*Liet. matem. rink,* **45***, spec. nr.*, 2005, 124–128

# Aspektai ir turinių atskyrimas informacinėse sistemose

Albertas ČAPLINSKAS (MII)
el. paštas: alcapl@ktl.mii.lt

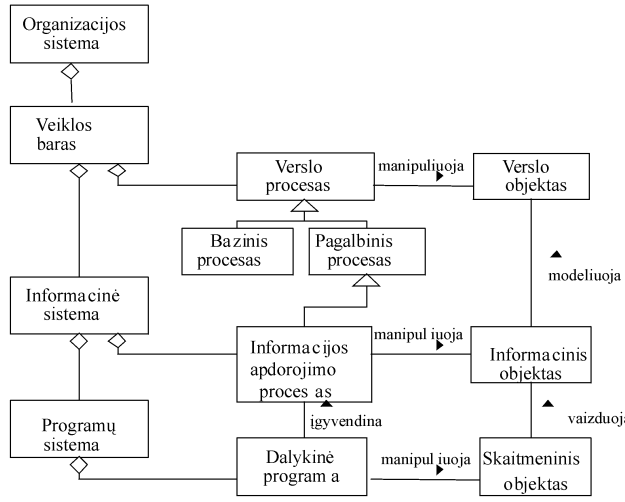
## Įvadas

Aspektinė paradigma [1] – tai nauja programų sistemų (PS) inžinerijos paradigma, nagrinėjanti kaip, kuriant programų sistemas, analizuoti, modeliuoti ir realizuoti persikertančius turinius. Turiniu (angl. concern) vadinama bet kuri griežtai apibrėžta kuriamos PS savybė, persikertančiais turiniais (angl. crosscutting concerns) – tokie turiniai, kurių tradicinėmis programavimo kalbomis negalima realizuoti ištisiniu kodo gabalu, nes jų realizacija yra išbarstoma po skirtingus sistemos komponentus, perpinant ją su tų komponentų kodu. Aspektinis programavimas persikertančių turinių problemą sprendžia įvesdamas specialius modulius (aspektus). Persikertantys turiniai yra projektuojami ir programuojami kaip aspektai. Aspektai programuojami kokia nors aspektine programavimo kalba, pavyzdžiui, AspectJ [2]. Aspekto kodas yra vientisas, jis perpinamas su atitinkamų komponentų kodu tik kompiliavimo metu. Tai daroma automatiškai, tačiau kompiliatoriui reikia nurodyti, kur turi būti terpiamas aspekto kodas. Šitokia metodika tinka PS kurti, bet ja negalima pasinaudoti kuriant organizacijų informacines sistemas (IS), nes tokiose sistemose persikertančius turinius tenka išbarstyti ne tik po atskirus programinius komponentus, bet ir po rankiniu būdu vykdomas verslo bei informacijos apdorojimo procedūras. Šio straipsnio tikslas – pasiūlyti vieną iš galimų šios problemos sprendimo būdų.

## Turinių atskyrimas informacinėse sistemose

Projektuojant organizacijų IS, organizacijų veiklą tenka nagrinėti trimis lygmenimis: verslo procesų (veiklos barų), informacijos apdorojimo procesų ir PS. Informacijos apdorojimo procesų visuma sudaro organizacijos IS. Šie procesai palaiko atitinkamus organizacijos veiklos barus ir patys, savo ruožtu, gali būti didesniu ar mažesniu mastu palaikomi atitinkamų PS. Kitaip tariant, organizacijos IS niekuomet nebūna visiškai kompiuterizuota, dalis informacijos apdorojimo procesų yra kompiuterizuojami tik iš dalies. Kompiuterizuojamų ir rankinių procesų atskyrimas ir harmoninga jų tarpusavio darna yra vieni iš svarbiausiųjų organizacijos IS projektavimo tikslų. Nors praktikoje šis reikalavimas dažnai pažeidžiamas, gerai suprojektuotoje IS rankiniai informacijos apdorojimo procesai turėtų būti projektuojami taip pat kruopščiai, kaip ir kompiuterizuoti procesai. Tai tampa ypač svarbu kuriant IS aspektiškai.

Persikertantys turiniai turi būti išryškinti ir specifikuoti analizuojant organizacijos veiklos barus. Projektuojant IS, jie turi būti nuleisti į informacijos apdorojimo procesų




1 pav. Organizacijos informacinės sistemos projektavimo lygmenys.

lygmenį, po to – į PS lygmenį (1 pav.). Kadangi IS lygmenyje yra modeliuojami ne visi verslo aspektai, tai į tą lygmenį yra nuleidžiami tik tam tikri persikertančių turinių realizavimo fragmentai. Tas pats vyksta ir nuleidžiant persikertančius turinius iš IS lygmens į PS lygmenį. Taigi, pasikeitus persikertantį turinį nusakantiems reikalavimams, atitinkamus pakeitimus tenka daryti ne tik programose, bet ir informacijos apdorojimo bei verslo procedūrose. Neturint unifikuoto visos organizacijos veiklos sistemos struktūrizavimo būdo, tokius pertvarkimus atlikti yra labai sudėtinga.

## Komponentinis požiūris į organizacijos veiklos sistemą

Prielaidas visai organizacijos veiklos sistemai struktūrizuoti unifikuotu būdu sudaro komponentinis požiūris, ypač to požiūrio atmaina, pasiūlyta darbe [3]. Taikant šį požiūrį mūsų nagrinėjamai problemai spręsti, komponentas apibrėžiamas kaip savarankiška organizacijos veiklos sistemos dalis kiek įmanoma mažiau priklausanti nuo kitų tos sistemos dalių. Kitaip tariant, reikalaujama maksimizuoti kiekvieno komponento rišlumą ir minimizuoti jo sankibą su kitais komponentais. Komponentas pats savaime nebūtinai tiesiogiai apibrėžiamas organizacijos veiklos terminais. Organizacijos veiklos požiūriu prasmingi yra tik atitinkami komponentų rinkiniai (ansambliai).

Pareikalausime, kad komponentai tenkintų šiuos papildomus reikalavimus:

- Teiktų tiksliai apibrėžtas paslaugas, kurių pobūdis priklauso nuo komponento vaidmens ir jo atsakomybės srities organizacijos veiklos sistemoje. Apie bet kurį komponentą žinoma tik tai, ką jis moka daryti. Kaip komponentas atlieka jo teikiamas paslaugas, įskaitant ir tai, ar paslaugą realizuoja programa, informacinis procesas, ar verslo procesas, nėra žinoma. Tačiau, jei komponentas pats naudojasi kitų komponentų teikiamomis paslaugomis, kokių komponentų ir kokiomis paslaugomis jis naudojasi turi būti žinoma.

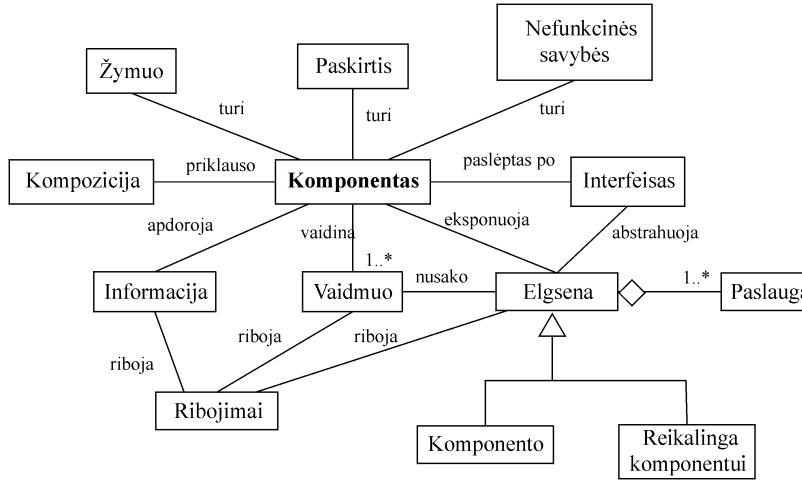
- Būtų išreiškiami paprastesnių komponentų kompozicija (išimtis – baziniai komponentai) ir, kita vertus, patys galėtų dalyvauti kompozicijose, kuriančiose sudėtingesnius komponentus.
- Turėtų išsamiai specifikuotas funkcines ir nefunkcines savybes (komponento teikiamos paslaugos specifikuojamos aprašant jo interfeisą).

Šitaip suprantamo komponento metamodelis pateiktas 2 pav.

Pagal savo tikslinę paskirtį komponentai yra skirstomi į veiklos komponentus, IS komponentus ir PS komponentus. Kiekvienas organizacijos veiklos sistemos lygmuo komponuojamas iš atitinkamos rūšies komponentų (1 pav.). Veiklos komponentai naudojami verslo procesams realizuoti, IS komponentai – informacijos apdorojimo procesams realizuoti ir PS komponentai – informacijos apdorojimo procesus palaikančioms dalykinėms programoms realizuoti.

Veiklos komponentai teikia verslo paslaugas. Jie gali naudotis tiek kitų veiklos komponentų, tiek ir IS komponentų tiekiamomis paslaugomis, jų elgsena nusakoma atitinkamomis verslo taisyklėmis. Paprastai, veiklos komponentas įgyvendina kokią nors verslo lygmens užduotį (angl. use case). Kaip taisyklė, į veiklos komponentą kuriančios kompozicijos sudėtį įeina bent vienas IS komponentas.

IS komponentai teikia informacijos apdorojimo paslaugas. Jie gali naudotis tiek kitų IS komponentų, tiek ir PS (jos traktuojamos kaip komponentai) tiekiamomis paslaugomis. IS komponentų elgsena nusakoma informacijos apdorojimo proceso organizavimo taisyklėmis. Tos taisyklės atspindi verslo taisykles, bet paprastai nustato dar ir tam tikrus papildomus reikalavimus. Dažniausiai, į IS komponentą kuriančios kompozicijos sudėtį įeina bent viena PS. Ji gali būti ir instrumentinio pobūdžio, pavyzdžiui, kokiam nors informacijos procesui realizuoti gali būti panaudota kompiuterinė tekstų redagavimo sistema. IS komponentai nesukuria jokių verslo prasme išbaigtų verčių,




2 pav. Komponento metamodelis.

bet be tokių komponentų verslo paslaugų teikti neįmanoma, nes jie realizuoja tam tikrus verslo užduoties žingsnius, t.y. IS lygmens užduotis.

Programiniai komponentai teikia paslaugas, reikalingas IS komponentams įgyvendinti. Jie naudojasi kitų to pačio arba žemesnio lygmens programinių komponentų tiekiamomis paslaugomis. Programiniai komponentų elgsena nusakoma taisyklėmis, kurios atspindi informacijos apdorojimo taisykles, bet gali numatyti ir kai kuriuos papildomus reikalavimus. Į programinį komponentą kuriančios kompozicijos sudėtį gali įeiti ne tik programiniai komponentai. Pavyzdžiui, operatorius, dialogo režimu gali į PS darbą įterpti kokios nors rankinės procedūros rezultatus. Kai kurie programiniai komponentai gali pilnai realizuoti atitinkamus IS komponentus, t.y. jie gali kurti IS prasme išbaigtas vertes. Tačiau dažniausiai tokie komponentai realizuoja tik tam tikrus kokios nors IS lygmens užduoties žingsnius, t.y. tam tikras PS lygmens užduotis.

Modeliuojant komponentus UML kalba, komponentų rūšį siūloma nusakyti rūšies specifikatoriumi (angl. stereotype).

### Komponentinis požiūris į organizacijos veiklos sistemą

Persikertantys turiniai esti įvairūs. Vienok, organizacijos veiklos sistemos kontekste svarbiausiais yra persikertantys turiniai išplaukiantys iš nefunkcinių veiklos sistemos reikalavimų: verslo operacijų sauga bei patikimumas, verslo transakcijų greitis ir pan. Būtent šitokie persikertantys turiniai „persmelkia“ visą organizacijos veiklos sistemą nuo viršaus iki apačios ir turi išsibarstyti po veiklos, IS ir programinius komponentus. Kita vertus, yra žinoma kaip nefunkcinius reikalavimus operacionalizuoti (t.y. paversti atitinkamomis užduotimis) ir modeliuoti UML priemonėmis. Todėl siūloma persikertančių (nefunkcinių) ir funkcinių turinių perpynimą atlikti komponentų reikalavimų lygmenyje. Siūloma tokia perpynimo strategija:

1) formuluojamos verslo taisyklės;
2) verslo taisyklės transformuojamos į funkcinius ir nefunkcinius organizacijos veiklos sistemos, tarkime, banko sistemos, reikalavimus;
3) organizacijos veiklos sistema dekomponuojama į veiklos komponentus, funkciniai reikalavimai lokalizuojami veiklos komponentuose ir detalizuojami jų terminais;
4) veiklos komponentai dekomponuojami, dekompozicija išplečiama į IS ir PS lygmenis, suformuojama komponentinė organizacijos veiklos sistemos architektūra;
5) funkciniai reikalavimai nuleidžiami iki žemiausiojo lygmens, tapatinami su atitinkamo lygmens užduotimis ir, panaudojant UML, kiekvienas komponentas, nepriklausomai nuo jo pobūdžio, aprašomas atitinkamu užduočių modeliu (modelį sudaro: užduočių diagrama, sekų diagramos, būsenų diagrama, klasių diagrama ir galbūt ansamblių diagramos);
6) nefunkciniai reikalavimai lokalizuojami komponentuose, operacionalizuojami ir perpinami su funkciniais reikalavimais, išreikštais atitinkamo lygmens užduočių modeliais;
7) atitinkamai modifikuojami kiti komponentą aprašančio užduočių modelio elementai, t.y. klasių diagrama, būsenų diagrama ir ansamblių diagramos;
8) taip gauti galutiniai užduočių modeliai panaudojami komponentui realizuoti.

Kas vadinama komponento realizavimu, priklauso nuo komponento rūšies. Veiklos ir IS komponentai realizuojami atitinkamomis instrukcijomis. Jos gaunamos, verbalizuojant sekų diagramas, transformuojant klasių diagramas į terminų žodinėlius ir būsenų diagramas – į instrukcijas, kaip pasinaudoti komponentu. Programiniai komponentai realizuojami programiškai, t.y. programuojant AspectJ kalba.

Persikertantiems turiniams perpinti modelio lygmenyje su funkcinių komponentų turiniu siūloma naudoti tris metodus:

- perklojimą (angl. overlaping): nefunkcinis reikalavimas modifikuoja funkcinį reikalavimą, t.y. UML sekų diagramos yra papildomos atitinkamais plėtiniais;
- perrašymą (angl. overriding): nefunkcinis reikalavimas keičia komponento veikimo būdą, t.y. kai kurie sekų diagramų žingsniai yra perrašomi iš naujo;
- pakavimą (angl. wrapping): funkcionalumas įpakuojamas į nefunkcinį reikalavimą, tampa operacionalizuoto nefunkcinio reikalavimo použdutimi.

## Išvados

Nagrinėjant persikertančių turinių realizavimą tik programų sistemų lygmeniu, yra pažeidžiamas organizacijos veiklos sistemos vientisumo principas, nes ignoruojama ta dalis persikertančių turinių realizacijos, kuri išsibarsto po nekompiuterizuotus verslo ir informacijos apdorojimo procesus. Problema gali būti išspręsta sudarant komponentinę organizacijos sistemos struktūrą ir vienodai traktuojant visus sistemos komponentus, nepriklausomai nuo jų realizavimo būdo. Vienas iš galimų unifikuoto traktavimo būdų yra modeliuoti visus komponentus UML užduočių diagramomis ir perpinti tuos modelius su persikertančių turinių modeliais.

## Literatūra

1. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, Aspect-oriented programming, *ACM Computing Surveys*, **28**(4es), Articles No. 154, December (1996).
2. G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W.G. Griswold, An overview of AspectJ, in: *Proceedings of ECOOP 2001*, *LNCS*, vol. 2072, Springer-Verlag (2001), pp. 327–353.
3. Z. Stojanovic, A. Dahanayake, Components and viewpoints as integrated separations of concerns in system designing, in: *Proceedings of Workshop on Aspect-Oriented Design* (*in conjunction with the 1st International Conference on Aspect-Oriented Software Development*), Enschede, April 22–26, 2002. `http://www2.umassd.edu/swsoc/workshops/aosd2002/papers/11-stojanovic.pdf`

SUMMARY

***A. Čaplinskas. Aspects and concerns separation in IS***

The paper discusses how to proceed crosscutting concerns in enterprise information sistems, which include computer-aided as well as manual information processing procedures. It proposes to consider such systems as component systems including diferent level components: business components, information processing components and software components. All components should be treated in an uniform manner and modelled using appropriate use case models. Using the proposed approach, croscutting concerns are tangled with functional concerns at the use case level. After this, software components are implemented using conventional techniques of aspect-oriented programming and other components in the form of guides and manuals how to perform manual and computer-aided procedures.

*Keywords:* information systems engineering, aspects, concerns separation.