D.B.Alibiev, A.Sh.Kazhikenova, M.A.Seksembaeva

Investigation of computer system optimization for Windows 7

In this work questions of transportation of oil products by rail with use of special tanks are In the article the most important parts of control of the Windows 7 operating system are described and possible ways of optimaziton of computing systems for Windows 7 are considered. The way a reasoning and researches of materials offers the main councils for maintenance of an operating system in an initial state and councils for optimization of the systems providing its growth in productivity.

References

1 Kolisnichenko D. *Secrets, tweak and optimize Windows 7* Registry, St. Petersburg: BXB-Petersburg, 2010, 320 p.

2 Glazyrin B.E., Glazyrina I.B., Berliner E. *Microsoft Windows 7 user guid*, St. Petersburg: BXB-Petersburg, 2010, 416 p.

3 Chekmaryov A., Reitman M. *Installing and Configuring Windows 7 for maximum performance*, St. Petersburg: BXB-Petersburg, 2010, 368 p.

4 Progdi R.G., Trubnikov A., Tikhomirov V.V. *Teach Yourself Windows 7. Installation, configuration, use*, St. Petersburg: BXB-Petersburg, 2010, 304 p.

ӘОЖ 004.9

Д.Б.Әлібиев, Г.А.Сұлтанова

Е.А.Бөкетов атындағы Қарағанды мемлекеттік университеті (E-mail: gasultanova@mail.ru)

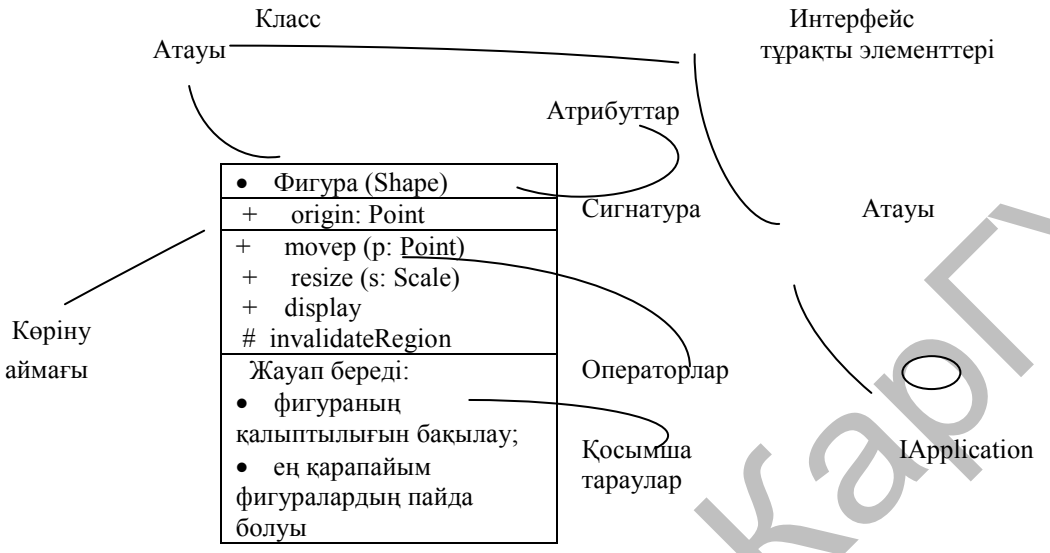
Кластар диаграммасын жобалауда қолдану

Мақалада жобалау іскерілігін үйрену үшін нақты есептер шығарылып, диаграмма құру жолдары қарастырылды. Есепте диаграммалау мен оны құру UML тілінде көрсетілді. Есеп кластар диаграммасының навигациясымен ұйымдастырылды. Есептің жобалық пішіні жасалып, диаграмманың құрылуы оның практикалық маңыздылығын арттырды.

Кілт сөздер: UML, OOSE, IDL, Rational Rose, ассоциация, атрибут.

Қазіргі заманғы коммерциялық программалық жүйелер күрделі және көлемді болып келеді. Күрделіліктің өсуі программалық жүйелерді жобалау әдістемелік салада маңызды зерттеулерді жүргізуге себепші болады, сонымен қатар декомпозициялау, абстракциялау және иерархияны құру әдістері жетілдірілді. Құрылымдық жобалау әдістері күрделі жүйелердің өңдеу үрдісін жеңілдету мақсатында алгоритмдік тәсілге негізделген. Жобалау әдістері кластарды және объектілерді бейнелеуге негізделген. Күрделі программалық жүйелерді құру үшін объектіге-бағытталған ғылымды пайдалану кластар түрінде негізгі UML құрылымдық блоктары жобаланады. Қазіргі таңда объектілерді жобалау, талдау, модельдеу үрдістерін іске асыру Unified Modeling Language — UML тілі болып саналады. Unified Modeling Language тілі — бұл бағдарламалық жүйелерді ерекшелендіру, бұрыштама қою, конструкциялау және құжаттамалау, сондай-ақ модельдер бизнесі мен өзге де бағдарламалық емес жүйелердің тілі. UML бұдан бұрын да үлкен және күрделі жүйелерді модельдеу кезінде ойдағыдай қолданылып жүрген инженерлік әдіс-тәсілдердің бірлестігін көрсетеді. UML-дің құрамалы бөлігі болып OCL табылады (Object Constraint Language — объектілерді шектеу тілі). UML-ды өңдеу 1994 жылғы қазан айында басталды, бұл кезде Rational Software Corporation-нан шыққан Гради Буч (Grady Booch) және Джим Рамбег (Jim Rumbaugh), OMT (Object Modeling Technique — объектілік модельдендіру техникасы) әдістемесін бірыңғайландыру бойынша жұмыстарды бастаған болатын. 1995 жылғы қазан айында бірыңғайландыру әдісінің алдын ала шамаланған болжамы ұсынылды. 1995 жылғы экономиялық құлдырау кезінде Иве Иакобсон (Ivar Jacobson) және оның

Objectory компаниясы Rational-мен бірікті. Бірлесу қорытындысы болып OOSE (Object-Oriented Software Engineering) әдісімен бірыңғайландыру әдісінің қосылуы табылды [1].



1-сурет. Кластың графикалық нотациясы

Rational Rose — автоматтандыру үрдістерін талдау және программалық объектілерді жобалау үшін арналған, сонымен қатар әр түрлі тілдердегі кодтарды генерациялауға және жоба құжатнамаларды шығаруға арналған Rational Software Corporation фирмасының объектілі-бағытталған Case құралдары. Rational Rose UML тіліне негізделіп жобалау және объектілі-бағытталған талдау әдістерін қолданады. Rational Rose осы болжамасы C++, Visual C++, Visual Basic, Java, PowerBuilder, CORBA Interface Definition Language (IDL) бағдарламалар үшін кодтар генерациясын және ANSI SQL, Oracle, MS SQL Server, IBM DB2, Sybase үшін мәліметтер қорының генерация бейнеленуін, сонымен қатар диаграмма түріндегі жобалау құжаттарын іске асырады. Rational Rose жаңа модельдерде бағдарламалық компоненттерінің қайта қолдануын қамтамасыз ететін бағдарламалар мен мәліметтер қорының резервтік инжинирингтің құралдарынан тұрады. Rational Rose да жұмыс істеудің негізі жүйе архитектурасының статикалық және динамикалық аспектілерін анықтайтын UML тілі диаграммаларды құру болып табылады (1-сур.).

Кластар диаграммасы объектіге-бағытталған амалда негізгі рөлді атқарады. Негізінде кез келген әдістеме түрлі кластар диаграммасынан тұрады. Сонымен қатар оның құрамында модельдеу түсініктерін көбісін құрайды. Оның негізгі элементтерінің көпшілігі пайдаланса да, өте күрделі ұғымдары көп пайдаланыла бермейді. Сондықтан да алдымен негізін, сонан соң қосымша түсініктерді қарастырайық [2].

Кластар диаграммасы жүйе объектілерінің типтерін, олардың арасындағы статистикалық қатынастары түрлерін сипаттайды. Статистикалық қатынастың екі негізгі түрі бар:

– ассоциациялар;

– ішкі типтер.

Кластар диаграммасында кластар атрибуттары, кластар операциялары мен объектілер байланыстарына қойылатын шектеулер де бейнеленеді.

Кластар диаграммасын көрсету ерекшеліктері UML тілін сипаттау құрамына жатпайды. Бірақ олар модельді құру және зерттеу кезінде маңызды рөл атқарады. UML тілін осы көзқарастың кез келгенінде қолдануға болады. Сіз класты стереотиппен толықтырып көріністің ерекшелігін айқын көрсетуіңізге болады. Сонымен бірге класты «іске асыру класы» деп немесе концептуалды және спецификация көзқарасы үшін «тип» ретінде белгілеуге болады. Бұл амал іске асырудың көзқарас түрін айқын көрсету үшін жасалады. Мысалы, 2-суретте кластар диаграммасының тұтынушылар тапсырыстарын өңдеумен айналысқан өңдеушілердің барлығына түсінікті, қарапайым түрі берілген. Бұл диаграмманың әрбір бөлігін қарастырып, түрлі көзқарас тұрғысынан олардың интерпретациясы көрсетілді.

Ассоциация класс көшірмелері арасындағы қатынасты реттейді. Концептуалды көзқарас тұрғысынан қарағанда, ассоциация кластар арасындағы концептуалды қатынастарды көрсетеді [3].

Диаграммада тапсырыс тек бір тұтынушыдан келуі мүмкін, ал тұтынушы уақыт аралығында бірнеше тапсырыс түрін бере алатыны көрсетілген. Берілген тапсырыстың әрбіреуі бірнеше тапсырыс жолынан тұруы мүмкін және әрбір жол бір тапсырысқа сәйкес келуі керек.

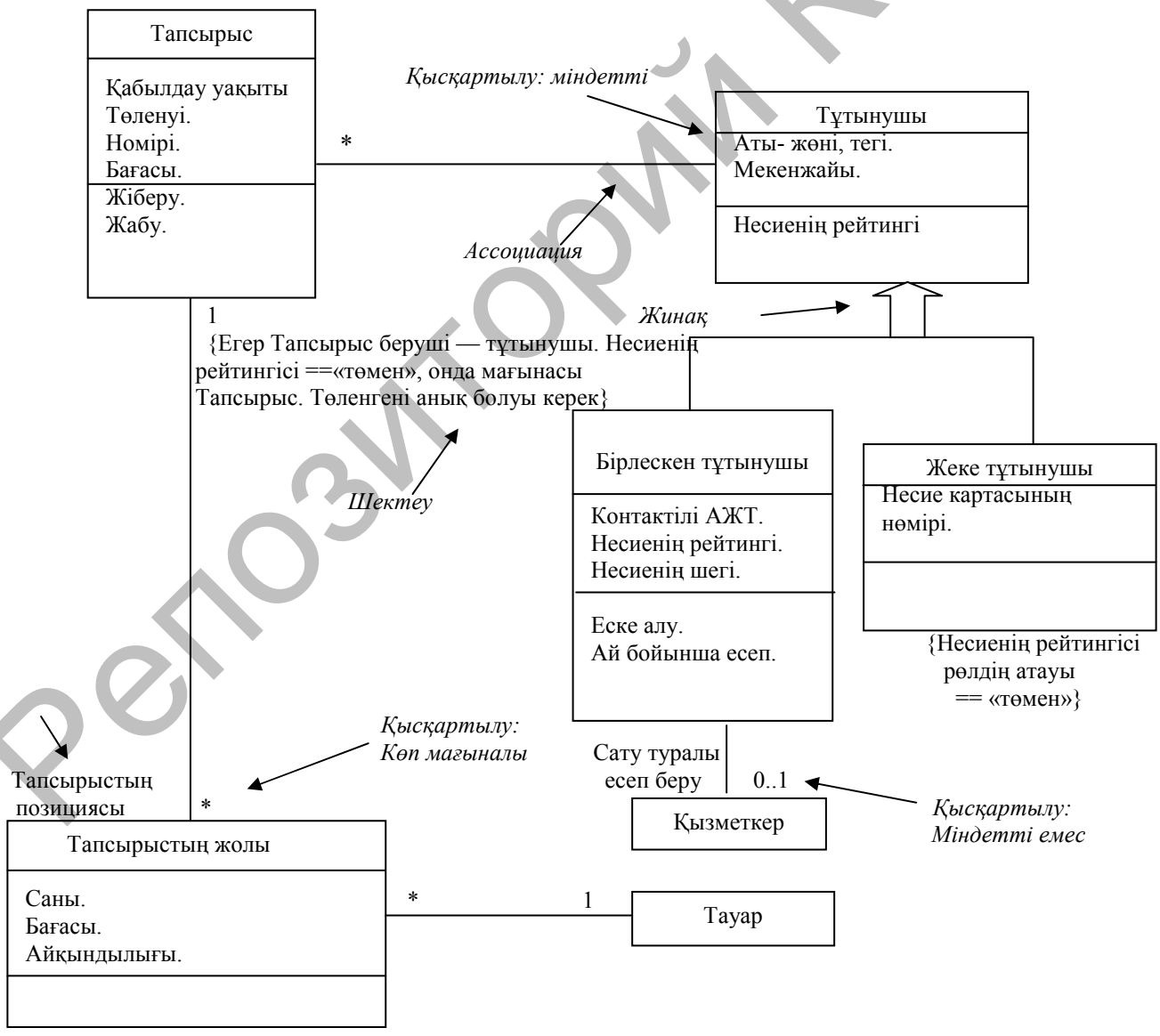
Әрбір ассоциация ассоциацияның екі ұшына ие және оның әрбір ұшы осы ассоциацияның кластарының біріне жалғанады. Ассоциация ұшы белгімен берілуі мүмкін. Мұндай белгі рөл атауы деп аталады. Мысалы, 2-суретте «Тапсырыс» класынан «Тапсырыс жолына» бағытталған ассоциация ұшы «Тапсырыс позициясы» деп аталады. Егер мұндай белгі болмаса, ассоциация ұшына кластың атауы меншіктеледі. «Тапсырыс» класынан «Тұтынушы» класына бағытталған ассоциация ұшы деп аталуы мүмкін. Ассоциация ұшы берілген қатынаста қанша объект қатысатынын көрсететін қысқартылуға ие. 2-суретте «Тапсырыс» класының қасындағы «*» символы бір тұтынушымен бірнеше тапсырыс байланысуы мүмкін екенін көрсетеді, ал, керісінше, «1» символы әрбір тапсырыс тек бір тұтынушыдан келу мүмкін екенін білдіреді. Жалпы жағдайда, қысқартылу қатынаста қатысатын объектілер санының төменгі және жоғарғы шегін береді. Мұндағы «*» символы *0..шексіздік* диапазонын береді, тұтынушы бір де бір тапсырыс бермеуі мүмкін, бірақ ол беретін тапсырыс саны шектелмеген, «1» және «1..1» диапазонын береді, яғни тапсырысты тек жалғыз тұтынушы бере алады [3].

Қысқартылудың жиі қолданылатын түрлері:

– 1;

– *;

– 0..1 (нөл немесе бір).

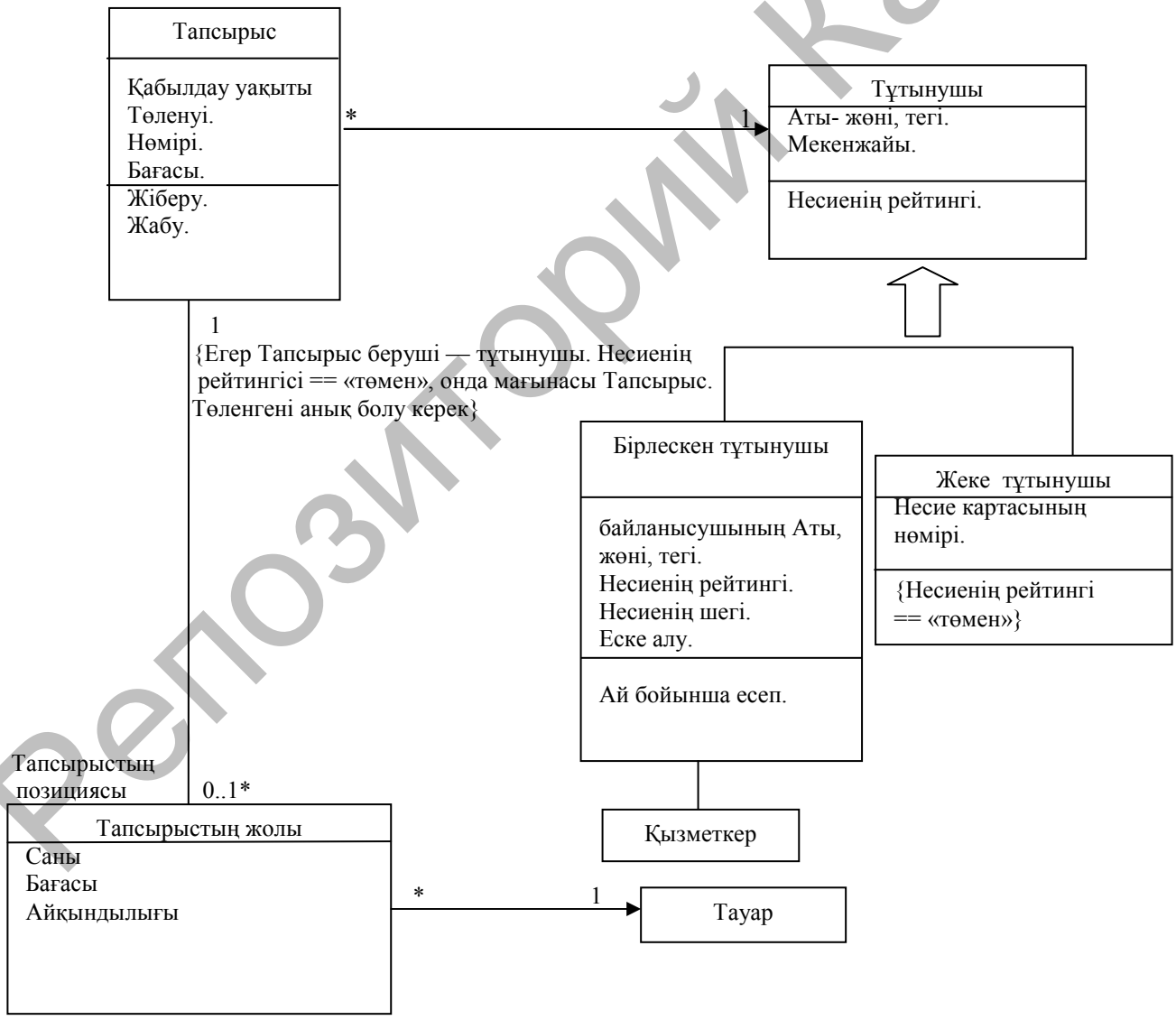


2-сурет. Кластар диаграммасы

Жалпы жағдайда, қысқартылу үшін бір ғана сан, диапазон немесе диапазон немесе сандардан тұратын дискретті комбинация қолданылуы мүмкін. Спецификация көзқарасы тұрғысынан ассоциациялар кластар жауапкершіліктерін көрсетеді. Мысалы, 2-суретте тұтынушымен байланысты бір немесе бірнеше әдістер бар деп тұжырымдалады. Ол арқылы нақты тұтынушы қандай тапсырыс бергенін білуге болады. Осыған ұқсас «Тапсырыс» класында, қандай тұтынушы нақты қандай тапсырыс береді және осы тапсырысқа қандай тапсырыс жолдары кіретінін көруге болатын әдістер бар.

Диаграмманы спецификация көзқарасы тұрғысынан қарастыра отырып, біз класқа арналған мәліметтер құрамы жайында тұжырым айта алмаймыз. Берілген класс тапсырыспен байланысты екенін анықтау үшін, «Тапсырыс» класы «Тұтынушы» класына бағытталған нұсқауға ие екенін немесе «Тапсырыс» класы «Тұтынушы» класының әрбір көшірмелеріне құрылған программалық тапсырысты жүзеге асыру арқылы орындалуын іске асыра алатынын айтамыз. Диаграмма тек интерфейсті сипаттайды. Егер модель іске асыру көзқарасы тұрғысынан қарастырылса, байланысқан кластар арасында екі бағытта да нұсқау бар деп ұйғаруға болады [4].

Бұл жағдайда диаграмма бізге «Тапсырыс» класының құрамындағы «Тапсырыс жолы» класына бағытталған нұсқаулары бар жолды және «Тұтынушы» класына бағытталған нұсқауды көрсетеді. Мысалы 3-суретті қарастыратын болсақ, бұл бағыттауыштар навигацияны білдіреді. Спецификация моделінде бұл амалмен «Тапсырыс» қай «Тұтынушыға» байланысты сұрағына жауап беруі тиіс, ал «Тұтынушыға» қай «Тапсырысқа» қатысы бар екеніне жауап беру тиіс еместігін көрсетуге болады. Симметриалды жауапкершіліктердің орнына біз тек бір бағытты жауапкершіліктерді көрсетеміз.



3-сурет. Кластар диаграммасы навигациясы

Іске асыру диаграммасында бұл «Тапсырыс» класы «Тұтынушы» класына бағытталған нұсқауға ие, бірақ «Тұтынушы» «Тапсырыс» класына бағытталған нұсқауға қамтымайтынын білдіреді.

Бұдан навигация спецификация мен іске асыру диаграммалары үшін өте маңызды мағынаға ие екенін көруге болады. Бірақ концептуалды диаграммаларды құру барысында ол өз маңыздылығын жоғалтуы мүмкін.

Ең басында құрылатын концептуалды диаграммаларда навигация бағыты көбіне болмауы мүмкін. Олар спецификация мен іске асыру деңгейіндегі кластар диаграммасын құру барысында пайда болады. Спецификация мен іске асыру көзқарасы тұрғысынан навигация бағыты ерекшеленуі мүмкін.

Егер навигация бағыты тек бір жаққа қараса, онда мұндай ассоциация бір бағытты ассоциация деп аталады. Екі бағытты ассоциацияда навигация екі жаққа да бағытталады. Егер ассоциация навигация бағытына ие болмаса, онда UML тілі оны келесі түрлерде береді: навигация бағыты белгісіз немесе ассоциация екі бағытты болады. Екі бағытты ассоциациялар құрамына, екі навигация бір-біріне қарағанда, инверсті болып келетін қосымша шектеулер кіреді. Бұл математикадағы кері ұғымына ұқсас.

Мәліметтерді модельдеу мамандары қабылдағандай, ассоциацияны атаудың дәстүрлі тәсілі етістік пен сөйлемнің байланысқан түрін пайдалану негізделеді. Бұл амал сөйлемде қолдануға ыңғайлы болу үшін пайдаланылады. Объектілерді модельдеу мамандарының көпшілігі атау ретінде пайдалану дұрыс деп санайды. Оны ассоциацияның бір немесе бірнеше ұштарының рөлін атқару үшін қолданылады. Себебі мұндай амал операциялар мен жауапкершіліктермен жақсы байланысады.

Кейбір өңдеушілер әрбір ассоциацияға белгілі бір атау меншіктейді. Мысалы, «қамтиды» немесе «байланысты» сияқты. Ассоциация екі объект арасындағы перманенттік байланысты көрсетеді. Дегенмен, мұндай байланыс, онымен байланысқан көшірмелер уақыт аралығында өзгерсе де, объектілердің барлық өмір ұзақтығы барысында болады. Концептуалды деңгейде атрибут «Тұтынушының аты, жөні, тегі» атрибуты тұтынушылар атына, жөніне, тегіне ие екенін көрсетеді. Спецификация деңгейінде бұл атрибут тұтынушы объектісі өзінің атын, жөнін, тегін хабарлауы мүмкін және атын, жөнін, тегін орнату үшін тәсілдерге ие. Иелену деңгейінде тұтынушы объектісі өзінің атауы үшін өрістен тұрады.

Диаграммаларды детализациялау дәрежесінен тәуелділігінен атрибут белгілеуі атрибуттың атауын, типі мен үндемей беретін мәнін меншіктеуді жатқызуы мүмкін. UML тілінің синтаксисінде келесі түрде болады: <көрінуі><атауы>:<тип> = <үндемей берілетін мән>, мұндағы көрінуі төменде көрсетілгендей операция үшін сондай мағынаға ие.

Диаграммада, ереже бойынша, атрибут міндетті немесе міндетті емес екендігі көрсетілмейді, бірақ мұны істеуге болады, атрибуттың атынан кейін квадрат жақшаға қысқасысын көрсетеміз, мысалы, мәлімет алу [0..1]: мәліметтер.

Спецификация мен иелену деңгейінде айырмашылық білінеді. Атрибуттар навигациясының жалғыз бағытын ұйғарады — типтен атрибутқа. Бұдан басқа, тип объектінің өзінің меншікті көшірмесінен тұратыны түсіндіріледі. Өз кезегінде, атрибут ретінде қолданатын кез келген тип сілтеменің семантикасына қарағанда мәннен тұратындығы ұйғарылады [5].

Кейбір типтің жалпыға бірдей әдістеріне операцияны спецификациялау деңгейінде сәйкес келеді. Әдетте мұндай операцияларды көрсетпесе де болады, олар жай ғана атрибутты игереді. Бірақ кейбір кезде көрсету керек болады, себебі берілген атрибут тек оқу үшін арналған (*read-only*) не өзгеріссіз (*frozen*) болады, яғни оның мәні ешқашанда өзгермейді.

UML тілі сұранысты кластан әлдебір мәндер алатын және жүйенің жағдайын өзгертпейтін операция деп анықтайды. Мұндай операцияны шектеумен {сұранысты} белгілейді. Сұраныс кезектесіп орындала алады, бірақ жағдайды өзгертетін операциялардың кезектесіп орындалуын қадағалау қажет ететін әдістер:

– мәндерді ығыстыру әдістері (*getting methods*);

– мәндерді орнату әдістері (*setting methods*).

Мәндерді ығыстыру әдісі өрістен мәнді қабылдайды. Мәндерді орнату әдісі мәнді өріске орналастырады. Кластан тыс болғанда тұтынушы, мәнді алу әдісінің сұрауы немесе мәнді орнату әдісінің модификаторы болып табылатынын анықтай алмайды.

Операция әлдебір процедураны шақыру, ал осы жерде әдіс процедураның денесі болып табылады. Бұл екі ұғымды полиморфизммен кездескенде ажыратады. Егер сізде үш ішкі типтен тұратын супертип бар болса, онда оның әрбіреуі бір және тура сол супертиптің операциясын

анықтайды, яғни сіз бір операция және оны жасайтын төрт әдіспен жұмыс істейміз. Әдетте «операция» және «әдіс» терминдері өзара ауыстыратын болып қолданады, бірақ кейбір жағдайларда оларды ажырата білу керек. Кейбір кезде құрастырушылар оларды әдісті шақыру, немесе әдісті анықтау (операция үшін), «әдіс денесі» деген терминдерді қолданып ажыратады.

Класс диаграммасы түгелдей объектіге-бағытталған әдістердің негізгі қалаушысы болып табылады. Класс диаграммасын қолдануына байланысты кездесетін қиындықтар:

– барлық рұқсаты бар ұғымды бірден пайдалануға тырыспаңыз. Ең қарапайым ұғымнан бастаңыз, олар мына бөлімдерде қарастырылған: кластар, атрибуттар, ассоциациялар, жалпылаулар және шектеулер;
– модельді құру үшін көзқарастар таңдауы жобамен жұмыс істеу кезеңдеріне сәйкес келуі қажет;
– әлемдегі барлығына модель құра беруге болмайды, оның орнына негізгі аспектілерде көңіл бөлген дұрыс.

Саны көп ескі модельдерден гөрі, әрқашанда жұмыс істеуге болатындай және барлық енгізілген өзгерістер кескінделетіндей, аз диаграмма болғаны жөн. Класс диаграммасына байланысты ең үлкен қауіп — ол бөлшекті игеру. Бұған қарсы тұру үшін концептуалды аспектіге және аспектінің спецификациясына көңіл бөлуіңіз керек.

Әдебиеттер тізімі

1 *Розенберг Д., Скотт К.* Применение объектного моделирования с использованием UML и анализ прецедентов. — М.: ДМК Пресс, 2002. — С. 150–160.

2 *Ларман К.* Применение UML и шаблонов. Введение в объектно-ориентированный анализ и проектирование. — СПб.: Вильямс, 2002. — С. 658–624.

3 *Шмуллер Дж.* Освой самостоятельно UML. — М.: Вильямс, 2002. — С. 330–340.

4 *Трофимов С.А.* Case-технологии. Практическая работа в Rational Rose. — М.: Бином, 2001. — С. 260–267.

5 *Гома Х.* UML-проектирование систем реального времени, распределенных и параллельных приложений. — М.: ДМК Press, 2002. — С. 735–740.

Д.Б.Алибиев, Г.А.Султанова

Использование диаграмм классов в планировании

В статье рассмотрены решение конкретных задач и способы создания диаграмм. Создание диаграмм задачи основывалось на языке UML. Задача была организована с навигацией диаграммы классов. Создание диаграммы и проектной формы задачи увеличивает ее практическую значимость.

D.B.Alibiev, G.A.Sultanova

Using diagrams class at planning

The article is discussed the solution of specific problems, methods for creating diagrams. Diagrams problem and its creation shows the language UML. Tasks organized navigation class diagram. Create a chart increases its practical significance, making the design task form.

References

1 Rozenberg D., Skott K. *Application object modeling using UML and analysis of precedents*, Moscow: DMK Press, 2002, p. 150–160.

2 Larman K. *Applying UML and Patterns. Introduction to Object-Oriented Analysis and Design*, Saint-Petersburg: Williams, 2002, p. 658–624.

3 Shmuller Dzh. *Teach Yourself UML*, Moscow: Williams, 2002, p. 330–340.

4 Trofimov S.A. *Case-technology. Practical work in Rational Rose*, Moscow: Bean, 2001, p. 260–267.

5 Goma H. *UML Design of real-time systems, distributed and parallel applications*, Moscow: DMK Press, 2002, p. 735–740.