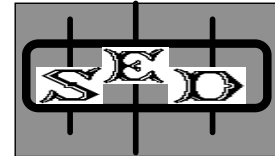




**Matematikos ir  
informatikos institutas**  
L I E T U V A



---

PROGRAMŲ SISTEMŲ INŽINERIJOS SKYRIUS

---

# **KOMPONENTINIS ORGANIZACIJŲ INTEGRUOTŲ INFORMACINIŲ SISTEMŲ KŪRIMAS**

Lina Bagušytė

Gruodis 2007

Techninė ataskaita MII-PSIS-07-

Matematikos ir informatikos institutas, Akademijos g. 4, Vilnius

<http://www.mii.lt>

## **Santrauka**

Techninėje ataskaitoje nagrinėjama komponentinė sistemų kūrimo paradigma, siekiant perkelti komponentinių programų sistemų kūrimo idėjas į organizacijų integruotų informacinių sistemų lygmenį. Pateikiama organizacijos integruotos informacinės sistemos samprata ir problemos, kurias reikia išspręsti norint kurti tokias sistemas. Nagrinėjamos komponento ir jo pagrindinių dalių sampratos, įvairių abstrakcijos lygmenų komponentų ypatumai, jų jungimo būdai. Suformuluoti reikalavimai informacinės sistemos komponentui.

**Raktiniai žodžiai:** programinis komponentas, verslo komponentas, informacinės sistemos komponentas, komponentinė paradigma, kompozicija, reikalavimai

## Turinys

|  |    |
|--|----|
| Santrauka.....   | 2  |
| Įvadas .....   | 4  |
| 1. Organizacijų komponentinių integruotų informacinių sistemų kūrimas: esama situacija ir sprendtinės problemos..... | 5  |
| 2. Komponentinė sistemų kūrimo paradigma .....   | 7  |
| 3. Komponento samprata .....   | 8  |
| 4. Komponentinis programų sistemų kūrimo procesas.....   | 11 |
| 5. Komponentų klasifikavimas .....   | 12 |
| 5.1 Komponentas kaip realizacijos konstrukcija ir kaip architektūrinė abstrakcija.....                               | 12 |
| 5.2 Skirtingų lygmenų komponentai.....   | 14 |
| 5.2.1 Programinio komponento samprata .....  | 14 |
| 5.2.2 Verslo komponento samprata.....  | 21 |
| 5.2.3 Informacinės sistemos komponento samprata .....  | 22 |
| 5.2.4 Sudėtiniai komponentai.....  | 23 |
| 6. Reikalavimai informacinės sistemos komponentui .....  | 23 |
| Išvados .....  | 26 |
| LITERATŪRA .....   | 27 |

## **Ivadas**

Šiuolaikinės įmonės – tai sudėtingos sistemos iš darnią visumą jungiančios įvairias sudėtingas dalis, apimančios informacijos apdorojimo ir programinės įrangos komponentus. Informacinės sistemos (IS), kaip verslo sistemos posistemės, turi atitikti verslo poreikius ir padėti siekti strateginių tikslų, teikdamos reikiamas informacijos apdorojimo paslaugas.

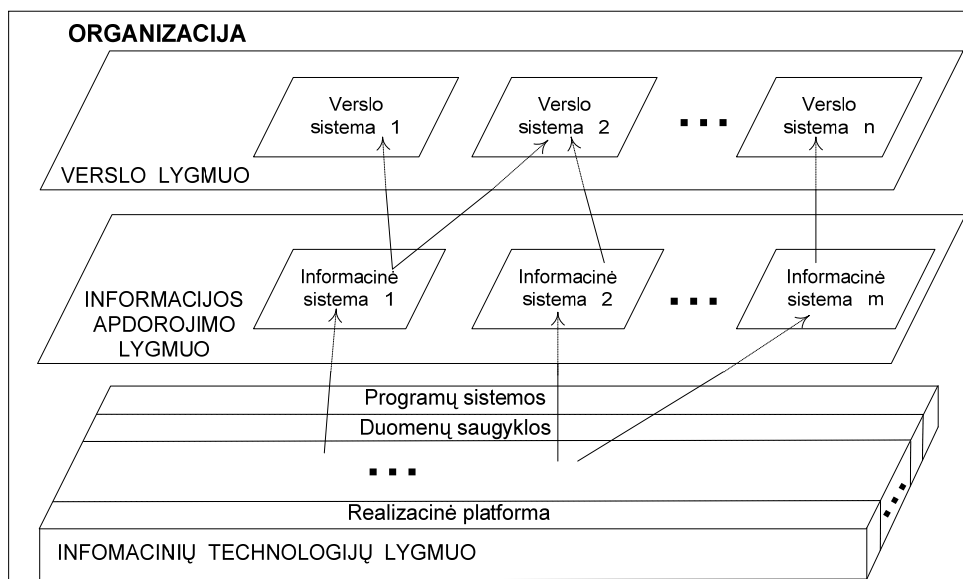
Sparčiai kintant aplinkos reikalavimams reikalinga tokia integruotų informacinių sistemų infrastruktūra ir metodai, kurie leistų nesunkiai modifikuoti egzistuojančias sistemas, integruoti naujas bei užtikrinti visų organizacijoje veikiančių sistemų vientisumą. Kadangi tokios sistemos yra didelės ir sudėtingos, pagrindinės jų kūrėjų problemos – efektyviai tvarkytis su sudėtingumu ir užtikrinti pakankamai greitą bei paprastą sukurtų sistemų modifikavimą bei integravimą. Teorija, kaip kurti integruotas, apjungiančias heterogeninius komponentus informacines sistemas, yra nepakankamai išplėtotas. Akcentuojama [Ly01], kad mažinant atotrūkį tarp verslo poreikių ir IS, reikalingi metodai, kurie leistų kiek įmanoma anksčiau atvaizduoti kompiuterizuojamus procesus iš konkrečių IS komponentų ir nustatyti jų sąveiką. Vienas iš galimų sprendimo būdų yra komponentinės inžinerijos idėjų taikymas aukštesniame, t.y. informacijos apdorojimo, abstrakcijos lygmenyje. Reikalingos priemonės ir metodai, kurie koncepciniame lygmenyje leistų suprojektuoti, kaip turėtų būti jungiami komponentai. Nėra aišku, tiek realizaciniame, tiek koncepciniame lygmenyje, kokius komponentus galima jungti, o kokius ne, nes ne visada komponentai logiškai tinka vienas kitam.

Norint projektuoti ir realizuoti integruotas informacines sistemas, reikia aiškiai apibrėžti, kas yra informacinės sistemos komponentas (ISK), nusakyti jo struktūrą ir leistinas savybes, suformuluoti reikalavimus, kuriuos turi tenkinti bet koks informacinės sistemos komponentas. Šiuo tikslu techninėje ataskaitoje pateikiama organizacijos komponentinės integruotos informacinės sistemos samprata, pagrindinės tokių sistemų kūrimo problemos, apžvelgiamos ir nagrinėjamos įvairių abstrakcijos lygmenų (verslo, informacijos apdorojimo (programinio), informacinių technologijų) komponentų sampratos, komponentų rūšys ir galimos klasifikacijos, komponentų jungimo būdai, komponentinio programų sistemų kūrimo ypatumai.

# 1. Organizacijų komponentinių integruotų informacinių sistemų kūrimas: esama situacija ir spręstinios problemos

Kiekvieną organizaciją sudaro trijuose lygmenyse veikiančios sistemos – verslo sistemos, informacinės sistemos ir programų sistemos (1 pav.). Verslo sistema yra suprantama kaip „integruota veiklų visuma, skirta tam tikriems socialiniams ekonominiams (dažniausiai ilgalaikiams) tikslams siekti, kurianti tam tikrą funkcionalumą, išplaukiantį iš siekiamų tikslų pobūdžio, ir tenkinanti tam tikrus nefunkcinio pobūdžio reikalavimus, užtikrinančius racionalų išteklių naudojimą, patikimumą bei kitas pageidautinas tos sistemos savybes” [CapL01]. Organizacijai siekiant savo tikslų, svarbu, kad visų lygmenų sistemos būtų tinkamai integruotos. Organizacijos veiklos pokyčiai daro įtaką informacinėms sistemoms, kadangi informacinė sistema yra vienas iš verslo sistemos posistemų, sudarytų iš tarpusavyje susijusių informacijos apdorojimo procesų. Taigi, turi būti užtikrinamas adekvatus pokyčių verslo lygmeniu atspindėjimas informacinės sistemos lygmeniu.

Programų sistema suprantama kaip integruota programų, failų, duomenų bei žinių bazių ir galbūt kitų komponentų visuma, skirta tam tikros klasės uždaviniams spręsti arba tam tikriems įrenginiams ar procesams valdyti [Cap96]. Taigi jos yra vienos svarbesnių informacinių sistemų sudėtinė dalis. Programų sistemos kompiuterizuoja informacijos apdorojimo procesus, todėl turi modeliuoti tuos pačius objektus, kuriais naudojasi informacinė sistema. Kitaip tariant, informacinės sistemos objektai perkeliama į programų sistemas, kur yra modeliuojami programų sistemos objektais. Kaip ir verslo sistemų atveju, šis „perkėlimas“ nėra tiesioginis – vienas informacinės sistemos objektas gali būti modeliuojamas keliais programų sistemos objektais ir, atvirkščiai, vienas programų sistemos objektas gali modeliuoti kelis informacinės sistemos objektus. Be to, programų sistemos turi savus, vidinius objektus, neturinčius atitikmenų informacinėse sistemose [CapL01].



1 pav. Organizacija kaip trijų lygmenų sistema

Apibendrinant pabrėžtina, kad visų trijų lygmenų sistemos turi sudaryti vieningą visumą. Vadinasi, turi būti kuriamos integruotos organizacijų informacinės sistemos (IOIS).

Integruotos organizacijų informacinės sistemos yra didelės ir sudėtingos sistemos. Pagrindinės jų kūrėjų problemos – efektyviai tvarkytis su sudėtingumu ir užtikrinti pakankamai greitą bei paprastą sukurtų sistemų modifikavimą. Kitos problemos su kuriomis susiduriama yra heterogeninių komponentų integravimas ir įmonių prisitaikymas prie aplinkoje vykstančių pokyčių. Keičiasi ne tik technologijos, bet ir įmonių organizacinės struktūros – nuo vertikalų, izoliuotų padalinių, su aiškiai apibrėžtomis atsakomybėmis, iki horizontalių, verslo procesais grindžiamų struktūrų ir išskirstytų verslo paslaugų. Siekiant spręsti heterogeniškumo, tarpusavio sąveikos (interoperabilumo) ir nuolat kintančių reikalavimų problemas, reikalinga infrastruktūra, kuri leistų kurti ir integruoti silpnai sukibusius, pakartotinai panaudojamus, turinčius aiškiai apibrėžtus, suderintus su atitinkamais standartais, interfeisus, artefaktus.

Vienas galimų sprendimo būdų – komponentinės inžinerijos metodų taikymas aukštesniame, informacijos apdorojimo, lygmenyje. Komponentinės paradigma yra brandi ir plačiai taikoma kuriant du IOIS struktūrinius elementus: techninę įrangą ir programų sistemas. Komponentinės technologijos Microsoft ActiveX/DCOM/COM, Corba komponentai, Java Beans, vėliau COM+/.NET, Enterprise Java Beans (EJB) vis plačiau naudojamos projektuojant ir įgyvendinant sudėtingas programų sistemas. Reikia paminėti, kad komponentai faktiškai yra dabartinis programų sistemų (ypač išskirstytų) realizavimo standartas. Tačiau kuriant aukštesniojo abstrakcijos lygmens (informacinių, verslo) sistemas, komponentinis požiūris nėra plačiai naudojamas.

Kuriant komponentines informacines sistemas, projektavimo eigoje turi būti pasirenkama tinkama architektūra, aprašanti:

- informacinės sistemos komponentus;
- leistinas informacinių sistemų komponentų jungtis (t. y. kaip jie gali būti jungiami vienas su kitu);
- darnos reikalavimus, ribojančius IOIS veikimą, kad būtų išvengta nepageidautinų efektų.

Šiuo metu naudojamos informacinių sistemų kūrimo metodikos, viena vertus, paprastai skirtos informacinėms sistemoms siaurąja prasme projektuoti; antra vertus, nėra netgi ir tokių informacinių sistemų kūrimo darnios ir išsamios teorijos (IS siaurąja prasme suprantama kaip programų, duomenų bazių ir aparatūros sistema). Tačiau organizacijų informacinės infrastruktūros kūrimo kontekste IS nagrinėjamos ir šiek tiek platesne prasme. Pavyzdžiui, darbe [Cum02] pateikiamas organizacijos IS infrastruktūros integravimo modelis, apimantis kompiuterių tinklo, techninės įrangos, kompiuterinio pašto, sistemos valdymo, apsaugos modelio, portalų, žinių valdymų, ir duomenų saugyklų integravimą.

Komponentinių informacinių sistemų teorija dar tik pradedama kurti. Pagrindinės IOIS kūrimo problemos, kurios turi būti išspręstos, yra šios [BL06].

- elementariųjų ir sudėtinių IOIS komponentų nustatymas;
- IOIS komponentų rūšių aibės nustatymas;
- komponentinių informacinių sistemų architektūrų ištyrimas, apimantis komponentų sąveiką, jų jungimo būdus;
- komponentinių informacinių sistemų teorijos sukūrimas, apimantis konceptų ir apibrėžčių visumos sudarymą, IOIS analizės ir projektavimo metodų sukūrimą.

## **2. Komponentinė sistemų kūrimo paradigma**

Sistemų kūrimas iš komponentų leidžia suprojektuoti nesunkiai modifikuojamas sistemas, pakartotinai naudojant jau turimus artefaktus. Kitaip tariant, tikslinga komponuoti sistemas iš maksimaliai nepriklausomų tarpusavyje sudėtinių dalių. Toks sistemų kūrimo būdas yra plačiai naudojamas įvairiose inžinerijos šakose.

Programų sistemų kūrimas iš nepriklausomų sudėtinių dalių taip pat nėra nauja idėja. 1968 metais įvykusioje tarptautinėje NATO konferencijoje, kurioje pirmą kartą oficialiai prabilta apie programavimo (programų sistemų) kūrimo krizę, problemai spręsti buvo pasiūlyta naudoti komponentus. 1972 metais D. L. Parnas [Par72] pasiūlė, kaip dekomponuoti sistemas į modulius. Pagrindiniais privalumais įvardijo greitesnį programų kūrimą, kadangi kiekvieną modulį gali kurti kitas programuotojas; modulių nepriklausomumą – galimybę pakeisti vieną modulį, nekeičiant kitų; geresnį programų projektavimą - dėl aiškesnės pačios sistemos struktūros. Modulis turėjo atlikti priskirtą funkciją, jo vidinę konstrukciją ir projektinius sprendimus slėpė modulio interfeisas. Pagal [Par72] modularizavimo efektyvumas priklauso nuo modularizavimo kriterijų pasirinkimo.

Objektinė paradigma leido verslo atstovams ir programinės įrangos kūrėjams vienodai suprasti nagrinėjamą dalykinę sritį, naudojant objektus. Pagal [Boo86] objektai susieja funkcionalumą ir duomenis, atvaizduoja realaus pasaulio esybes ir yra pagrindinės sudedamosios programų sistemos dalys, kurias galima pakartotinai panaudoti. Tačiau praktika parodė, kad sudėtinga valdyti didelius projektus naudojant objektines kūrimo technikas [Szy02, Whi02]. Taikant objektinę metodiką gaunamas didelis kiekis smulkaus granuliarumo susijusių objektų/klasių. Kadangi objektai stipriai sukibę, sukuriamos sudėtingos sistemos, turinčios monolitines struktūras, kurias sunku keisti, plėsti ar adaptuoti [SW99].

Komponentinė programų sistemų kūrimo paradigma pasiūlyta kaip vienas iš galimų šios problemos sprendimo būdų [BW98, Szy02, CL02]. Autoriai teigia, kad sistemos kūrimą sudaro inkapsuliuotų, pakeičiamų, sąveikaujančių tarpusavyje, su aiškiai apibrėžtu funkcionalumu ir paslėpta realizacija elementų pasirinkimas, konfigūravimas, adaptavimas, surinkimas.

Komponentinės technologijos Microsoft ActiveX/DCOM/COM, Corba komponentai, Java Beans, vėliau COM+/.NET, Enterprise Java Beans (EJB) vis plačiau naudojamos projektuojant ir įgyvendinant sudėtingas programų sistemas. Pastebėsime, kad komponentai faktiškai yra dabartinis programų sistemų (ypač išskirstytų) realizavimo standartas. Tačiau kuriant aukštesniojo abstrakcijos lygmens (informacinių, verslo) sistemas, komponentinis požiūris nėra plačiai naudojamas.

Paslaugomis grindžiamoje architektūroje pakylama į aukštesnį abstrakcijos lygmenį. Sudėtinės dalys yra realios verslo veiklos, inkapsuliuojančios verslo teikiamas paslaugas išorei. Domenas yra nagrinėjamas kaip paslaugų, teikiančių verslo lygmens funkcionalumą, visuma. [Sto05] teigia, kad komponentų kūrimas ir paslaugų (angl. *web service*) technologija padeda realizuoti paslaugomis grindžiamą architektūrą.

Tačiau technologinių sprendimų nepakanka verslo ir informacinių sistemų reikalavimams įgyvendinti. Iškyla pavojus, kad kompiuterizuota sistema neatitiks verslo poreikių. [HS00, CL02] teigiama, kad negalima pilnai pasinaudoti komponentinės sistemų kūrimo paradigmos privalumais, kadangi nėra tinkamų metodikų, nusakančių, kaip tai daryti. Taigi, reikalingi komponentiniai analizės ir projektavimo būdai.

### 3. Komponento samprata

Šiuo metu komponentinės programų sistemų kūrimo paradigmos „pasaulyje“ komponentas suprantamas labai įvairiai, tie patys terminai žymi skirtingas sąvokas. Apibrėždami komponentus, autoriai paprastai akcentuoja svarbiausius jų nagrinėjamajame kontekste komponentų ypatumus, tokius kaip autonomiškumas, granuliariškumas, naudojimo kontekstas.

Pagal [Szy02] programinis komponentas yra kompozicijos vienetas, turintis kontrakto principu specifikuotus interfeisus ir aiškias konteksto priklausomybes. Programinis komponentas gali būti naudojamas nepriklausomai nuo kitų ir kuriamas ar įsigyjamas išorėje.

Detaliai aptarsime komponento savybes (remdamiesi darbe [Vol03] pateiktais paaiškinimais):

- „kompozicijos vienetas“ - turima omenyje, kad komponentas yra jungiamas su kitais komponentais. Komponentinė sistema surenkama iš tarpusavyje galinčių sąveikauti komponentų aibės.
- „kontrakto principu specifikuoti interfeisai“. Komponentas turi turėti vieną ar daugiau interfeisų, kad galėtų būti komponuojamas į sistemą. Interfeisas nusako kontraktą tarp komponento ir jo aplinkos. Kitaip tariant, interfeisas aiškiai nusako komponento teikiamas paslaugas, t.y. atsakomybę.
- „aiškios konteksto priklausomybės“. Programinė įranga priklauso nuo aplinkos, kurioje vykdoma, t.y. sistemos resursų, kuriais gali pasinaudoti. Norint komponuoti sistemas reikia,

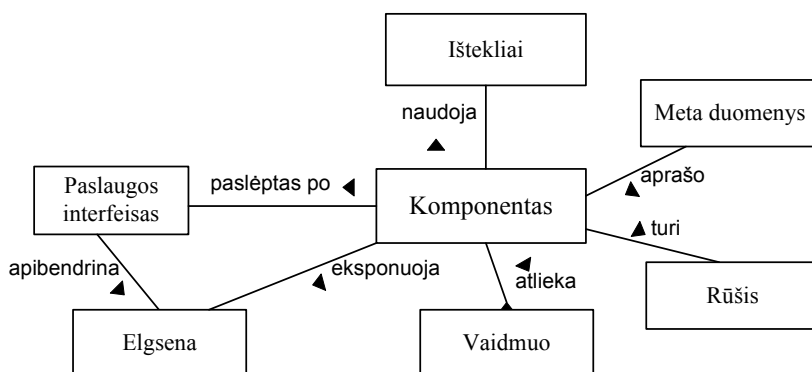
kad aiškiai būtų specifiukuota, su kokiais kitais komponentais galima bendradarbiauti ir kokiomis jų teikiamomis paslaugomis pasinaudoti.

- „gali būti naudojamas nepriklausomai nuo kitų“. Komponentas yra savarankiškas (angl. *self-contained*) komponavimo vienetas. Komponento realizacijos keitimas neįtakoja kitų komponentų. Kitaip tariant, komponento specifikacija yra atskirta nuo realizacijos.
- „kuriamas ar įsigyjamas išorėje“. Pabrėžiamas pačių komponentų ir komponentinių sistemų kūrimas. Akcentuojama, kad komponentai yra pakartotinai panaudojami elementai, kuriuos galima įsigyti išorėje ir panaudoti sistemoms kurti.

Aptartos savybės nusako esminius komponento bruožus. Ši komponento apibrėžtis yra bendra, todėl gali reikšti skirtingus dalykus: posistemius, vykdomus modulius, JavaBeans, ActiveX valdiklius, .NET, Enterprise JavaBeans, COM+, CORBA komponentus ir pan.

Darbo [Vol03] autorius, parodydamas termino „komponentas“ nevienareikšmiškumą, naudoja komponentų aibės sutvarkymą ir aprašo leistinas jų savybes.

Komponentui nusakyti vartojamos paslaugų interfeiso, meta duomenų, rūšies, išteklių ir vaidmens sąvokos (2 pav.). Reikia, kad būtų specifiukuotos komponento teikiamos paslaugos, o tai paprastai specifiukuojama operacijų ir jų parametrų terminais. Be to, specifiukuojant komponentą, turi būti nusakyti jam reikalingi ištekliai.



2 pav. Komponento metamodelis

Paslaugos interfeisas atskiria komponento išorę ir vidų, atsakydamas į klausimą kokias naudingas paslaugas gali teikti komponentas tam tikroje aplinkoje. Detali ir tiksliai apibrėžta komponento interfeiso specifikacija leidžia sudaryti kontraktą tarp komponentų - paslaugų vartotojų ir komponentų-paslaugų teikėjų, nežinant vidinės realizacijos [StoD03]. Kontraktas turi apibrėžti funkcinius aspektus, tokius kaip paslaugos ir joms taikomi ribojimai, bei nefunkcinius reikalavimus. [Bac00] teigia, kad komponentas per interfeisus įgyvendina įsipareigojimus, dar vadinamus kontraktu. Be to, minėti kontraktiniai įsipareigojimai leidžia užtikrinti, kad nepriklausomai sukurti komponentai galės sąveikauti tarpusavyje numatytais būdais, arba bus panaudojami standartinėse kūrimo ir vykdymo aplinkose. [Bac00] autoriai nagrinėja kontraktą dviem skirtingomis prasmėmis:

vienu atveju kontrakto objektas yra pats komponentas, kitu – sąveika. Kitaip tariant, skiriamas komponento kontraktas ir sąveikos kontraktas. Komponentų kontraktai yra būtini, norint nusakyti komponento teikiamas paslaugas ir jų savybes. Sąveikos kontraktas specifikuoja interfeisų tipų abipusius išsipareigojimus. Toks kontraktas suprantamas kaip projektavimo specifikacija, kurią turi „užpildyti“ komponentai. Kadangi komponentas gali realizuoti keletos interfeisų tipus, tai jis gali atlikti keletą vaidmenų. Iš to seka, kad kiekvienas sąveikos kontraktas aprašo skirtingų vaidmenų, esančių sistemoje, sąveiką.

Kontraktinis interfeisas yra viena iš pagrindinių komponento dalių, kadangi tik per jį paslaugos vartotojas sužino komponento galimybes.

Pagal [BRJ99] interfeisas specifikuoja iš išorės matomas klasės ar komponento operacijas, tačiau nepateikia vidinės struktūros aprašo. Kiekvienas interfeisas specifikuoja tam tikrą konkrečios klasės elgsenos dalį. Interfeisai neturi realizacijos. Jiems nusakyti nenaudojami atributai, būsenos ar asociacijos ryšiai. Jie turi tik operacijas ir gali dalyvauti apibendrinimo ryšyje. UML 1.3 metamodelyje interfeisas, kaip ir klasė, yra klasifikatorius (angl. *classifier*). Ryšys tarp klasės ir jos interfeiso, kaip ir ryšys tarp komponento ir jo interfeiso, vadinamas realizacija.

[Szy02] komponento interfeisą apibrėžia kaip prieigos (angl. *access*) taško specifikaciją. Per šiuos taškus klientai pasiekia komponento teikiamas paslaugas. Interfeisas nesiūlo operacijų realizacijos – pateikiama operacijų su aprašymais aibė. Teikiamų paslaugų realizacijos ir specifikacijos atskyrimas leidžia pakeisti komponento realizaciją, nekeičiant sistemos interfeiso, ir tokiu būdu pagerinti sistemos našumą.

Bendruoju atveju, skiriami dviejų rūšių interfeisai. Teikiamas interfeisas vaizduoja paslaugas ir operacijas, kurias teikia komponentas aplinkai, atlikdamas tam tikrą vaidmenį. Reikiamas interfeisas specifikuoja tas paslaugas ir operacijas, kurių reikia iš aplinkos, kad atliktų priskirtą vaidmenį.

Pažymėtina, kad interfeisų aprašymo kalbos yra sintaksinio lygmens. Kitaip tariant, jomis specifikuojama signatūros dalis, kurioje aprašomos komponento teikiamos operacijos. [HeCo01] autoriai skiria tokias sudėtingas interfeiso dalis: semantiškai susijusių operacijų pavadinimai, jų parametrai, galimi parametrų tipai. [CL02] teigia, kad šis interfeiso aprašas yra nepakankamas aiškiai ir tiksliai nusakyti komponento elgseną. Tiksliai komponento interfeiso specifikacija reikalinga jungiant komponentus į bendrą problemos sprendinį, norint pakeisti komponentą kitu, suderintu komponentu, ieškoti komponentų kataloge pagal specifikaciją, pakartotinai panaudoti komponentą kitoje aplinkoje.

Nagrinėdami interfeisų specifikuojimo problemą [Bac00] skiria abstrakčius, t.y. nepriklausančius nuo realizacijos, interfeisus ir susijusius su realizacija (angl. *bound*), kurie gali pateikti savybes, neesančias abstrakčiuose interfeisuose.

[Vol03] autorius nurodo komponentui būdingus vaidmenis: esybė, procesas, paslauga. Komponentas atliekantis esybės vaidmenį turi pastovią būseną. Komponentas vaizduojantis paslaugą paprastai būsenos neturi (pvz., komponentas skaičiuojantis pridėtinės vertės mokestį ar atliekantis kitus matematinius skaičiavimus, ar pakuotojas (angl. *wrapper*) fiziniam įrenginiui ar liktinei (angl. *legacy*) sistemai. Proceso komponentas inkapsuliuoja procesą, pvz., pirkinių krepšelio užpildymas ar sudėtinis darbų srautas (angl. *workflow*). Nors šios rūšies komponentas ir turi būseną, tačiau ji nepastovi.

Komponentai turėtų teikti metainformaciją, t.y. juos pačius aprašančius duomenis. Išteklių specifikacija sudaro meta duomenų dalį. Meta duomenys galimi veikimo ir/ar kūrimo metu.

#### **4. Komponentinis programų sistemų kūrimo procesas**

Komponentus galima kurti nepriklausomai nuo sistemos, nors komponentinių sistemų gyvavimo ciklas apima ir komponentų, ir pačių sistemų kūrimą [AS98, Br00, CL02, RW01]. Tačiau šie procesai turi bendrų sąveikos taškų, t.y. komponentų reikalavimai ir sistemos reikalavimai įtakoja vienas kitą, todėl projektuojant būtina atsižvelgti į jų ypatumus. Sprendžiant šią problemą ir norint išvengti nesuderinamumo tarp šių procesų, siūloma naudoti standartus. Šiuo metu standartai nėra apibrėžti, nes procesai ir technologijos, naudojami komponentinių sistemų kūrimo, nuolat kinta. Komponentai bendrauja su aplinka tik per interfeisus, todėl svarbiausia veikla yra suprojektuoti tinkamus interfeisus.

Komponentiniame programų sistemų kūrimo procese išskiriamas pačių komponentų kūrimas ir komponentinių sistemų kūrimas.

Kuriant komponentines sistemas, pagrindinės veiklos yra tinkamų komponentų identifikavimas ir komponentų, tenkinančių visos sistemos funkcionalumo reikalavimus, parinkimas. Sistemų kūrimas iš komponentų panašus į nekomponentinių sistemų kūrimą. Esminis skirtumas – egzistuojančių komponentų panaudojimas. Todėl svarbu rasti tinkamas sudėtines dalis, kurios tenkintų sistemos funkcionalumą. Svarbi valdymo veikla.

Išskiriamos tokios komponentinio programų sistemų kūrimo gyvavimo ciklo stadijos:

1) *Reikalavimų analizę ir specifikavimą* (apima ir komponentų specifikacijų kūrimą) sudaro trys etapai: sistemos reikalavimų ir jos ribų nustatymas; sistemos architektūros apibrėžimas, kuri reikalinga, kad būtų galima nusakyti komponentų sąveiką; komponentų reikalavimų apibrėžimas, kad būtų galima atrinkti arba sukurti reikiamus komponentus.

Pagrindinės problemos, su kuriomis susiduriama šioje stadijoje: sudėtinga nusakyti reikalavimus komponentams, turint sistemos reikalavimus; neplanuotas funkcionalumas, t.y. gaunami nesuderinti komponentų rinkiniai.

2) *Komponentų parinkimas ir įvertinimas.* Reikalavimų specifikacija turi būti rengiama atsižvelgiant į tai, kad komponentas gali būti įsigyjamas išorėje.

3) *Sistemos projektavimas* prasideda nuo sistemos specifikavimo ir architektūros apibrėžimo. Pradinė architektūra priklauso nuo sistemos reikalavimų ir komponento modelio, nusakančio galimus sąveikos būdus tarp komponentų, ir infrastruktūros, pasirinkimo. Projektavimas glaudžiai susijęs su parinkimo procesu – nagrinėjamos įvairios galimos parinktų komponentų kombinacijos.

4) *Sistemos realizacija* specifikuoja komponentų sujungimą ir komponentų pritaikymą.

5) *Sistemos integravimo* metu atliekama parinktų komponentų kompozicija, atsižvelgiant į sistemos architektūrą, naudojimo ir sąveikos standartus, realizuojama tam tikroje infrastruktūroje.

Komponentinėje programų sistemų inžinerijoje išskiriamos dvi kryptys: komponentų kūrimas pakartotiniam panaudojimui (angl. *for reuse*) ir kūrimas panaudojant turimus komponentus (angl. *with reuse*). Pirmuoju atveju, programų sistemos gali būti kuriamos naudojant įprastus programų sistemų inžinerijos metodus, pabrėžiant komponentų standartus. Kūrimo panaudojant turimus komponentus kryptyje svarbiausios yra programinių komponentų paieškos ir išrinkimo veiklos.

## **5. Komponentų klasifikavimas**

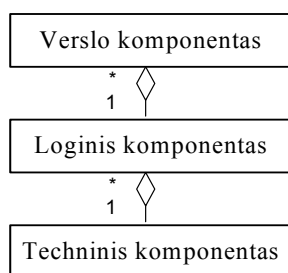
Komponentai klasifikuojami pagal įvairius požymius. 5.1 skyrelyje bus nagrinėjami loginiai ir techniniai (realizaciniai) komponentai, jų tarpusavio ryšiai. 5.2 skyrelyje aptariami skirtingų lygmenų komponentai bei jų ypatybės.

### **5.1 Komponentas kaip realizacijos konstrukcija ir kaip architektūrinė abstrakcija**

Pagal [Bac00] programiniai komponentai apjungia dvi skirtingas perspektyvas: komponentus, kaip realizacijos konstrukciją, ir architektūrinę abstrakciją.

Komponentai - realizacijos konstrukcijos suprantami kaip rinkoje įsigyjami pagaminti komponentai (angl. *commercial-off-the-shelf (COTS)*). Tačiau autorių [Bac00] nuomone, COTS produktų realizuojamas funkcionalumas ir koordinavimas (kaip komponentas sąveikauja su išore) yra unikalus kiekvienam produktui. Tuo tarpu architektūriniai komponentai išreiškia projektavimo taisyklės, nusakančias visiems komponentams būdingą standartinį koordinavimo modelį. Minėtos projektavimo taisyklės vaizduojamos komponento modeliu arba standartų ir susitarimų aibe, kurią turi tenkinti komponentai.

[Vol03] pabrėžiama, kad būtina skirti loginius ir techninius komponentus (3 pav.). Loginis komponentas suprantamas kaip paketas, turintis tam tikrą funkcionalumą ir sąlygojantis sistemos sudėtingumą. Pažymima, kad loginiai komponentai dar skirstomi į dalykinės srities, duomenų ir vartotojo komponentus – dalykinės srities komponentuose nusakoma verslo logika; duomenų komponentai užtikrina prieigą prie duomenų; vartotojo komponentai įgyvendina vartotojo interfeiso funkcionalumą. Techniniai komponentai yra programiniai elementai, iš kurių konstruojamos dalykinės programos, veikiančios tik tam tikroje aplinkoje (angl. *container*). Taikomas turinių atskyrimo principas – dalykinės programos aspektai realizuojami skirtinguose techniniuose artefaktuose. Techniniai komponentai diegiami kliento programose arba, jei yra kelių lygmenų sistema, serveryje.



3 pav. Ryšys tarp komponento rūšių

Tarp techninių ir loginių komponentų yra agregavimo ryšys. Kitaip tariant, techniniai komponentai yra sudėtinės loginio komponento dalys. Darbe [Vol03] vartojama verslo komponento sąvoka. Verslo komponentas apibrėžiamas kaip loginių komponentų (duomenų, dalykinės srities ir vartotojo) agregatas. Taigi, sudėtingesni komponentai konstruojami iš paprastesnių. Be to, išskiriami skirtingi sistemos lygmenys – techninis, loginis ir verslo. Jiems įgyvendinti turi būti naudojami skirtingų rūšių komponentai.

Kadangi bet kurios įmonės veiklą galima traktuoti kaip sistemą, susidedančią iš verslo ir jas palaikančių informacinių sistemų, realizuojamų naudojant tinkamas technologijas, skirsime verslo, informacinės sistemos ir techninius, t.y. programų sistemų, komponentus.

Siekiant nustatyti, kokius reikalavimus turi tenkinti informacinės sistemos komponentas, detaliau nagrinėsime skirtingų rūšių komponentus ir jų ypatumus.

## 5.2 Skirtingų lygmenų komponentai

### 5.2.1 Programinio komponento samprata

Kalbant apie programinį komponentą dažniausiai sakoma (pvz., [Bac02]), kad tai yra fizinė sistemos dalis, kuri atitinka ir realizuoja tam tikrą kontraktais specifišką interfeisų aibę. Kitais žodžiais tariant, komponentu laikomas tam tikrą funkcionalumą turintis binarinio kodo paketas. Nuo komponentinės paradigmos atsiradimo iki UML 1.3 [UML00] versijos sukūrimo tai buvo vyraujanti komponento samprata. Be to, komponentinių programų sistemų kūrimo pradininkai labiausiai domėjosi, kaip komponentas atitinka naudojamos kūrimo metodikos ir infrastruktūros standartus.

UML 1.3 versijoje komponentu vadinama fizinė keičiama sistemos dalis, kuri realizuoja tam tikrą interfeisų aibę. Komponentas yra fizinis sistemos realizacijos fragmentas, apimantis programinės įrangos kodą (išeities, binarinį ar vykdomąjį) ar komandinius failus. Jis gali pats realizuoti interfeisų aibę, kurią nusako paslaugos, esančios komponento viduje. Paslaugos apibūdina komponento egzemplioriaus elgseną kitiems komponento egzemplioriams. Komponentas turi atributus, operacijas ir realizuoja interfeisus

Nors UML 1.3 [UML00] metamodelyje komponentai ir klasės yra panašūs (realizuoja interfeisų aibę, gali turėti egzempliorius, dalyvauti priklausomybės, apibendrinimo, asociacijos ryšiuose), tačiau reikia paminėti keletą esminių skirtumų. Klasės vaizduoja logines abstrakcijas, o komponentai yra fizinės sistemos dalys. Kitais tariant, komponentus galima realizuoti sistemos mazguose. Komponentas yra fizinė loginio modelio elementų, tokių kaip klasės, realizacija. Klasė turi atributus ir operacijas, o komponentai – tik operacijas, pasiekiamas per interfeisus. Komponentų diagramoje pateikiamos priklausomybės tarp programinių komponentų, kartu ir išeities kodo, binarinio kodo ar vykdomųjų komponentų. Modeliuojant verslą, programiniai komponentai papildomi verslo procedūromis ir dokumentais. Programinės įrangos modulį siūloma vaizduoti kaip komponento stereotipą. Dalis komponentų egzistuoja kompiliavimo metu, dalis – vykdymo metu. Komponentų diagrama gali būti tik tipų lygmenyje, t.y. egzempliorių lygmenyje negalima sudaryti.

UML 2.0 [UML05] pateikta daug apibrėžčių ir modeliavimo technikų, specifikuojančių komponentą, tačiau nėra nuoseklių nurodymų, kaip jomis pasinaudoti kūrimo procese. Komponentas vaizduojamas ir kaip loginis (verslo komponentas, proceso komponentas), ir kaip fizinis (EJB, COM+, .NET komponentai), kartu su komponentus realizuojančiais artefaktais ir mazgais, kuriuose vykdomi. Kitaip tariant, komponentai vaizduojami ne tik kaip realizacinio lygmens, bet ir projektavimo lygmens artefaktai, kurie kartu su klasėmis naudojami kuriant loginę sistemos architektūrą.

Komponentas, kaip vykdomas sistemos elementas, vaizduojamas specializuota klase, turinčia išorinę specifikaciją, išreikštą per interfeisus, ir vidinę realizaciją. Taip pat yra apibrėžiami komponentų konektoriai (angl. *connectors*), kurie sujungia (anlg. *wire*) komponentus pagal suderinamumą. Komponentas yra keičiama sistemos dalis, kuri gali būti pakeista projektavimo arba vykdymo metu kitu komponentu, siūlančiu lygiavertį funkcionalumą, remiantis jo interfeisų suderinamumu. Sistemą galima papildyti naujais komponentais, papildančiais jos funkcionalumą.

[UML05] komponentas vaizduojamas kaip klasės potipis, kuris turi atributus ir operacijas, bei gali dalyvauti asociacijos ir apibendrinimo ryšiuose. Jis taip pat gali turėti vidinę struktūrą ir „nuosavą“ portų aibę, formalizuojančią sąveikos taškus. Teikiamas interfeisas realizuojamas komponento arba vieno iš realizuojančių klasifikatorių, arba yra teikiamo komponento porto tipas. Reikiamas interfeisas projektuojamas naudojant naudojimo priklausomybę (angl. *usage dependency*) arba yra reikalaujamo porto tipas. Komponentas papildomas, kad būtų galima apibrėžti loginių sudėtinių komponentų (angl. *packaging*) grupavimo aspektus. Tokių sudėtinį komponentą gali sudaryti užduotys ir priklausomybės, paketai, komponentai ir kiti artefaktai. Komponentas formaliai specifikuoja paslaugų kontraktą, kurį teikia klientams ir/arba reikalauja iš kitų sistemos komponentų ar paslaugų.

Reikiamuose/teikiamuose komponento interfeisuose numatomos struktūrinės savybės (atributai ir asociacijos) ir dinaminės savybės (operacijos ir įvykiai). Komponentas realizuoja teikiamą interfeisą tiesiogiai arba per jį realizuojančius klasifikatorius. Reikiami/teikiami interfeisai gali būti jungiami per portus. Tokiu atveju, nusakomos reikiamų/teikiamų interfeisų aibės, į kurias paprastai kreipiamasi vykdymo metu.

[UML05] komponentas nagrinėjamas „juodosios dėžės“ ir „baltosios dėžės“ požiūriu. Komponento išorėje (komponentas kaip „juodoji dėžė“) matomos savybės ir operacijos. Gali būti, kad pvz., būsenos mašinos protokolas prijungiamas prie interfeiso, porto ar paties komponento, išvardinant komponento išorėje esančių operacijų iškvietimų seką. Kiti elementai, turintys funkcionalumą, gali būti jungiami su interfeisais ar konektoriais, sudarant bendradarbiavimo kontraktą (specifikuojant užduotis, veiklas, sąveiką). Tokiu atveju komponentų sujungimas į sistemą, t.y. sudėtinio komponento sudarymas, struktūriškai apibrėžiamas naudojant priklausomybes tarp komponento interfeisų (vaizduojama struktūrinėse diagramose). Detalesnė struktūrinio bendradarbiavimo specifikacija sudaroma naudojant sudėtinių struktūrų dalis ir konektorius. Tokiu būdu aprašomas komponentų bendradarbiavimas egzempliorių lygmenyje arba jų atliekama rolė.

Komponento viduje („baltosios“ dėžės požiūriu) yra iš išorės nematomos (angl. *private*) savybės ir jas realizuojantys klasifikatoriai. Kitaip tariant, matoma išorėje stebimos elgsenos realizacija. Atvaizdavimas tarp vidaus ir išorės atliekamas naudojant priklausomybes (struktūrinėse

diagramose) arba per delegavimo konektorius vidinėms dalims (sudėtinių struktūrų diagramose). „Baltosios dėžės“ atveju išvardinami visi realizuojantys klasifikatoriai, t.y. sudėtinės dalys, konektorai ir realizuojantys artefaktai. Vidiniai klasifikatoriai, realizuojantys komponento elgseną, siejami priklausomybės ryšiais.

Konektoariaus apibrėžtis apima ribojimus, susijusius su interfeisu ir jų notacija. Yra delegavimo ir surinkimo (angl. *assembly*) konektoariai. Delegavimo konektorius susieja išorėje matomą komponento kontraktą (kaip specifiukuota portuose) su vidine realizacija. Jis vaizduoja signalų (operacijų iškvietimų ir įvykių) perdavimą: kai signalas ateina į portą, turintį delegavimo konektoarių į kitą dalį arba portą, jis nukreipiamas apdorojimui. Delegavimo konektorius reiškia, kad elgsena, esanti komponento egzemplioriuje, nerealizuojama paties komponento, o perduodama kitam, turinčiam „suderinamas“ galimybes. Tai gali būti kitas komponentas arba klasė.

Delegavimo konektoariai naudojami modeliuoti hierarchinės elgsenos dekompoziciją, kur komponento teikiamos paslaugos realizuojamos komponentu, esančiu žemesniame, t.y. detalesniame lygmenyje. Tokiu atveju perduodama žinutė arba signalas tarp sujungtų portų. Reikia pažymėti, kad signalas, perduodamas į detalesnį abstrakcijos lygmenį, realizuojamas ne visose aplinkose. Portas deleguoja funkcionalumą pavaldžių komponentų portų aibei, teikiančiai tokį patį funkcionalumą. Vykdomo metu signalai perduodami į reikiamą portą. Jei tą patį signalą gali apdoroti keletas portų, tai signalas perduodamas jiems visiems.

Surinkimo konektorius susieja komponentus, iš kurių vienas reikalauja paslaugos, o kitas ją teikia, t.y. nuo reikalaujamo interfeiso arba porto į teikiančią interfeisą arba portą. Surinkimo konektoariaus atveju, signalas, atsirandantis reikiamame porte, perduodamas į teikiamą portą. Jei yra daug konektoarių iš vieno reikiamo interfeiso ar porto į daug teikiamų komponentų, tada komponento egzempliorius, kuris perims signalo apdorojimą, parenkamas vykdomo metu.

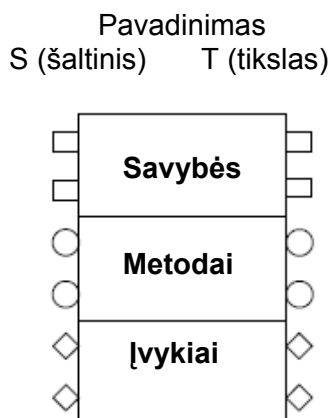
Interfeisų suderinamumas tarp sujungtų reikiamų/teikiamų portų, leidžia pakeisti komponentą kitu, kuris siūlo ne mažesnę paslaugų aibę. Sistemos praplėtimo atveju, kai pridedami komponentai, turintys naujo funkcionalumo, surinkimo konektoariai sujungia jų portus su sistemos portais.

[Wan02] pabrėžiama programinių komponentų infrastruktūros svarbą. Autoriai teigia, kad įvairių komponentinių technologijų infrastruktūros skiriasi, todėl skirtingai apibrėžiamas ir pats komponentas. Komponento infrastruktūrą supranta, kaip komponentų konstravimo ir jų valdymo struktūrą (angl. *framework*) bei priemones (angl. *facilities*). Ją sudaro trys modeliai: komponento modelis, jungimo (angl. *connection*) modelis ir realizacinis (angl. *deployment*) modelis. Taip apibrėžta komponento infrastruktūra literatūroje dar vadinama komponento technologija arba komponento architektūra.

Komponento modelis apibrėžia, kas yra komponentas, kaip jį sukurti tam tikroje infrastruktūroje. Kiekviena komponento infrastruktūra turi pakartotinai panaudojamą komponentų biblioteką, sudarytą iš sudedamųjų dalių, atitinkančių komponento modelį. Jungimo modelis apibrėžia konektorių aibę ir priemones reikalingas komponentų surinkimui. Kitais žodžiais tariant, šis modelis nustato kaip sukurti programą ar didesnę komponentą iš egzistuojančių komponentų. Realizacinis modelis charakterizuoja kaip patalpinti komponentus į darbinę aplinką.

Komponentinėse infrastruktūrose (JavaBeans, EJB, .NET, CORBA, OSGi ir Web services) išskiria visus tris modelius. Modelių formalizavimui naudoja algebrą.

[Wan02] komponentas yra apibrėžiamas kaip savybių, metodų ir įvykių aibė. Savybės inkapsuliuoja komponento būsenas ir atributus. Metodai nusako komponento elgseną ir paslaugas. Įvykiai aprašo veiksmus, kuriuos komponentas gali inicijuoti. Interfeisas suprantamas kaip savybių, metodų ir įvykių aibių Dekarto sandaugos poaibis. Programinį komponentą vaizduoja grafiniu būdu (4 pav. ).



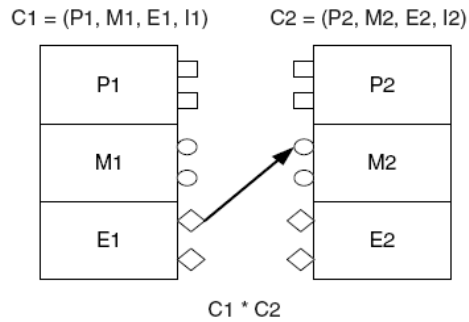
4 pav. Komponento samprata pagal [Wan02]

Komponentas formaliai apibrėžiamas kaip ketvertas  $C = (P, M, E, I)$ , kur P – savybės, M – metodai, E – įvykiai, I – interfeisai.

Pateikta komponento apibrėžtis yra programinių komponentų abstrakcija. Komponentų infrastruktūrose vartojamos „savos“ komponentų apibrėžtys.

[Wan02] autoriai komponentų jungimui naudoja algebros operatorius (+, -, \*, / ) nepriklausomai nuo komponento infrastruktūros:

1. Agregavimas arba sudėtis. Komponentai jungiami be tiesioginių sąveikų.
2. Atimtis . Komponentai atskiriami.
3. Įvykių valdymas arba daugyba (5 pav.). Įvykis iš šaltinio komponento susiejamas su tikslo komponento metodu. Kai reikia išreikšti, kurie įvykiai inicijuoja bendravimą, užrašoma taip  $C1 * e C2$ , kur e yra inicijuojantysis įvykis „sukabinimo“ (angl. hookup) operacijai.



5 pav. Komponentų daugyba

4. Įvykio susiejimas su savybe arba dalyba. Įvykis iš šaltinio komponento susiejamas su tikslo komponento savybe. Šis operatorius panašus į daugybą, tačiau skiriasi jo poveikis tikslo komponentui.
5. Įvykio susiejimas su įvykiu. Šaltinio įvykis susiejamas su tikslo įvykiu. Šis operatorius panašus į daugybą, išskyrus poveikį tikslo komponentui.
6. Interfeiso modifikavimas  $(+p, - p, +m, - m, +e, - e)$ . Šie operatoriai naudojami interfeiso modifikavimui.

Komponentų sąveikai aprašyti naudojamos dvi pagrindinės asociacijos – pranešimų perdavimas (angl. *message passing*) ir įvykių valdymas (angl. *event driving*). Įvykio asociacija yra suprantama, kaip sąryšis tarp šaltinio komponentų ir tikslo komponentų. Tikslo komponentai atitinkamai apdoroja šaltinio komponento įvykius. Realizaciniame komponento infrastruktūros modelyje nustatomas komponentų parengimo vykdymui procesas, apimantis diegimą ir konfigūraciją. Programinė įranga suskaidoma į komponentus, kurie projektuojami ir realizuojami atsižvelgiant į jų tikslus. Tai leidžia naudotojams ir kūrėjams perskirstyti sistemos aspektus į bendras sritis, kurios aiškiai identifikuojamos, specifikuojamos, projektuojamos, realizuojamos ir gali būti panaudotos kitose dalykinėse programose.

Pagal [CD01] komponentinės sistemos kuriamos naudojant principą „skaldyk ir valdyk“. Pagrindinis skirtumas nuo struktūrinių metodų – naudojamas objektas, kuris apjungia funkcijas ir reikalingus duomenis į vieną vienetą. Autoriai pabrėžia komponento pakeičiamumą. Jų nuomone, pakartotinis panaudojimas – antraeilis dalykas, nes visos sistemos valdymas yra daug svarbesnis už atskirų komponentų pakartotinį panaudojimą skirtingose sistemose. Komponentus apibrėžia kaip programinės įrangos vienetus, struktūrizuotus atsižvelgiant į šiuos principus:

- duomenų ir funkcijų unifikavimą, t.y. programinės įrangos objektą sudaro duomenų reikšmės (arba būsenos) ir funkcijos, apdorojančios tuos duomenis;
- inkapsuliaciją;
- identiškumą, nepriklausantį nuo būsenos, t.y. objektas turi tapati.

Komponentas teikia paslaugas ir reikalauja paslaugų iš kitų komponentų. [CD01] autoriai teigia, kad komponentas nėra paslauga, nors iš komponentų gali būti kuriamos paslaugų (angl. *service-based*) architektūros. Išskiria du kontraktų tipus:

- naudojimo (angl. *usage*) – tarp komponento objekto interfeiso ir jo klientų;
- realizacijos – tarp komponento specifikacijos ir realizacijos.

Interfeiso specifikaciją sudaro:

- teikiamų interfeiso operacijų sąrašas, kartu su signatūra ir apibrėžimais;
- informacinis modelis.

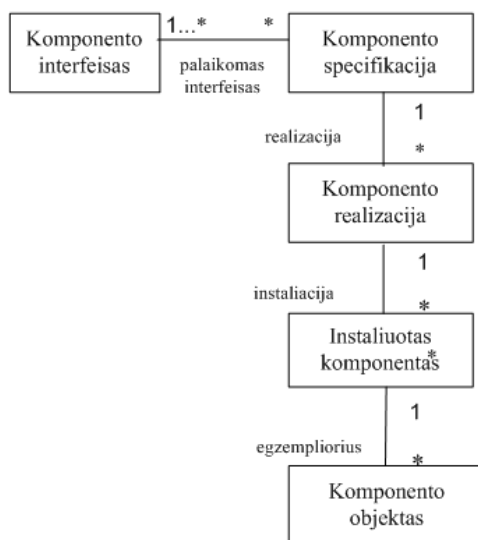
Kiekviena operacija laikoma kontraktu. Operacija turi prieš-sąlygas ir po-sąlygas. Realizacijos kontraktą apima komponento specifikacija.

Komponentai gali būti panaudojami, jei jie atitinka tam tikrus infrastruktūros standartus. Kitais žodžiais tariant, komponento bet kur įjungti neįmanoma – turi būti suderinamumas su infrastruktūra.

Darbe [CD01] plėtojama objekto specifikacijos sąvoka. Pabrėžiama abipusės priklausomybės svarba tarp komponentų, o ne tarp jų veikimo būdo. Komponentai klasifikuojami į penkias formas (6 pav.): komponento specifikacija; komponento interfeisas; komponento realizacija; instaliuotas komponentas; komponentas objektas.

Pagal [CD01] aiškiai atskirta komponento specifikacija ir jo realizacija yra esminė komponento charakteristika. Komponento specifikacija nusako, ką reikia sukurti ir kokios sudedamosios dalys egzistuos vykdymo metu. Komponento specifikacija, t.y. ką komponentas daro, išskaidoma į interfeisus. Komponento specifikacijos architektūra nusako komponentų realizacijos ir surinkimo ribojimus komponuojant sistemą. Interfeisas nenusako, kaip realizacija turi sąveikauti su kitais komponentais. Interfeiso specifikacija aprašo sąveikas, bet nenusako kaip jos vyksta. Tai daro kitos komponento specifikacijos dalys.

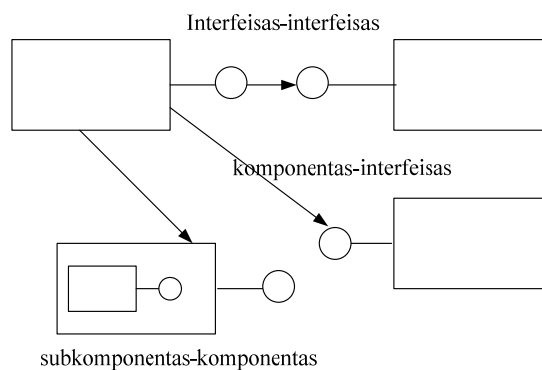
Kompiuteriuose instaliuota komponento realizacija vadinama instaliuotu komponentu. Vykdymo metu taip pat reikia atsižvelgti į komponento būklę ir turinį. Iš instaliuoto komponento sukuriama egzemplioriai vadinami komponento objektais (angl. *component – objects*), kurie turi tapstį ir duomenis. Instaliuotas komponentas gali turėti vieną arba kelis komponentus objektus.



6 pav. Komponento formos pagal [CD01]

[CD01] elgsena suprantama kaip interfeisų aibė, kurią gali pasiūlyti komponentas objektas. Komponentas nėra objektas, nors vykdymo metu atsirandantys komponentai objektai turi daug objektų savybių, tačiau jie egzistuoja komponento standarto aplinkoje. Komponentinę architektūrą apibrėžia kaip taikomųjų programų lygmens komponentų aibė, susieta struktūriniais ir elgsenos ryšiais.

Struktūriniai ryšiai suprantami kaip asociacijos ir paveldėjimo ryšiai tarp komponento specifikacijos ir interfeiso, kompozicijos ryšiai tarp komponentų. Elgsenos priklausomybės egzistuoja tarp komponentų, tarp komponentų ir interfeisų, tarp interfeisų (7 pav.).



7 pav. Elgsenos priklausomybės pagal [CD01]

Pagal [Kru03] komponentas yra beveik savarankiška, keičiama sistemos dalis, atliekanti konkrečią funkciją tam tikroje architektūroje. Komponentas atitinka ir realizuoja interfeisų aibę. Gyvavimo ciklo metu išskiria tokias komponentų rūšis:

- *vykdomieji* (angl. *runtime*) komponentai. Jiems priskiriami elementai, kurie gali būti instaliuojami ir vykdomi tam tikroje aplinkoje. Pavyzdžiui, vykdomieji failai, procesai ir dinamiškai sudaromos bibliotekos (angl. *dynamic link libraries – DLL*).

- *Projektavimo* komponentai (sistemos kūrimo organizavimo požiūriu) yra realizacinės (angl. *implementation*) posistemės, turinčios stiprų vidinį rišlumą ir silpną išorinę sankibą, bei pakartotinai panaudojamos kitų kūrėjų. Ne visada galima atvaizduoti projektavimo komponentą vienu vykdomuoju komponentu ir atvirkščiai.
- *Verslo* komponentai suprantami kaip sukibusios vykdomųjų arba projektavimo komponentų aibės, kurios teikia verslo lygmens funkcionalumą.

[Mes07] skiria techninius-programinius komponentus. Autorius teigia, kad nereikėtų skirstyti komponentų į techninius arba programinius, kadangi sistema reikalauja abiejų – t.y. techninė įranga reikalinga vykdyti programinį komponentą. Kadangi techninės įrangos kainos ženkliai sumažėjo, tampa įprasta, kad techninis elementas skiriamas vienam tikslui, nenaudojamas bendrai. Paties komponento viduje gali būti sujungti atskiri infrastruktūros ir programų elementai. Kaip pavyzdį mini paslaugas (angl. *web service*), kurių interfeise nėra ribos tarp programų ir infrastruktūros, tačiau gali atsirasti moduliarizuojant vidinę realizaciją.

Pagal [Mes07] aukščiausiam abstrakcijos lygmenyje nagrinėjami socialiniai-technologiniai komponentai, kurių interfeisai yra su žmonėmis ir įmonėmis. Kitaip tariant, nagrinėjamos socialinės sistemos. Tokio komponento realizacija apima socialinius ir technologinius procesus, jų konfigūraciją ir plėtimą.

### **5.2.2 Verslo komponento samprata**

[HS00] pakyla į verslo sistemos lygmenį. Tačiau verslo sistemos kontekste nagrinėja programinius komponentus. Autorių nuomone, informacinėje sistemoje programiniu komponentu turi būti realizuojama kuri nors verslo sistemos esybė (pvz., daiktas, verslo procesas). Darbo autoriai skiria įvairaus granuliariškumo komponentus – programinį komponentą, verslo komponentą, verslo komponentų sistemą. Verslo komponentų sistemos modelyje aprašomi šios sistemos funkcionalumą nusakantys verslo komponentai. Verslo komponentas yra programinių komponentų kompozicija. Vadinasi, vis dėlto liekama programų sistemų lygmenyje. Informacinių sistemų programinė įranga gali būti sukomponuota iš rinkoje išigytų pagamintų komponentų. COTS produktai yra projektuojami taip, kad juos būtų galima nesudėtingai integruoti į organizacijoje egzistuojančias programų sistemas.

[Sto05] komponentinių sistemų kūrimo gyvavimo ciklo metu išskiria įvairias komponentų rūšis. Aukščiausiam lygmenyje yra įmonės (angl. *enterprise*) komponentas, galintis egzistuoti kaip savarankiška sistema. Įmonės komponentas teikia verslo procesų automatizavimo paslaugas arba gali būti naudojamas kelių įmonių verslo sistemų integravimui. Realizuojamas konstruojant iš žemesnio lygmens komponentų arba kaip monolitinė struktūra.

Verslo komponentas teikia paslaugas, turinčias aiškiai nusakytą, išmatuojamą naudą verslo sistemai. Kitaip tariant, verslo paslaugos nusako funkcionalumą, reikalingą konkrečiam verslo tikslui įgyvendinti. Verslo komponentas teikiamas paslaugas susieja su verslo esybėmis, verslo taisyklių aibe. Realizuojamas konstruojant iš žemesnio lygmens komponentų arba kaip monolitinė struktūra. [Sto05] apibrėždamas programinius komponentus, išskiria du tipus: komponentus-esybes ir komponentus-procesus. Komponentas-esybė, kuri priskiria turintiems pasyvią atsakomybę, inkapsuliuoja informaciją apie verslo ar dalykinės programos esybę. Komponentais-procesais vadina aktyvius komponentus, kurie teikia žemesnio lygmens veiklas arba su dalykine programa susijusius skaičiavimus. Komponentai-esybės pasiekiami tik per komponentus-procesus, kurie tarsi juos inkapsuliuoja.

Programų komponentai egzistuoja tik vienoje dalykinėje programoje. Jų pavyzdžiai: objektinės klasės projektavimo lygmenyje ir šių klasių objektai egzempliorių lygmenyje. [Sto05] autorius neišskiria informacinės sistemos komponentų – nuo dalykinių programų, programų sistemų pereina prie verslo sistemų.

### 5.2.3 Informacinės sistemos komponento samprata

Darbu nagrinėjančių informacinės sistemos komponentus yra tik keletas. [TLA03] autoriai aprašo duomenų bazes kaip svarbiausią informacinių sistemų struktūrinį elementą. Todėl informacinės sistemos komponentą apibrėžia kaip trejetą  $ISC = \langle SS, DS, IR \rangle$ . Jų nuomone, informacinės sistemos komponentas yra koncepcinis objektas, kurį sudarant turi būti nusakyta:

- statinė sritis SS, aprašanti informacinės sistemos statines savybes, ir modeliuojama UML klasių diagrama;
- dinaminė sritis DS, t.y. modeliuojamos dinaminės informacinės sistemos savybės;
- darnos taisyklių sritis IR, t.y. nusakomi ribojimai, užtikrinantys korektišką sistemos veikimą.

Darbe [Din05] informacinės sistemos komponentas nagrinėjamas koordinuojančiųjų (angl. *coordination – ready*) informacinių sistemų kūrimo kontekste. Autorius teigia, kad norint veiksmingai valdyti informacinių sistemų kūrimą, svarbu tinkamai modularizuoti informacinius išteklius koncepciniu lygmeniu, todėl informacinės sistemos komponentą apibrėžia kaip pakartotinai panaudojamą, turintį tam tikrą funkcionalumą informacinės sistemos artefaktą. Informacinės sistemos komponentas turi visas savybes, kad galėtų funkcionuoti kaip atskira informacinė sistema. Kitaip tariant, informacinės sistemos komponentai yra savarankiški, nepriklausomi artefaktai. Tačiau informacinę sistemą gali sudaryti ir keletas informacinės sistemos komponentų, t. y. komponentai bendradarbiauja įgyvendindami organizacijos tikslus. Specifikuojant informacinės sistemos komponentą analizės metu, aprašomos klasės, jų atributai,

būsenos, metodai, įvykiai, darnos taisyklės. Pastebėsime, kad šiame darbe pateikta IS komponento samprata labiau primena programinį, o ne bet kurios rūšies IS komponentą.

#### **5.2.4 Sudėtiniai komponentai**

Kadangi komponentas yra suprantamas kaip komponavimo vienetas, tai reiškia, kad pagrindinis komponentų panaudojimo tikslas yra jų jungimas su kitais komponentais [Koz99, MMY01]. Atsižvelgiant į tai, skiriami elementarieji komponentai (struktūriniai primityvai, kurių negalima dekomponuoti į paprastesnius) ir sudėtiniai komponentai, kurie konstruojami kuriant sudėtingas sistemas [Sto05]. Autorius taip pat pabrėžia bendradarbiavimo (angl. *collaboration*) ir koordinavimo veiklų svarbą jungiant komponentus. Sudėtinio komponento savybės priklauso nuo sudėtinių dalių ypatumų ir jų jungimo būdo. Taigi kompozicija nusako koordinuotą bendradarbiavimą tarp komponentų kuriant aukštesnio lygmens komponento elgseną.

Pagal [Bac00] komponento modelis nusako komponentų rūšis ir galimus sąveikos šablonus, interfeisų rūšis. Išskiria dvi esybių rūšis, komponentus ir karkasus, kurios naudojamos komponavimo metu, ir pagrindines sąveikos klases:

- komponentas-komponentas kompozicijoje sąveikauja komponentai. Sąveikos rezultatas yra taikomosios programos lygmens funkcionalumas, atitinkamai specifikuojamas taikomųjų programų lygmens kontraktais;
- karkasas-komponentas sąveika leidžia valdyti komponentų išteklius, todėl ją specifikuojantys kontraktai yra sistemos lygmens;
- karkasas – karkasas sąveika suteikia galimybę jungti komponentus, esančius heterogeniniuose karkasuose per tarpusavio sąveikos (angl. *interoperation*) kontraktus.

[Bac00] pastebi, kad komponentai turi būti karkasuose, norint juos komponuoti ir vykdyti. Naudojimo (angl. *deployment*) kontraktas aprašo komponento realizuojamą interfeisą, kad karkasas galėtų valdyti reikiamus išteklius.

## **6. Reikalavimai informacinės sistemos komponentui**

Nagrinėtoms komponentų rūšims yra bendra, kad komponentas gali būti komponuojamas su kitais ir sudaryti sudėtingas struktūras. Kadangi egzistuoja skirtingų rūšių komponentai, atitinkamai reikalingos skirtingų rūšių kompozicijos. Be to, reikalingi skirtingi kontraktų ir interfeisų tipai tarp to paties lygmens komponentų ir tarp skirtingų lygmenų. Sudėtinių komponentų elgsena priklauso nuo sudedamųjų dalių ir jų jungimo būdo. Pagal [Kru03] kompozicijos sąvoka rekursyvi – sistema žemesniame abstrakcijos lygmenyje gali būti aukštesnio abstrakcijos lygmens komponentas.

|                                   |  |
|-----------------------------------|--|
| Cheesman ir Daniels [CD01]        | Realizacija, specifikacija, instaliuotas komponentas, komponentas objektas |
| Kruchten [RUP03]                  | Vykdomasis komponentas, projektavimo komponentas, verslo komponentas       |
| Völter [Vol03]                    | Techninis komponentas, loginis komponentas, verslo komponentas             |
| Verslo komponentų požiūris [HS00] | Verslo komponentų sistema, verslo komponentas, programinis komponentas     |

1 lentelė Komponentų granuliarumas ir rūšys

Galima daryti išvadą, kad nedaug dėmesio skiriama įvairaus granuliarumo komponentams, kurie galėtų būti panaudoti skirtingose sistemų kūrimo gyvavimo ciklo stadijose apibrėžti. To reikia norint turėti aiškų ryšį nuo reikalavimų iki realizacijos bei aiškiai apibrėžtas sudėtines dalis, kurias galima integruoti ir gauti didesnio funkcionalumo komponentus. UML 2.0 yra mechanizmai, leidžiantys projektuoti komponento vidų iš smulkesnių sudėtinių dalių, tačiau specifinės komponentų rūšys nėra išskiriamos.

|                                   | Komponentų identifikavimas                         | Komponentinė analizė ir projektavimas  | Komponentinė realizacija               |
|-----------------------------------|--|--|--|
| Cheesman ir Daniels [CD01]        | Iš dalykinės srities esybių ir užduočių            | Objektinės – UML modeliavimo technikos   | Nepateikta                             |
| Kruchten [RUP03]                  | Susijusių programinių objektų apjungimas į paketus | Neatliekama  | Naudojamos komponentinės technologijos |
| Verslo komponentų požiūris [HS00] | Iš verslo esybių ir procesų                        | Apibrėžiamos skirtingos komponentų rūšys, naudojamos skirtinguose architektūros sluoksniuose | Naudojamos komponentinės technologijos |

2 lentelė Komponentų gyvavimo ciklo stadijos

Darbe [BM06] pabrėžiama pokyčių verslo procesuose įtaka programų sistemoms. Autorių nuomone, būtent informacinės sistemos funkcijos yra jungiamoji grandis tarp verslo procesų ir IT sistemų, aprašanti sistemą funkcionalumo terminais. Informacinių sistemų funkcijos yra pakartotinai panaudojamos verslo procesuose, kai reikia dokumentuoti verslo funkcijai reikalingą informacinių technologijų sistemos funkcionalumą.

Apibendrinant informacinės sistemos komponentas turi tenkinti šiuos reikalavimus, kylančius iš komponentinės paradigmos ir bendrosios sistemų teorijos:

- ISK turi teikti informacijos apdorojimo, komunikavimo, saugojimo, informacijos srautų valdymo paslaugas (t.y. reikalingi funkciniai ir nefunkciniai ISK) verslo procesui, siekiant verslo tikslų.
- Sudėtiniai komponentai išreiškiami paprastesnių komponentų kompozicija. Sudėtinio komponento savybės priklauso nuo jo sudedamųjų dalių charakteristikų ir jų jungimo būdo.
- Turi būti galimybė pakeisti arba atnaujinti sudėtinio ISK sudedamąsias dalis, nekeičiant kitų. Iš to seka, kad ISK išorinė elgsena turi būti atskirta nuo realizacijos

(turinio), kuri yra inkapsuluota ISK. Kitaip tariant, informacinės sistemos komponentas realizuojamas kaip savarankiškas artefaktas.

- Informacinės sistemos komponento paslaugos nusakomos per reikiamus/teikiamus interfeisus.
- Reikalingas ISK metaaprašas, kad būtų galima juos jungti tarpusavyje bei įkelti į egzistuojančią IS infrastruktūrą.
- Informacinės sistemos komponentai turi tenkinti tam tikrus standartus, sudarančius galimybę juos jungti tarpusavyje ir integruoti į egzistuojančią IS.

## Išvados

1. Komponentinių sistemų kūrimo idėjos IOIS lygmenyje nėra plačiai naudojamos. Metodai ir technikos dar tik pradedami kurti. Komponentinės paradigma yra brandi ir plačiai taikoma kuriant du IOIS struktūrinius elementus: techninę įrangą ir programų sistemas.
2. Nors autoriai skiria įvairaus granuliarumo komponentus, tačiau nėra aiškiai apibrėžiama, kuriose sistemų kūrimo gyvavimo ciklo stadijose jie galėtų būti naudojami, koks ryšys tarp jų.
3. Informacinės sistemos komponentas turi teikti informacijos apdorojimo, komunikavimo, saugojimo, informacijos srautų valdymo paslaugas (t.y. reikalingi funkciniai ir nefunkciniai ISK) verslo procesui, siekiant verslo tikslų, per standartizuotus interfeisus. Jis turi būti realizuojamas kaip savarankiškas artefaktas, todėl ISK išorinė elgsena turi būti atskirta nuo realizacijos. Reikalingas ISK metaaprašas, kad būtų galima ISK jungti tarpusavyje bei įkelti į egzistuojančią IS infrastruktūrą.
4. Sudėtiniai informacinės sistemos komponentai išreiškiami paprastesnių komponentų kompozicija. Sudėtinio ISK savybės priklauso nuo jo sudedamųjų dalių charakteristikų ir jų jungimo būdo.

## LITERATŪRA

- [AS98] P. Allen, S. Frost. *Component-based development for Enterprise Systems Applying the SELECT Perspective*. Cambridge University Press (1998)
- [Bac00] 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)
- [BL06] L. Bagušytė, A. Lupeikienė (2006). Organizacijų integruotų informacinių sistemų komponentinio kūrimo problemos. *Informacinės technologijos' 2006, Konferencijos pranešimų medžiaga*. Kaunas, Technologija (2006)
- [BM06] P.Büch, D. Maurer. From business process design to Enterprise Architecture. *ARIS Expert Paper* (2006)
- [Boo86] G. Booch. Object-oriented development. In: *IEEE Transactions on Software Engineering* 12, p. 211-221 (1986)
- [BRJ99] G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley (1999)
- [BW98] A.W.Brown, K.C. Walnau. The Current State of Component-Based Software Engineering. In: *IEEE Software* 15 (5), p. 37-46 (1998)
- [Br00] A. W. Brown. *Large – Scale Component-based development*. Prentice Hall (2000)
- [CapL01] A. Čaplinskis, A. Lupeikienė. Sistemų inžinerijos intelektualizavimo problemos. *Informacinės technologijos' 2001, Konferencijos pranešimų medžiaga*. Kaunas, Technologija, p. 200-205 (2001)
- [Cap96] A. Čaplinskis. *Programų sistemų inžinerijos pagrindai, I dalis*. Matematikos ir informatikos institutas, Vilnius (1996)
- [CD01] J. Cheesman, J. Daniels. *UML Components*. Addison-Wesley (2001)
- [CL02] I.Crnkovic, M.Larsson. *Building reliable component-based software systems*. Artech-House (2002)
- [Cum02] F. A. Cummins. *Enterprise Integration. An architecture for Enterprise Application and Systems Integration*. John Wiley & Sons Inc. p. 58 – 87 (2002)
- [Din05] T. Le Dinh. *Information System Upon Information Systems: a Conceptual Framework*. Doctoral dissertation, University of Geneva (2005)
- [HeCo01] G. T. Heineman, W.T. Councill. *Component – based Software Engineering: Putting the Pieces Together*. Addison-Wesley: Upper Saddle River, NHJ (2001)
- [HS00] P. Herzum, O. Sims. *Business Component Factory: a Comprehensive Overview of Business Component Development for the Enterprise*. John Wiley & Sons, (2000)
- [Koz99] W. Kozaczynski. Composite nature of component. In: *1999 International Workshop on Component-Based Software Engineering*. <http://www.sei.cmu.edu/cbs/icse99/papers> (1999)
- [Kru03] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley (2003)
- [LY01] L. Liu, E. Yu. From Requirements to Architectural Design – Using Goals and Scenarios. In: *Proceedings of the First International Workshop From Software Requirements to Architectures (STRAW'01)* (2001).
- [MMY01] H. Mili, A. Mili, S. Yacoub. *Reuse-Based Software engineering*. John Wiley & Sons. 2001

- [Mes07] D. G. Messerschmitt. Rethinking Components: from Hardware and Software to Systems. To appear in *IEEE proceedings* (2007)
- [Par72] D.L Parnas. On the criteria to be used in decomposing systems into modules. In: *Communication of the ACM*. 15 (12). p. 1053-1058 (1972)
- [RW01] D. Robey, R. Welke. Traditional, Iterative, and Component-Based Development: A Social Analysis of Software Development Paradigms. In: *Information Technology and Management 2*, Kluwer Academic Publishers, Netherlands, p. 53–70 (2001)
- [Sto05] Z. Stojanovic. *A Method for Component-Based and Service-Oriented Software Systems Engineering*. Doctoral Dissertation, Delft University of Technology, Netherlands (2005)
- [StoD03] Z. Stojanovic, A. Dahanayke. Components and Viewpoints as Integrated Separations of Concerns in System Designing. In: *Workshop on Aspect-Oriented Design (in conjunction with the 1st International Conference on Aspect-Oriented Software Development)* (2002)
- [SW99] D.F.D'Souza, A.C. Wills. *Objects, Components and Frameworks with UML. The Catalysis Approach* (1999)
- [Szy02] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley (2002)
- [TLA03] M. Turki, N. Léonard, Arni-Bloch. From hyperclasses to IS components. MATIS Geneva Team, University of Geneva, Switzerland. In: *Proceedings of CE'2003*, (2003)
- [UML00] OMG Unified Modeling Language Specification Version 1.3, (2000)  
<http://www.omg.org/docs/ad/99-06-08.pdf>
- [UML05] OMG Unified Modeling Language Specification Version 2.0, (2005)
- [Vol03] M. Völter. A taxonomy for components. *Journal of Object Technology*, vol. 2, no. 4, p. 119-125 (2003)
- [Wan02] J. A. Wang. Algebra for components. In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, Vol. 5, *Computer Science I*, N. Callaos, T. Leng, and B. Sanchez (eds), International Institute of Informatics and Systemics, p. 213–218 (2002)
- [Whi02] K. Whitehead. *Component-based development: principles and planning for business systems*. Addison Wesley (2002)