užrašant taksonomijas, netaksonominius ryšius ir ypač aksiomas dažniausiai suliejami lingvistinis ir dalykinės srities lygmenys. Tai reiškia, kad ontologija pradedama interpretuoti ir kaip lingvistinis artefaktas, ir kaip dalykinės srities teorija.

Bendru atveju, ontologijų tikslas yra apibūdinti konceptualizaciją, kiek įmanoma ribojant loginės kalbos simbolių interpretacijas ir siekiant vienareikšmiško susitarimo

dėl šia kalba aprašytų žinių [5]. Tačiau, kaip jau minėjome, siekiant šio tikslo dažniausiai kuriamas žodyno ir dalykinės srities teorijos junginys. Šiame darbe ontologija yra suprantama kaip dalykinės srities teorija, o naudojamos loginės kalbos simboliai žymi atitinkamus dalykinės srities elementus.

Ontologijos yra sudėtingi artefaktai, jų sudarymas reikalauja tinkamo išmanymo ir pakankamai daug sąnaudų. Kaip parodyta [1], norint įgalinti ir palengvinti komponento ontologijos specifikacijos sukūrimą, turi būti taikomi abstrakcijos, dekompozicijos ir struktūrizavimo (moduliarizavimo) principai, kitais žodžiais sakant, turi būti:

- surenkama iš mažesnių pakartotinai naudojamų ontologijų,
- naudojamos fundamentinės ontologijos,
- konstruojama specializuojant ontologijas, aprašančias bendresniųjų dalykinių sričių konceptualizacijas [2].

Komponento ontologija turi būti konstruojama iš prasminių mažesnių ontologijų, kurių savybės yra patikrintos ir žinomos. Komponento ontologijai sudaryti reikalingų fundamentinių ir kitų pakartotinai naudojamų ontologijų pavyzdžiai yra mereologija, topologija, matematinių sąryšių ontologija, komunikavimo ontologija ir pan. Fundamentine yra vadinama ontologija, kuri aprašo metakonceptualizaciją, apimančią nuo dalykinės srities nepriklausomas teorijas, tokias kaip identifikavimo, priklausomybių, dalies–visumos ir pan. Kuriant komponento ontologiją, abstrakcijos principas realizuojamas įvedant aukštesniojo lygmens ontologiją, aprašančią bendrąsias sąvokas, o po to žemesniojo lygmens – kompiuterizuotos sistemos komponento ir programinio komponento – ontologijas.

Kaip jau buvo pabrėžta, komponento ontologijos išsamios specifikacijos sudarymas yra sunkus uždavinys (ontologijos išsamios specifikacijos sudarymas yra problemiškas netgi jau formalizuotoms dalykinėms sritims), todėl šiame darbe pateiksime tik specifikacijos dalį – programinio komponento mereologinę teoriją.

3. Komponento mereologinė teorija

Mereologijos ontologija apibrėžia dalies ir visumos ryšį bei šio ryšio savybes. Kitaip tariant, apibrėžiama dalių „suma“, kuri turi būti suprantama ne daugiau kaip nurodytų dalių visuma. Pavyzdžiui, interfeisas yra komponento dalis, komponentas yra sudėtinio komponento dalis. Ryšio savybių pavyzdžiai yra antisimetrija, tranzityvumas. Be to, ši ontologija nusako visumą sudarančių dalių tarpusavio santykį (persidengia, mereologiškai lygūs ir kt.).

Komponento mereologinė teorija turi nusakyti komponento (apimant sudėtinį) kaip visumos ir jo sudėtinių dalių santykį. Šiame darbe ji grindžiama pirmos eilės predikatų logika ir Hilberto skaičiavimu su identiškumu. Taigi, naudosime pirmos eilės kalbą, kurioje binarinę predikatinę konstantą P interpretuosime kaip dalies ir visumos, o PP – kaip tikros dalies ir visumos ryšį.

Kadangi bazinė mereologijos teorija [10], [11] yra apibrėžiama refleksyvumo, simetrijos ir tranzityvumo aksiomomis dalies–visumos predikatui, visų pirma aptarsime šias aksiomas sudarant komponento ontologiją. Pastebėsime, kad sudarant komponento mereologinę teoriją, bendrosios mereologijos teorijos aksiomos neturėtų būti perkeliamos automatiškai – svarbu atsižvelgti į konkrečios dalykinės srities ypatumus.

Dalies–visumos ryšys P yra refleksyvus, antisimetrinis ir tranzityvus. Šios aksiomos galioja fundamentinėje mereologinėje teorijoje ir gali būti perkeliamos į komponento mereologinę teoriją. Tačiau programinio komponento atveju dalis turi būti griežtai skiriama nuo visumos, t.y. nagrinėjamos tikrosios dalys:

$$PPxy =_{df} Pxy \wedge \neg Pyx. \tag{1}$$

Tikrosios dalies atveju refleksyvumas negalioja, t.y. tikroji dalis nėra pačios savęs dalis:

$$\neg PPxx. \tag{2}$$

Be to, turi būti tenkinamas dar griežtesnis reikalavimas, taip vadinamas papildymo principas:

$$PPxy \to \exists z(PPzy \wedge \neg Ozx). \tag{3}$$

Tai reiškia, kad komponentas konstruojamas iš bent dviejų sudedamųjų dalių. Be to, nesunku specifikuoti, kad elementarus komponentas konstruojamas iš bent dviejų primityvų, o sudėtinis komponentas gali būti sudarytas iš vieno elementaraus ir bent vieno primityvo.

Simetriškumas programinio komponento atveju taip pat negalioja. Kitaip tariant, komponentas negali būti savo dalies dalimi:

$$PPxy \wedge PPyx \to x = y. \tag{4}$$

Programinio komponento atveju tranzityvumą

$$PPxy \wedge PPyz \to PPxz \tag{5}$$

sąlygoja sudėtinio komponento ypatumai, kurie priklauso nuo šio komponento architektūros stiliaus. Tiksliau sakant, tai priklauso nuo ryšių, naudojamų komponentams sujungti į sudėtinį tipų. Paprastojo sujungimo atveju tranzityvumo aksioma negalioja.

Turint (1)–(4), gali būti įvesti reikiami komponento mereologinės teorijos predikatai: lygybė

$$EQxy =_{df} Pxy \wedge Pyx, \tag{6}$$

tikrasis išplėtimas

$$PExy =_{df} \neg Pxy \wedge Pyx, \tag{7}$$

persidengimas (sudėtinio komponento atveju persidengimas netaikomas jį sudarančių komponentų interfeisams)

$$Oxy =_{df} \exists z(Pzx \wedge Pzy), \tag{8}$$

perdengimas (angl. *underlap*)

$$Uxy =_{df} \exists z(Pxz \wedge Pyz). \tag{9}$$

Kadangi ne kiekvienas sutvarkymas gali būti kvalifikuojamas kaip elemento ir visumos ryšys, komponento mereologinė teorija turi apimti tai patikslinančius principus. Apibrėšime kompozicijos ir dekompozicijos principus. Norint nagrinėti kaip atskirus elementus ne tik visumą sudarančias dalis, bet ir pačią visumą, reikia turėti galimybę operuoti mereologine suma, kuri yra nusakoma šia aksioma:

$$\psi xy \rightarrow \exists z \forall w(Owz \leftrightarrow (Owx \vee Owy)). \tag{10}$$

Iš komponento pašalinus kurią nors jo sudėtinę dalį, visada turi kažkas likti, jei lieka egzistuoti pats komponentas. Todėl komponento dekompozicijai nusakyti įvedamas taip vadinamasis stiprusis papildymas:

$$\neg Pyx \rightarrow \exists z(Pzy \wedge \neg Ozx). \tag{11}$$

Programiniai komponentai turi būti vienareikšmiškai identifikuojami. Norint atskirti vieną komponentą nuo kito, fundamentinėje mereologinėje teorijoje siūloma naudoti mereologinį ekstensionalumo principą. Tačiau programinio komponento atveju šis principas negalioja, t.y. tų pačių sudėtinių dalių buvimo nepakanka, kad būtų tapačios ir visumos:

$$\neg((\exists z PPzx \vee \exists z PPzy) \rightarrow (x = y \leftrightarrow \forall z(PPzx \leftrightarrow PPzy))). \tag{12}$$

Sudėtinis komponentas yra sujungtų elementarių komponentų visuma. Elementarus komponentas turi būti sudarytas iš atomarinių dalių, o sudėtinis – iš atomarinių ir neatomarinių dalių. Atomas programinio komponento atveju apibrėžiamas taip:

$$Ax =_{df} \neg \exists y PPyx. \tag{13}$$

Be to, galioja atomiškumo principas:

$$\exists y(Ay \wedge Pyx). \tag{14}$$

Pastebėsime, kad programinio komponento atveju negalioja ekstensionalumo principo variantas atomariniam atvejui:

$$\neg(x = y \leftrightarrow \forall z(Az \rightarrow (Pzx \leftrightarrow Pzy))). \tag{15}$$

Kitaip tariant, tiek elementarių, tiek sudėtinių komponentų identiškumui nustatyti nepakanka juos sudarančių dalių tapatumo.

Turint (1)–(15), nusakysime komponentą ir loginį komponentą, apibrėždami predikatines konstantas atitinkamai K ir LK:

$$Kxyzw =_{df} \forall x \exists y \exists z \exists w(PPyx \wedge PPzx \wedge PPwx), \tag{16}$$

$$LKxyz =_{df} \forall x \exists y \exists z(Pyx \wedge PPzx). \tag{17}$$

4. Išvados

Straipsnyje parodyta, kad programinio komponento ontologija yra sudėtingas, nevienareikšmiškai suprantamas artefaktas. Todėl, prieš sudarant specifikaciją, reikia

pasirinkti naudojamą ontologijos interpretaciją. Kai tai nėra padaroma, neišvengiamai neleistinai sujungiamos lingvistinė bei konkrečios dalykinės srities teorijos.

Šiuo metu nėra sukurtos komponento ontologijos, ji pakeičiama komponentų modelių aprašais, kurie yra menkai pritaikyti automatizuotam programinių komponentų kūrimui. Kaip parodyta darbe, komponento ontologija turi būti specifikuojama panaudojant fundamentines ir kitas mažesnes pakartotinai naudojamas ontologijas. Komponento mereologinė teorija turi nusakyti komponento (apimant sudėtinį) kaip visumos ir jo sudėtinių dalių santykį. Tačiau komponento mereologinės teorijos nepakanka, kad būtų apibrėžti šie dalykai:

- komponento sukonstravimo iš atomarinių dalių būdas,
- sudėtinio komponento sukonstravimo iš atomarinių ir neatomarinių dalių būdas,
- komponento tapatumas.

Literatūra

1. L. Bagušytė, A. Lupeikienė, Programinio komponento ontologija ir jos sudarymas, *Informacijos mokslai*, **34**, 119–124 (2005).
2. A. Caplinskas, A. Lupeikiene, O. Vasilecas, Shared conceptualisation of business systems, information systems and supporting software, in: H.-M. Haav, A. Kalja (Eds.), *Databases and Information Systems II. Fifth International Baltic Conference "BalticDB&IS'2002"*, Tallinn, Estonia, June 3–6, 2002, Selected Papers, Kluwer Academic Publishers, Dordrecht/Boston/London (2002), pp. 109–120.
3. V. Giedrimas, A. Lupeikienė, Komponento specifikacijos formalizavimas, *Liet. matem. rink.*, **44** (spec. nr.), 276–280 (2004).
4. T. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition*, **5**(2), 199–200 (1993).
5. N. Guarino, Understanding, building and using ontologies, *International Journal of Human-Computer Studies*, **46**(2–3), 293–310 (1997).
6. N. Guarino, P. Giaretta, Ontologies and knowledge bases: towards a terminological clarification, in: N. Mars (ed.), *Towards Very Large Knowledge Bases*: *Knowledge Building and Knowledge Sharing*, IOS Press (1995), pp. 25–32.
7. F. Luders, K.-K. Lau, S.-M. Ho, Specification of software components, in: I. Crnkovic, M. Larsson (Eds.), *Building Reliable Component-Based Software Systems*, Artech House Publishers (2002), pp. 22–37.
8. A. Maedche, *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers (2002).
9. Z. Stojanovic, *A Method for Component-Based and Service-Oriented Software Systems Engineering*, Doctoral Dissertation, Delft University of Technology, The Netherlands (2005).
10. A. Varzi, Parts, wholes, and part-whole relations: the prospects of mereotopology, *Data and Knowledge Engineering*, **20**, 259–286 (1996).
11. A. Varzi, Philosophical issues in the logic of space, in: M. Aiello, I. Pratt-Hartmann, J. van Benthem (Eds.), *The Logic of Space*, Kluwer Academic Publishers (2005).

SUMMARY

L. Bagušytė, A. Lupeikienė. Formalisation of the sofware component ontology specification

This paper is devoted to the development and specification problems of software component ontology. The formalisation of the proposed component ontology is presented, too. The choice of ontology interpretation is motivated. The paper discusses the content of the ontology specification and its development. The mereological theory of software component, one of the constituent parts of a formal component specification, is presented.

Keywords: component, ontology, ontology specification, specification formalisation.