Liet. matem. rink, **46***, spec. nr.*, 2006, 97–102

Reprezentatyvių pavyzdžių specifikavimo kalbų funkcionalumui vertinti konstravimas

Jelena GASPEROVIČ (MII)
el. paštas: j.gasperovic@algoritmusistemos.lt

Reziumė. Straipsnyje nagrinėjamas reprezentatyvių pavyzdžių, skirtų specifikavimo kalbų funkcionalumui vertinti, konstravimas. Trumpai aptariami pagrindiniai reprezentatyvių pavyzdžių konstravimo principai, aprašyta jų konstravimo metodikos esmė, pateiktas metodikos taikymo pavyzdys. Pagrindinis nagrinėjimų tikslas – pasiūlyti metodiką, kurią panaudojus būtų galima sukonstruoti reprezentatyvius pavyzdžius ir naudoti juos nustatant specifikavimo kalbos pakankamumą programų sistemos reikalavimams specifikuoti.

Raktiniai žodžiai: specifikavimo kalba, funkcionalumas, kokybės charakteristika, kokybės vertinimas, reprezentatyvus pavyzdys.

Įvadas

Paprastai kokybė vertinama nustatant ar produktas ar paslauga tenkina tam tikrus apibrėžtus reikalavimus. Kokybės vertinimas atliekamas praktiškai visose veiklos srityse, tame tarpe programų sistemų inžinerijoje, kur yra vertinama naudojamų priemonių, pavyzdžiui, specifikavimo kalbų kokybė. Nepaisant didelės kokybės svarbos, iki šiol nėra pasiūlyta bendrosios kokybės vertinimo teorijos, neaišku koks turi būti kalbos kokybės charakteristikų matas, ir kaip turi būti vertinamos kalbos kokybės charakteristikos. Savo ankstesniuose darbuose [1], [2] specifikavimo kalbų vidinę kokybę atskyrėme nuo panaudojimo kokybės. Panaudojimo kokybė nusako kalbos tinkamumą konkrečiam projektui, o vidinė kokybė nusako santykinį kalbos „gerumą“ lyginant ją su kitomis kalbomis arba, kitaip tariant, tos kalbos savybių išsamumo laipsnį. Atlikus išsamią svarbiausiųjų specifikavimo kalbų (UML, Z, VDL ir Alloy) analizę bei atsižvelgiant į kokybės vertinimo patirtį gretimose srityse (programavimo kalbų, koncepcinių modelių kokybė ir kt.), buvo nustatyta [1], kad specifikavimo kalbų vidinė kokybė aprašoma taksonomine hierarchija, kurios aukščiausiojo lygmens charakteristikos yra funkcionalumas, patikimumas, panaudojamumas ir našumas. Hierarchija turi 34 savarankiškai matuojamas (elementariąsias) charakteristikas. Panaudojimo kokybė gali būti įvertinta tik turint kokybės charakteristikų įverčius. Kokybės charakteristikas pasiūlėme matuoti tikėtinu sėkmingo kalbos panaudojimo dažniu [3], su kuriuo atitinkama kalbos savybė bus pakankama bet kuriai teoriškai įmanomai sistemai specifikuoti. Tačiau klausimas kaip turi būti vertinamos kokybės elementariosios charakteristikos taip ir liko neatsakytas. Tai yra sudėtinga problema, nes, visų pirma, vidinės kokybės samprata yra santykinė. Nors vidinė kokybė nepriklauso nuo konteksto, reikia atsižvelgti į globalų kontekstą. Tai reiškia, kad kokybės charakteristikų

svarba priklauso nuo konkrečios programų sistemų populiacijos. Kai kurių charakteristikų reikia daugeliui populiacijų, tam tarpe kitos naudojamos labai retai. Tokiu būdu, keičiantis sistemų populiacijai keičiasi kalbos kokybės laipsnis. Antra, tiesiogiai apskaičiuoti tikėtinus sėkmingo kalbos panaudojimo dažnius yra neįmanoma, kadangi neįmanoma turėti pakankamą statistiką apie visų teoriškai įmanomų programų sistemų arba visos šiuolaikinės programų sistemų populiacijos reikalavimus.

Šio straipsnio tikslas – pasiūlyti specifikavimo kalbos funkcionalumo vertinimo metodiką, kurioje būtų atsižvelgta į minėtas problemas. Šios metodikos pagrindą sudaro reprezentatyvių pavyzdžių bibliotekų panaudojimas elementariosioms kokybės charakteristikoms vertinti.

Pagrindiniai reprezentatyvių pavyzdžių konstravimo principai

Pirmiausia, visos teoriškai įmanomos sistemos turi būti padalintos į sistemų kategorijas, kurios tenkina iš esmes skirtingus reikalavimus. Sistemų klasifikavimas turi būti atliktas tokiu būdu, kad kiekviena sistemų kategorija būtų charakterizuojama tam tikra šiai kategorijai specifinių reikalavimų grupe. Nagrinėjimų objektas yra programų sistemos, nes tiktai programų sistemų savybės turi būti specifikuojamos panaudojus tam tikrą specifikavimo kalbą. Pagrindinis tikslas yra įvertinti kalbos funkcionalumą, kuris apibrėžiamas kaip reikalavimams specifikuoti skirtų kalbos savybių visuma [1]. Šiam tikslui pasiekti tinkama taksonomija buvo pasiūlyta Jacksono [5]. Pagrindinis šios taksonomijos klasifikavimo kriterijus yra sistemoje spendžiamų problemų rūšis. Įvairių kategorijų sistemoms aprašyti naudojami probleminiai freimai (angl. *problem frames*). Jacksonas išskyrė septynias sistemų kategorijas, kiekvieną iš kurių aprašo elementarusis probleminis freimas: transformavimo, kontrolės, komandinės elgsenos, sąveikaujančių dalių (angl. *workpiece*), sujungimo (angl. *connection*), informacijos parodymo (angl. *information display*) ir atsakymų į užklausas (angl. *information answer*) sistemos. Sudėtingesnės sistemos turi būti aprašomos panaudojus elementariųjų freimų kombinacijas, taip vadinamus sudėtinius freimus (angl. *multi-frames*).

Pasirinkus tam tikrą programų sistemų kategoriją taikomas nestatistinio imties parinkimo metodas – tikslingas imties parinkimas (angl. *purposeful sampling*). Iš realiai egzistuojančių programų sistemų populiacijos kiekvienai sistemų kategorijai parenkama sistemų imtis. Pagrindinė nestatistinių imties parinkimo metodų problema yra imties parinkimo rizikos įvertinimas. Nors imties patikimumas negali būti teoriškai įvertintas, egzistuoja kiti priimtino patikimumo užtikrinimo būdai. Sprendimas kokius populiacijos elementus reikia įtraukti į imtį turi būti priimamas dalykinės srities eksperto, ir turi remtis šiais pagrindiniais kriterijais: elemento reprezentatyvumas, elemento naudingumas ir elemento santykinė rizika (angl. *relative risk*). Pagal pirmą kriterijų sistema yra pakankamai reprezentatyvi tuomet, kai jinai gali būti aprašyta panaudojus tam tikrą probleminį freimą. Tai reiškia, kad tokiai sistemai sukurti turėtų prireikti visos atitinkamame freime numatytos informacijos. Pagrindinė reprezentatyvumo vertinimo priemonė yra klausimynas, kuris turi būti sudarytas kiekvienai programų sistemų kategorijai. Pagal antrą kriterijų sistema yra pakankamai naudinga tuomet, kai ji yra aktuali, populiari tarp vartotojų ir jos naudojimo sritis yra pakankamai svarbi. Pagal trečią kriterijų santykinė sistemos išrinkimo rizika yra priimtina tuomet, kai šios sistemos reikalavimai gali būti atkartoti (pvz., sistema

turi išsamią pagalbos sistemą, prieinamas demonstracinės versijas). Be to, santykinė rizika sumažinama tinkamai suformulavus reikalavimus renkamiems duomenims. Prieš pradedamas duomenų analizę, ekspertas turi įsitikinti, kad klausimyno duomenys atitinka šiuos reikalavimus. Vienintelis klausimyno duomenų patikrinimo būdas yra rankinė procedūra, kurią turi įvykdyti ne duomenis rinkęs, bet kitas ekspertas. Jis turi nustatyti neatsakytus klausimus, netikslius atsakymus ir kitus duomenų kokybės reikalavimų pažeidimus.

Kiekvienos imtį sudarančios sistemos savybės nustatomos panaudojus dalykinės srities analizės metodą FODA (*Feature-Oriented Domain Analysis Method*) [6]. Sukuriamas savybių modelis (angl. *feature model*), kuris aprašo bendrąsias ir skirtingas visų imties sistemų savybes ir priklausomybes tarp jų.

Specifikavimo kalbų funkcionalumo vertinimo metodika

Siūlomą funkcionalumo vertinimo metodiką sudaro aštuoni žingsniai (1 pav.).

Pirmame žingsnyje turi būti pasirinkta tam tikra programų sistemų kategorija, ir remiantis reprezentatyvumo, naudingumo ir santykinės rizikos kriterijais išrinktos konkrečios šiai kategorijai priklausančios programų sistemos.

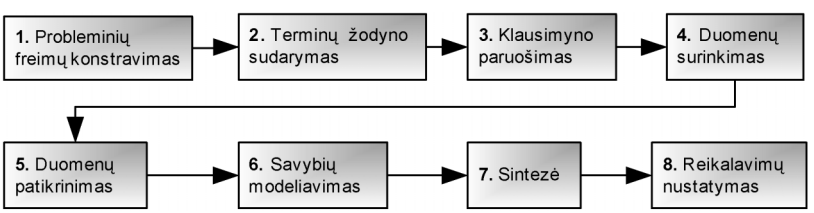
Antrame žingsnyje pasirenkamas tam tikras konceptualizavimas ir apibrėžiami baziniai terminai nustatant jų ekvivalentus visose imties sistemose.

Trečiame žingsnyje paruošiami reikalavimai duomenims ir klausimynas. Kaip klausimynas, taip reikalavimai duomenims turi būti suformuluoti žodyno terminais. Klausimyno ir reikalavimų duomenims išsamumą ir teisingumą turi įvertinti išorinis ekspertas.

Ketvirtame žingsnyje atliekamo duomenų surinkimo metu turi būti išanalizuota kiekviena imties sistema, ir turi būti atsakyta į visus klausimyno klausimus. Sistemų analizės metu kiekviena sudėtinė sistema turi būti išskaidyta į elementariuosius freimus, ir kiekvienas freimas turi būti išanalizuotas atskirai.

Penktame žingsnyje išorinis ekspertas turi atlikti duomenų patikrinimą nustatant neatsakytus klausimus, netikslius atsakymus ir kitus duomenų kokybės reikalavimų pažeidimus, kurie turi būti ištaisyti.

Šeštame žingsnyje pavyzdžių parinkimo rezultatams vertinti ir interpretuoti naudojamas dalykinės srities analizės metodas FODA. Kiekvienai imtį sudarančiai sistemai sudaromas savybių modelis. Modeliavimas atliekamas iš viršaus į apačią, pradedant nuo elementariųjų probleminių freimų.



1 pav. Reprezentatyviųjų pavyzdžių konstravimo metodika.

Septintame žingsnyje atliekama sintezė, kurios metu visų imtį sudarančių sistemų savybių modeliai apjungiami sukonstruojant apibendrintos sistemos, reprezentuojančios visas pasirinktai programų sistemų kategorijai priklausančias sistemas, savybių modelį.

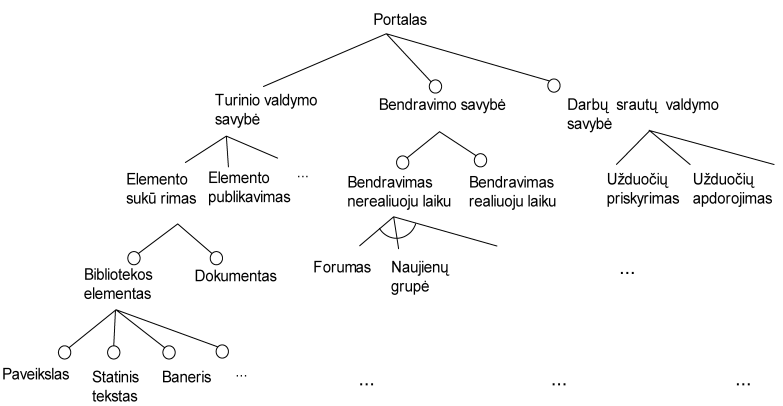
Aštuntame žingsnyje iš savybių modelio nustatomi ir dokumentuojami funkciniai ir nefunkciniai apibendrintos sistemos reikalavimai. Reikalavimai turi būti anotuoti nurodant ar tai nebūtinas, alternatyvus arba būtinas reikalavimas.

Specifikavimo kalbų funkcionalumo vertinimo metodikos taikymas

Pasiūlyta metodika buvo pritaikyta Web portalo reprezentatyviems pavyzdžiams konstruoti. Portalą aprašo dvi atsakymų į užklausas ir dvi sujungimo freimų sistemos. Pirmiausia, sukonstruojama portalą aprašanti sudėtinė freimų sistema. Tuomet pagal reprezentatyvumo, naudingumo ir santykinės rizikos kriterijus išrenkami trys imtį sudarantys Web portalai. Šių portalų analizės metu formuojamas terminų žodynas, suformuluojami reikalavimai renkamiems duomenims ir sudaromas klausimynas. Atsakius į visus klausimyno klausimus surenkami duomenys, pagal kuriuos kiekvienam imties portalui sudaromas savybių modelis (2 pav., 1 lentelė). Savybių modelį siūlome vaizduoti lentelių pavidalu. Lenteles siūloma naudoto dėl to, kad jos leidžia vienoje vietoje pateikti visą kiekvienos savybės informaciją, kuri FODA metode yra išskirstyta tarp savybių diagramų, savybių aprašymų ir savybių loginio pagrindimo.

Visų portalų savybių modeliai apjungiami į vieną apibendrintą savybių modelį, pagal kurį sukuriama Web portalo reikalavimų specifikacija (2 lentelė).

2 lentelėje kiekvienas reikalavimas turi unikalų identifikatorių, kuris atspindi visus aukštesnius hierarchijos lygmenis, reikalavimo rūšį (F – funkcinis reikalavimas, R – patikimumo reikalavimas, U – panaudojamumo reikalavimas, E – našumo reikalavimas, M – palaikomumo reikalavimas, P – pernešamumo reikalavimas) [4], reikalavimo formuluotę, reikalavimo tipą (M – būtinas, O – nebūtinas, A – alternatyvus), prioritetą (M – turi turėti, S – turėtų turėti, C – galėtų turėti, W – neturėtų turėti, bet galėtų turėti ateityje).



2 pav. Portalo savybių modelio fragmentas.

1 lentelė. Portalo savybių modelio lentelės fragmentas

Lygis	**Savybė**	**Aprašymas**	**Loginis pagrindimas**	**Tipas**
1	Turinio valdymo savybė	Portalo turinio kūrimo, valdymo ir publikavimo servisas		Būtinas
1.1	Elemento sukūrimas	Portalo turinio kūrimo servisas		Būtinas
1.1.1	Bibliotekos elementas	Nestruktūrizuotų, daugkartinio naudojimo turinio elementų sukūrimas	Jeigu reikia saugoti dokumentų dalis daugkartiniam naudojimui	Nebūtinas
1.1.1.1	Paveikslas	Vizualaus objekto, vietos, asmens, abstrakcijos ir kt. atvaizdavimo Web puslapyje servisas	Jeigu paveikslai turi būti naudojami keliuose dokumentuose	Nebūtinas
…	…	…	…	…

2 lentelė. Portalo reikalavimų lentelė

ID	**Rūšis**	**Reikalavimas**	**Tipas**	**Prioritetas**
REQ-1.	F	Turinio valdymas	M	M
REQ-1.1	F	Portalo **Pt** elementai galėtų buti bibliotekos elementai ***LbItm*** ir/arba dokumentai ***DocItm***.	O	C
REQ-1.2	F	Bibliotekos elementai ***LbItm*** turi būti nestruktūrizuoti, daugkartinio naudojimo elementai, dažniausiai paveikslai, statinis tekstas ir baneriai.	O	S
…	…	…	…	…

2 lentelėje patekti funkciniai reikalavimai sudaro Web portalo reprezentatyvų pavyzdį. Kiekvienas pavyzdžio reikalavimas buvo specifikuotas panaudojus UML 2.0 ir Z specifikavimo kalbas.

Išvados

Visų prima, pasiūlyta metodika leidžia sukonstruoti tokius reprezentatyvius pavyzdžius, kurie atspindi apibendrintos sistemos, reprezentuojančios tam tikrai programų sistemų kategorijai priklausančias sistemas, reikalavimus. Antra, pasiūlyta metodika leidžia vertinti specifikavimo kalbos funkcionalumą atsižvelgus į globalų kontekstą, tai yra konstruojant reprezentatyvius pavyzdžius kiekvienai programų sistemų kategorijai. Trečia, yra pasiūlyta kaip atlikti kalbos funkcionalumo vertinimą nevertinant visų teoriškai įmanomų tam tikrai kategorijai priklausančių programų sistemų.

Literatūra

1. A. Caplinskas, J. Gasperovic, A taxonomy of characteristics to evaluate specification languages, in: J. Barzdins (Ed.), *Computer Science and Information Technologies*, vol. 672, University of Latvia (2004), pp. 321–336.
2. A. Caplinskas, J. Gasperovic, Functionality of information systems specification language: concept, evaluation methodology, and evaluation problems, in: O. Vasilecas *et al.* (Eds.), *Information Systems Development. Advances in Theory, Practice and Education*, Kluwer Plenum Publishers (2005), pp. 341–351.
3. J. Gasperovič, Evaluation of the functionality of UML amd Z languages, *Technical Report IMI-SED-06-01*, Software Engineering Department, Institute of Mathematics and Informatics, Vilnius, Lithuania. Available at: `http://www.mii.lt/files/imi_sed_06_01_gasperovic.pdf` [accessed November, 2006].
4. ISO/IEC 9126. *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use*. First edition, 1991-12-15, reference number ISO/IEC 9126: 1991(E) (1991).
5. M. Jackson, *Problem Frames*, Addison-Wesley (2001).
6. K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report, CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh (1990).

SUMMARY

J. Gasperovic. Development of representative examples to evaluate specification language functionality characteristics

The paper describes shortly the main principles of representative examples development, provides the essence of the methodology to develop representative examples and an example of its application. The main contribution of the paper is the methodology to develop representative examples that could be used to evaluate the sufficiency of specification language for specification of software system requirements.

Keywords: specification language, functionality, quality characteristic, quality evaluation, representative example.