*Liet. matem. rink.*, **44**, *spec. nr.*, 2004, 276–280

# Komponento specifikacijos formalizavimas

Vaidas GIEDRIMAS, Audronė LUPEIKIENĖ (MII)
el. paštas: audronel@ktl.mii.lt

## 1. Įvadas

Šiuolaikinės informacinės sistemos yra sudėtinė verslo sistemų dalis. Informacinėms sistemoms realizuoti reikalingos atitinkamos programų sistemos. Todėl nuolat kintant aplinkos reikalavimams reikia turėti tokią sistemų kūrimo infrastruktūrą ir metodus, kurie leistų nesunkiai modifikuoti egzistuojančias sistemas. Taigi, infrastruktūra turi apimti ne tik techninę įrangą, organizacines ir instrumentines priemones, bet ir pakartotinai panaudojamus artifaktus, o sistemų kūrimas turi būti automatizuojamas. Komponentinis programų sistemų kūrimas yra vis plačiau naudojamas, tačiau dauguma pasiūlytų ir naudojamų šio būdo metodų yra empirinio pobūdžio, paradigma neturi reikiamų teorinių rezultatų koncepciniame ir specifikavimo lygmenyje.

Automatiniu būdu kuriant programų sistemas, vienas iš svarbiausių uždavinių yra komponento specifikavimas. Siekiant jį išspręsti reikia išspręsti šiuos dalinius uždavinius: apibrėžti komponento modelį, nustatyti komponento modelio reikalavimus programų sistemų sintezės kontekste, formaliai specifikuoti komponento modelį. Tačiau nei vienas iš šių uždavinių nėra pakankamai išnagrinėtas. Ypač mažai yra žinoma kaip formalizuoti komponento specifikaciją, siekiant ją panaudoti programų sistemų sintezėje. Komponento modelio samprata buvo suformuluota darbe [1]. Šiame straipsnyje nagrinėjama komponento specifikacija ir jos formalizavimas. Specifikuojant komponentus siūloma atsižvelgti į skirtingas komponentų rūšis ir įvesti jų sutvarkymą. Apibrėžiami bendrieji elementarus ir sudėtinis komponentai, kuriuos specializuojant reikia skirti dar dviejų rūšių komponentus: turinčius vidinę būseną ir be jos. 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. Komponento modelis

Kalbant apie komponentus paprastai sakoma (pvz., [3], [4]), kad tai yra fizinė keičiama sistemos dalis, kuri atitinka ir realizuoja tam tikrą interfeisų aibę. Tačiau bendruoju atveju ši samprata nėra nei vienareikšmiška, nei išsami. Ji apima tiek binarinius modulius, tiek informacinės sistemos sudėtines dalis. Komponentinių programų sistemų kūrimo pradininkai pagrindinį dėmesį kreipė į komponentų atitikimą naudojamos kūrimo metodikos ir infrastruktūros standartams. Nors komponentinės technologijos, tokios kaip COM, CORBA, JavaBeans, kt. yra plačiai naudojamos, faktiškai jos grindžiamos eksperimentų rezultatais.

Komponento modelis nusako pagrindines sąvokas ir jų tarpusavio ryšius. Komponento modelį apibrėšime kaip trivietį kortežą:

$$M =< K, SM, G >,$$

kur $K$ – elementaraus komponento apibrėžtis,

$SM$ – sudėtinio komponento apibrėžtis,

$G$ – komponento gyvavimo ciklo modelis.

Sudėtinis komponentas yra tarpusavyje sujungtų elementarių komponentų visuma. Tai reiškia, kad nusakant sudėtinį komponentą reikia apibrėžti komponavimą: kompozicijų rūšis, konstruktorių aibę. Komponento kūrimas gali būti sudėtingas ir ilgas procesas. Todėl svarbu nustatyti šio proceso darbus, kitaip tariant, apibrėžti gyvavimo ciklą. Be to, su gyvavimo ciklu gali būti siejami komponento nefunkcinių reikalavimų realizavimo klausimai ir tam tikra technologija. Pastebėsime, kad, nors nefunkciniai reikalavimai yra svarbūs, pagrindinis dėmesys kaip taisyklė yra skiriamas komponento funkcionalumo specifikavimui ir realizavimui.

## 3. Komponento specifikacija

Darbuose [5], [6] pateikiami bandymai formaliai specifikuoti komponento modelį. Tačiau, kaip reikėtų tai formalizuoti, vis dar nėra apibrėžta. Pirma, formalus komponento specifikavimas faktiškai suvedamas į tinkamos specifikavimo kalbos pasirinkimą ir taikymą arba jos sukūrimą. Antra, pateikiama formalizuota arba formali specifikacija kai taisyklė aprašo savitą tam tikram tikslui sukurtą komponento modelį, o ne apibrėžia bendrąsias komponentinės paradigmos sąvokas. Pavyzdžiui, darbe [5] pabrėžiama, kad, nepaisant didelės svarbos, COM komponentai vis dar neturi apibrėžtos formalios specifikacijos. Darbe [5] formalizuojama tik dalis svarbių COM taisyklių, o siekiant apibrėžti neformalias darbo su COM komponentais taisykles, sukuriama speciali komponentų specifikavimo kalba COMEL, kuria galima užrašyti ir formalizuotas, ir neformalias taisykles.

Darbe [6] parodoma, kaip formalizuoti komponentus ir jų kontraktus specifikavimo lygmenyje. Tam siūloma naudoti išskirstytą laiko logiką. Tačiau šiame darbe nagrinėjami komponentai, iš kurių sudaromos ne programų, o informacinės sistemos, t.y., komponentų specifikacija apima taip vadinamą verslo modelį [7]. Taip pat šie komponentai suprantami kaip išskirstyti. Be to, komponentų specifikacija formalizuojama atsižvelgiant į tai, kad jie yra realizuojami taikant objektinę metodiką.

Mūsų nuomone, siekiant formalizuotai specifikuoti komponentą, visų pirma reikia atsižvelgti į skirtingas komponentų rūšis ir specifikaciją konstruoti kaip apibendrinimo-specializacijos hierarchiją. Kitaip tariant, komponentų rūšims nustatytas sutvarkymas atspindi apibendrinimo ryšį. Aukščiausiame hierarchijos lygmenyje turi būti sudaroma bendroji komponento specifikacija, žemesniuose lygmenyse – ji specializuojama iki konkrečiose komponentinėse technologijose naudojamų komponentų specifikacijų.

Be to, automatiniu būdu kuriant programų sistemas, komponentas turi tenkinti šiuos reikalavimus:

- komponentai neturi priklausyti vienas nuo kito, kitaip tariant, komponento funkcionalumas turi būti atskirtas nuo jo realizacijos;

- turi būti galima rasti ir panaudoti reikiamus komponentus, kitaip tariant, komponentas turi turėti vienareikšmišką specifikaciją, atskirtą nuo jo turinio.

Atsižvelgdami į komponento modelio sampratą ir apibrėžtus komponento reikalavimus, bendrąjį elementarųjį komponentą aprašysime penkiaviečiu kortežu:

$$K =< P, R, \Phi, E, T >,$$

čia

$P = (p_1, p_2, \ldots, p_n)$ – funkcionalumui realizuoti pateikiama atributų aibė, $n > 0$;
$R = (r_1, r_2, \ldots, r_{m1}) \cup (r'_1, r'_2, \ldots, r'_{m2})$ – rezultato atributų aibė, $m_1 > 0$, $m_2 \geqslant 0$;
$\Phi = (\varphi_1, \ldots, \varphi_i) \cup (\varphi_{i+1}, \ldots, \varphi_k)$ – komponento funkcionalumą nusakančių sudėtinių funkcijų aibė, čia $i, k \geqslant 1$,

$$\Phi' = (\varphi_1, \ldots, \varphi_i), \quad \phi_j\colon\ p_1 \times p_2 \times \ldots \times p_l \Rightarrow r'_j, \quad j = 0, \ldots, m_2,\ l \leqslant n,$$

$$\Phi'' = (\varphi_{i+1}, \ldots, \varphi_k), \quad \phi_j\colon\ p_1 \times \ldots \times p_l \times \varphi_1 \times \ldots \times \varphi_s \Rightarrow r_j,$$

$$j = 1, \ldots, m_1, \quad s = 1, \ldots, i;$$

$$\varphi_j = (\psi_j^1, \psi_j^2, \ldots, \psi_j^l), \quad j = 1, \ldots, k;\ l \geqslant 1;$$

$E = \bigcup_{j=1}^{k} \{\psi_j^1, \psi_j^2, \ldots, \psi_j^s\}$ – komponento elgseną nusakanti baigtinė funkcijų $\phi_j$ sudėtinių elementų leistinų sekų aibė, $s > 0$;

$T$ – trigeris, kitaip tariant, funkcijų sekų $\{\psi_j\}$ inicijavimo sąlygos.

Taigi, bendrasis komponentas suprantamas kaip paslaugas teikiantis ir naudojantis elementas, kurio funkcionalumą nusakanti funkcija $\Phi$ yra realizuota tam tikroje kalboje. Aibė $\Phi'$ apibrėžia reikiamas paslaugas, $\Phi''$ – komponento $K$ teikiamas paslaugas. Pastebėsime, kad labai panašiai komponento funkcionalumas apibrėžiamas [8] darbe, tačiau, mūsų nuomone, minėtame darbe pateikta paprastojo komponento apibrėžtis labiau tinka moduliui, o ne komponentui nusakyti.

Grindžiant komponento sampratą bazinėmis reikalaujamų ir teikiamų interfeisų sąvokomis, kaip tai daroma, pavyzdžiui, [9] darbe, atitikmuo interfeisui mūsų modelyje yra sudėtinė funkcija $\phi_j$. Tačiau pastebėsime, kad [9] darbe taip ir lieka neaišku, ką autoriai vadina interfeisu, ir ar skiriasi tarpusavyje reikalaujamas ir teikiamas interfeisas.

Sudėtiniams komponentams sudaryti naudojami įvairių rūšių konstruktoriai. Pavyzdžiais gali būti sujungimo, agregavimo, išplėtimo konstruktoriai. Leistina konstruktorių aibė priklauso nuo požiūrio į komponentą. Nagrinėjant komponentą kaip „juodosios dėžės" abstrakciją, elementarūs komponentai jungiami naudojant sujungimo konstruktorių. Kadangi automatizuotam sudėtinių komponentų kūrimui bus taikoma struktūrinė sintezė, toliau nagrinėsime tik „juodosios dėžės" atvejį.

Sakysime, kad komponentai $K_1$ ir $K_2$ suderinami sujungimo prasme, jei tenkinamos visos šios sąlygos:

$\Phi'_1 \cap \Phi''_2 \neq \emptyset$ ir jei $\Phi'_1 \neq \emptyset$ bent vienam $\phi_j$ $(j = 1, \ldots, m_2)$,

$\Phi'_1 \cap \Phi''_1 = \emptyset$,

$\exists \phi_j \in \Phi'_1 \cap \Phi''_2$ toks, kad $E_1 \cap E_2 \neq \emptyset$.

Bendrąjį sudėtinį komponentą aprašysime septynetu:

$$SK =< K, SP, SR, S\Phi, SE, ST, J >,$$

čia $K = (k_1, k_2, \ldots, k_s)$ – elementarių komponentų, jungiamų į sudėtinį ir suderinamų sujungimo prasme, aibė, $s \geqslant 2$;

$SP = (P_{k_1} \cup P_{k_2} \cup \ldots \cup P_{k_s})$ – funkcionalumui realizuoti pateikiama atributų aibė;

$SR = (R_{k_1} \cup \ldots \cup R_{k_s}) \backslash (R'_{k_1} \cup \ldots \cup R'_{k_{s-1}})$ – rezultato atributų aibė[1];

$S\Phi = (\Phi_{k_1} \cup \ldots \cup \Phi_{k_s}) \backslash (\Phi'_{k_1} \cup \ldots \cup \Phi'_{k_{s-1}})$ – sudėtinio komponento funkcionalumą nusakanti sudėtinių funkcijų aibė;

$SE = \bigcup_{\varphi_j \in S\Phi} \{\psi_j^1, \psi_j^2, \ldots, \psi_j^s\}$ – komponento elgsena, nusakoma leistinų sekų aibe, $s > 1$;

$ST$ – funkcijų sekų $\{\psi_j\}$ inicijavimo sąlygos;

$J$ – sujungimo konstruktorius.

Trečiąjį komponento modelio elementą – komponento gyvavimo ciklą – aprašysime dviejų lygmenų baigtinių būsenų mašinų formalizmu:

$$M =< B, b_{prad}, I, f >,$$

čia $B = \{b_1, b_2, \ldots, b_n\}$ – komponento leistinų būsenų aibė, ir $\tau(b_i) \neq \tau(b_{i+1})$, kai $\tau$ yra būsenos tipą apibrėžianti funkcija;

$b_{prad} \in B$ – pradinė komponento būsena jo gyvavimo cikle;

$I$ – įvykių, sužadinančių būsenas keičiančių funkcijų realizavimo procedūras, aibė;

$f$: $B \times I \Rightarrow B$ – perėjimo iš būsenos į būseną funkcija.

Komponentai gali tenkinti skirtingus nefunkcinius reikalavimus. Pavyzdžiui, gali būti sukurtos skirtingo patikimumo, saugumo ir pan. funkcijos $\Phi$ realizacijos. Tai reiškia, kad kai kurio tipo būsenų gali būti daug, todėl antrajame hierarchijos lygmenyje turi būti apibrėžta komponento versijų kūrimo procedūra. Ją nusakysime šitaip:

$$PRC^k =< V^k, v^k_{prad}, I^k, PR^k, h^k >,$$

čia $V^k = \{v_1^k, v_2^k, \ldots, v_n^k\}$ – $k$-tojo komponento gyvavimo ciklo leistinų vieno tipo būsenų aibė,

$v^k_{prad} \in V$ – pradinė tam tikro tipo būsena;

$I^k$ – įvykių, sužadinančių būsenas keičiančias procedūras, aibė;

$PR^k = \{pr_1^k, pr_2^k, \ldots, pr_m^k\}$ – būsenas keičiančių procedūrų aibė;

$h^k$: $V^k \times I^k \times PR^k \Rightarrow V^k$ – perėjimo iš būsenos į būseną funkcija.

## 4. Išvados

Šiuo metu naudojami komponento specifikacijos formalizavimo būdai nepakankami arba netinka komponentinių programų sistemų sintezei. Norint automatiniu būdu kurti

[1] Siekiant paprastumo daroma prielaida, kad $\Phi'_{i1}$ turi tik vieną elementą ($i = 1, \ldots, s$), kas neturi įtakos sudėtinio komponento sąvokos apibrėžties ir formalizavimo esmei.

komponentines programų sistemas reikia aiškiai atskirti komponento specifikaciją nuo jo turinio, apibrėžti sudėtinio komponento sąvoką ir gebėti jį sukonstruoti. Atsižvelgiant į tai, komponento sąvoka darbe formalizuojama kaip „juodosios dėžės" abstrakcija ir suformuluojamos elementarių komponentų jungimo į sudėtinį sąlygos. Apibrėžtosios sąvokos ir jų formalizavimo būdas įgalina specifikuoti funkcinius ir nefunkcinius komponentų reikalavimus.

## Literatūra

1. V. Giedrimas, Komponento modelis struktūrinės programų sintezės kontekste, *Informacijos mokslai*, **26**, 246–250 (2003).
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. G. Booch, *Software Components with Ada: Structures, Tools and Subsystems*, Benjamin/Cummings Publishing Company Inc. (1994).
4. F. Bachmann *et al.*, *Volume II: Technical Concepts of Component-Based Software Engineering*, Technical Report CMU/SEI-2000-TR-008 ESC-TR-2000-007, Software Engineering Institute (2000).
5. R. Ibrahim, C. Szyperski, Formalization of component object model (COM) – the COMEL language, in: S. Demeyer, J. Bosch (Eds.), *Object Oriented Technology – ECOOP'98*, Lecture Notes in Computer Science, **1543** (1998).
6. J.K. Filipe, A logic-based formalization for component specification, *Journal of Object Technology*, **1**(3), 231–248 (2002).
7. J. Cheesman, J. Daniels, *UML Components*, Addison-Wesley (2001).
8. P.T. Cox, B. Song, A formal model for component-based software, in: *Proc. of the IEEE Symposium on Visual/Multimedia Approaches to Programming and Software Engineering*, Stresa, Italy (2001), pp. 304–311.
9. S. Moschoyiannis, M.W. Shields, Component-based design: towards guided composition, in: *Proceedings of Application of Concurrency to System Design (ACSD'03)*, Guimaraes, Portugal, IEEE Computer Society (2003), pp. 122–131.

SUMMARY

***V. Giedrimas, A. Lupeikienė. Formalisation of the component specification***

This paper deals with the component specification and formalisation problems. It defines the notion of component model, gives the formalised concept of a component, compound component, and proposes to use hierarchical finite state machines as the formalism to specify component's life cycle.

*Keywords:* component, component model, component specification, specification formalisation.