УДК 004.4’273 : 004.43 : 81’32

Н.Г. Кеберле, Д.В. Чесановський

Запорізький національний університет, Запоріжжя

МЕТОД ВИЗНАЧЕННЯ ГРАМАТИКИ ПРОБЛЕМНО-ОРІЄНТОВАНОЇ МОВИ ДІАГРАМ

Можливість створення синтаксично коректних діаграм, заданих на деякій проблемно-орієнтованій мові реалізується або процедурними засобами - вбудуванням у візуальні редактори інструментів перевірки коректності діаграми, або декларативними засобами - на рівні граматики мови діаграм. У роботі запропоновано метод декларативного визначення граматики проблемно-орієнтованої мови діаграм. Показано застосування методу для мови ER-діаграм.

Ключові слова: *діаграма, проблемно-орієнтована мова діаграм, контекстно-вільна граматика, синтаксичний аналізатор.*

Вступ

Графічні редактори складають великий клас програмного забезпечення. Це обумовлено тим, що людині зручно використовувати наочні засоби, такі як діаграми, графіки, візуальні моделі для прийняття рішень. На даний час відомо багато графічних нотацій для зображення діаграм, які вважаються розповсюдженими, наприклад, мови діаграм стандарту UML[1], мови діаграм для опису бізнес-процесів (сімейство стандартів IDEF[2]), мови блок-схем алгоритмів [1-2], Blockly[3] та інші. Для поширених стандартів досить легко знайти потрібний редактор, але якщо виникає потреба скористатися вузькоспеціалізованими нотаціями діаграм, необхідно або використовувати спеціальне програмне забезпечення, однією з функцій якого буде візуалізація діаграм, або, якщо це нова нотація, то є вірогідність того, що такого програмного забезпечення не існує взагалі.

Для подібних мов розробляються спеціальні візуальні редактори, що враховують особливості компонування елементів діаграм відповідно до стандартів. Водночас, менш поширені, відомі у спеціалізованих спільнотах, мови діаграм часто не мають «власного» редактора для візуального проектування, і тому контроль за компонуванням елементів залишається на розсуд людини, що створюватиме діаграми у деякому графічному редакторі загального призначення (такому, як, наприклад, Microsoft Visio [3]).

Огляд відомих візуальних редакторів діаграм загального призначення (Microsoft Visio [1-3], DIA [4], yEd [5] та ін.), які надають можливості роботи з поширеними діаграмами та мають можливість підключати додатково деякі нестандартні графічні елементи, показав, що такі редактори не надають користувачеві можливість самостійно описувати допустимі компонування таких елементів.

Метою даної роботи є розробка методу визначення граматики проблемно-орієнтованої мови діаграм, який надасть можливість створювати необхідні ресурси для подальшого використання цієї граматики у спеціалізованих редакторах діаграм.

1. Джерела дослідження та огляд відомих рішень

Проблемно-орієнтована мова [6] - це або мова програмування, або мова специфікацій, що виконуються, яка надає засоби опису (за допомогою відповідних абстракцій та нотації), які обмежені певною предметною областю. Елементи проблемно-орієнтованої мови відображають певні елементи предметної області, таким чином, кожний опис певного стану речей у предметній області є набором «речень» на проблемно-орієнтованій мові.

Проблемно-орієнтовані мови можуть не мати спеціальної візуальної нотації. Для випадків, коли така нотація існує, будемо називати зовнішню проблемно-орієнтовану мову, яка має візуальну нотацію, **проблемно-орієнтованою мовою діаграм**.

Виникає проблема малювання синтаксично коректних діаграм. Можливість створення синтаксично коректних діаграм реалізується або процедурними засобами - вбудуванням у візуальні редактори інструментів перевірки коректності діаграми, або декларативними засобами - на рівні граматики мови діаграм.

Багато редакторів діаграм загального призначення дозволяють використання нестандартних елементів на діаграмі. У Microsoft Visio [3-5] для існуючих елементів дозволяється змінювати їх заповнення, типи зв’язків, типи закінчень зв’язків, а також додаткові атрибути елементів, наприклад, мітки та значення. Діаграми з нестандартними елементами можна експортувати у Visio XML формат, як і звичайні діаграми, або генерувати з XML-описів та/або таблиць значень, при чому коректність такого опису перевіряється на рівні валідації XML, а не на рівні синтаксису мови діаграми.

[1] http://www.uml.org/
[2] http://ru.wikipedia.org/wiki/IDEF
[3] https://code.google.com/p/blockly/

У редакторі графів yEd [7] можливо додавання нестандартних елементів з файлів векторної, растрової графіки [8], а також з трафаретів Microsoft Visio. Додатково, редактор дозволяє додавати у палітру довільну комбінацію вже існуючих елементів діаграм. yEd дозволяє створювати графи за матрицею сумісності або за переліками вершин та ребер. Водночас, контроль за коректністю зв'язків вершин, вбудований у код редактора, ведеться лише для певних наборів елементів – UML, ER, Swim lane.

У редакторі DIA [9, 10] нестандартні елементи можна задати у XML-файлі за допомогою базових SVG специфікацій (line, polyline, polygon, circle та інші) та їх параметрів.

До інструментальних рішень із вбудування довільних проблемно-орієнтовних мов діаграм відноситься Microsoft DSL SDK [11, 12]. Microsoft DSL SDK дозволяє описати у XML-файлах окремі елементи діаграм (вершини, ребра) та об'єднати їх у палітру. Надалі описи діаграм можна перевірити на семантичну коректність, якщо додатково розробити програмний код [12] перевірки. Синтаксична коректність діаграми із нестандартними елементами не перевіряється.

Існує інструментарій перевірки синтаксичної коректності речень деякої формальної мови: відомі генератори лексичних аналізаторів для довільних текстів Flex[4], синтаксичних аналізаторів Bison[5], Boost.Spirit[6] (для внутрішніх проблемно-орієнтованих мов), Coco/R[7] та інші. Використання стандартних синтаксичних аналізаторів для деякої мови діаграм вимагає розробки власної граматики цієї мови.

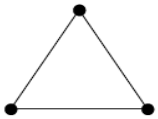
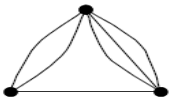
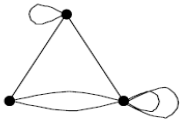
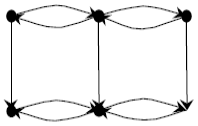
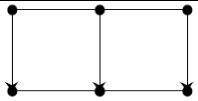
2. Формалізація типів діаграм

Діаграма –графічне зображення, що наочно у вигляді певних геометричних фігур показує співвідношення між різними величинами, які порівнюються. Одними з найвідоміших видів діаграм є графіки та гістограми. Діаграми в основному складаються з геометричних об'єктів (точок, ліній, фігур різних форм та кольорів) та допоміжних елементів (підписів, умовних позначень). Також діаграми можуть бути двовимірні та просторові (тривимірні або об'ємні). За змістом діаграми можна умовно поділити на такі, що відображають динаміку розвитку деякої величини (залежність від часу або відносна послідовність етапів), залежності розміщення елементів (просторова залежність), інші логічні зв'язки між елементами у предметній області (зумовлені семантикою предметної області).

В якості математичної основи для моделювання типів діаграм було обрано мову опису графів [13]. Вивчення типів графів та відомих типів діаграм показало, що кожному типу графа можна співставити один чи декілька типів нотацій діаграм з різних галузей знань. У табл. 1 перелічені типи графів та відповідні найбільш відомі типи діаграм. У подальшому, вважатимемо, що діаграма представляє собою орієнтований граф G={NodeTypeSet, EdgeTypeSet}, де EdgeTypeSet – множина типів ребер, NodeTypeSet – множина типів вершин. При цьому, множина типів вершин складається з множини вершин, які відповідають вузлам діаграми – EndNodeTypeSet, а також – з множини вершин, які відповідають центрам n-арних зв'язків – RelationCentreNodeTypeSet (рис. 1).

Таблиця 1

Типи графів та приклади типів діаграм, які можуть бути представлені такими графами

Вид графа	Приклад графа	Приклади типів діаграм
Простий граф		UML Objects
Мультиграф		DFD, IDEF3, UML Collaborations, UML Deployment
Псевдограф		Crow's feet (IDEF1X)
Орграф		Діаграми Ганта, PERT діаграми, UML sequences, UML activities, блок-схеми
Змішаний граф		ER (Chen), UML Use Case, UML Classes

В даній роботі підписи елементів (назви елементів, кардинальності зв'язків) діаграми не розглядаються як окремі елементи. Урахування таких видів елементів діаграми можливо в межах графової моделі діаграми, наприклад, за допомогою кратних ребер або зважених ребер/вершин.

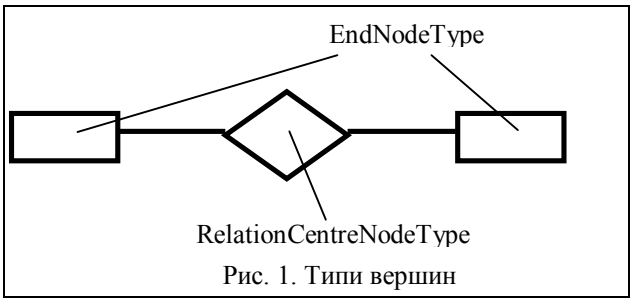


Рис. 1. Типи вершин

Типи ребер відрізняються виглядом та можливістю приєдання одного або декількох вершин типу RelationCentreNodeType.

[4] http://flex.sourceforge.net/
[5] http://www.gnu.org/software/bison/
[6] http://boost-spirit.com/home/
[7] http://www.ssw.uni-linz.ac.at/Coco/

Запропонована графова модель подання діаграм розрізняє два типа вершин та один чи більше типів ребер. Додавання нових типів вершин, обмеження на зв'язність графу, його деревоподібність, обмеження на типи зв'язків між певними типами вершин, урахування топології графу відкриває можливості для формалізації більш складних діаграм.

3. Граматика проблемно-орієнтованої мови діаграм

Для формалізації проблемно-орієнтованих мов використовуються інструменти формальних граматик [13], теорії автоматів [14 – 16].

Для зовнішніх проблемно-орієнтованих мов [17] найчастіше розробляються формальні граматики. Це дозволяє, не використовуючи уміння з програмування, визначити структуру речень мови. Формальна граматика [13] – це четвірка G={N,T,P,S}, де T – алфавіт термінальних символів, N – алфавіт нетермінальних символів, S – аксіома, або спеціально виділений нетермінальний символ, з якого починається опис граматики, та P – множина правил виведення.

Використаємо розширену форму Бекуса-Наура для опису контекстно-вільної граматики мови для опису ER та IDEF1X діаграм.

Термінальні символи є такими (зовнішній вигляд деяких з них показано у табл. 2 та 3):

T= { ' (', ')','=',' Arrow',' Rhombus', Line3', Line4', 'Empty', 'Line2', 'Line1', 'Circle', 'Solid', 'Dash', ' Dot', 'DashDot', 'DashDotDot', 'NULL', '0', '1', ' " ', ';' }.

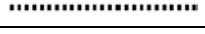
Таблиця 2

Перелік можливих закінчень дуг

Ім'я	Вигляд	ID
Arrow		0
Rhombus		2
Line3		3
Line4		4
NULL	<Empty>	-1
Line2		6
Line1		5
Circle		1

Таблиця 3

Перелік можливих типів дуг

Ім'я	Вигляд	ID
Solid		0
Dash		1
Dot		2
DashDot		3
DashDotDot		4

Множина нетермінальних символів є такою:

N = {node, edge, constraint, end_node,
relation_centre_node, picture_name, node_name,
relation_centre_node_def, left_arrow_type,
left_end_line_type, right_end_line_type,
right_arrow_type, left_end_multiplicity_type,
right_end_multiplicity_type,
edge_name, constraint_name}.

Аксіома S = {DIAGRAM}.

Множина правил виводу P складається з таких правил:

DIAGRAM::= node* constraint* [edge*]
node::= end_node | relation_centre_node
end_node::= picture_name
relation_centre_node::= node_name '=(%' relation_centre_node_def ');'
relation_centre_node_def::= {picture_name | NULL}
edge::= edge_name ' = (' left_arrow_type, left_end_line_type, relation_centre_node_def, right_end_line_type, right_arrow_type, left_end_multiplicity, right_end_multiplicity ');'
constraint::= constraint_name '=(" ' end_node ' " ', edge_name '('constraint_name*')'*');'
left_arrow_type::=' {Arrow' | ' Rhombus' | Line3' | Line4' | 'Empty' | 'Line2' | 'Line1'| 'Circle'}
right_arrow_type::=' {Arrow' | ' Rhombus' | Line3' | Line4' | 'Empty' | 'Line2' | 'Line1'| 'Circle'}
left_end_line_type::= {'Solid' | 'Dash' | ' Dot' | 'DashDot' | 'DashDotDot'}
right_end_line_type::= {'Solid' | 'Dash' | ' Dot' | 'DashDot' | 'DashDotDot'}
left_end_multiplicity::= {'0' | '1'}
picture_name::= рядок символів
node_name::= рядок символів
edge_name::= рядок символів
constraint_name::= рядок символів

4. Опис метода визначення граматики

Запропонований метод побудови граматики мови діаграм складається з наступних етапів:

Етап 1. Опис мета-мови діаграм, що відповідає окремому виду графу.

Етап 2. Генерація синтаксичного аналізатора для мов, описаних в даній мета-мові діаграм.

Етап 3. Побудова конкретної проблемно-орієнтованої мови діаграм.

Для реалізації запропонованого методу був використаний один з відомих генераторів синтаксичних аналізаторів. Coco/R [18] - програма генерації компіляторів або інтерпретаторів мови. Coco/R зчитує файл опису мета-мови та генерує ряд наступних файлів:

– лексичний аналізатор (сканер);
– синтаксичний аналізатор (парсер);
– інформаційні файли (лістинг, таблиця лексем мови).

У опис мета-мови додаються спеціального виду коментарі (..), в яких укладено код для виконання

додаткових дій. Як правило, це код для занесення даних в таблиці ідентифікаторів, генерації коду або його інтерпретації.

На рис. 2 показано опис однієї мета-мови діаграм, підготовлений для Coco/R версії для C++.

```
#include <iostream>
COMPILER META
CHARACTERS
Blet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ-
abcdefghijklmnopqrstuvwxyz".
Num = "0123456789".
cr  = '\r'.
lf  = '\n'.
tab = '\t'.
TOKENS
ident= Blet{Blet|Num}.
Numbr=Num{Num}.
StrEnd=cr|lf|(cr lf).
File="\""(Blet|Num){Blet|Num}".png\"".
IGNORE tab
PRODUCTIONS
ER={Strk}.
Strk=ident(. NameTmp=coco_string_create(t->val); .)"="("( node|
edge | constraint|)");" {StrEnd}. //
node='%'(File|"NULL")(.
(*p).AddObj(0,NameTmp);(*p).AddObj(1,t->val); .).
edge=ident(. (*p).AddConn(0,NameTmp); (*p).AddConn(1,t-
>val); .)',
'ident(. (*p).AddConn(2,t->val); .)',
'ident(. (*p).AddConn(3,t->val); .)',
'ident(. (*p).AddConn(4,t->val); .)',
'ident(. (*p).AddConn(5,t->val); .)',
'('0'|'1')(. (*p).AddConn(6,t->val); .)',
'('0'|'1')(. (*p).AddConn(7,t->val); .).
constraint=File(. (*p).AddKlass(0,NameTmp);
(*p).AddKlass(1,t->val); .){',
'(ident(. (*p).AddKlass(2,t->val); .)
'('ident(. (*p).AddKlass(3,t->val); .){',
'ident(. (*p).AddKlass(3,t->val); .)}')')}.
END META.
```

Рис. 2. Приклад опису мета-мови для Coco/R (виділено дозволені типи елементів діаграм)

В результаті роботи Coco/R створюється набір файлів: SLang.h, Scanner.h, Parser.h. За допомогою цих файлів візуальний редактор діаграм, у якому будуть використовуватися ці інструменти, зможе завантажувати та обробляти описи проблемно-орієнтованих мов діаграм, що можуть бути описані в обраній метамові. Таким чином, мова діаграм вбудовується у програму візуалізації та доступна для користувачів вже через панелі інструментів роботи з діаграмами одного типу.

5. Приклад опису мови діаграм

Продемонструємо роботу методу опису граматики для мови ER діаграм [19]. Граматика мета-мови діаграм у вигляді, необхідном для Coco/R, задається одноразово (рис. 2). На рис. 3 показано зміст файлу з описом конкретного підкласу ER-діаграм – діаграм, що дозволяють визначати типи сутностей та складені сутності-суперкласи. Одна з діаграм, яка створена вручну з використанням елементів, поданих на рис. 3, у спеціалізованому редакторі діаграм, показана на рис. 4. Підписи додані вже після отримання малюнку діаграми.

```
node1=(%NULL);
node2=(%"rb.png");
edge1=(Empty,Solid,node2,Solid,Empty,1,1);
edge2=(Empty,Solid,node1,Solid,Empty,0,0);
edge3=(Arrow,Solid,node1,Solid,Empty,1,0);
constraint1=("sq.png",edge1(constraint1),
edge2(constraint2,constraint3),
edge3(constraint1));
constraint2=("el1.png",edge2(constraint1));
constraint3=("el2.png",edge2(constraint1));
```

Рис. 3. Опис підкласу ER-діаграм

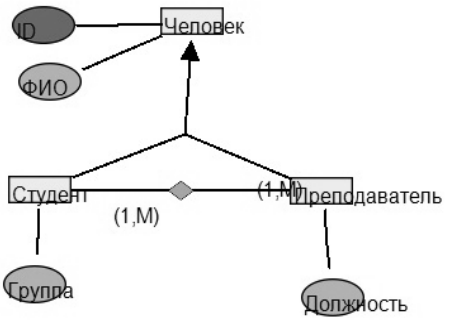


Рис. 4. Приклад ER-діаграми

Висновки та напрям подальших досліджень

Запропонована графова модель подання діаграм розрізняє два типа вершин та один чи більше типів ребер. Додавання нових типів вершин, обмеження на зв'язність графу, його деревоподібність, обмеження на типи зв'язків між певними типами вершин, урахування топології графу відкриває можливості для формалізації складних діаграм.

Запропонований метод визначення граматики мови діаграм спирається на створення мета-мови для опису певного типу графів. Ця мета-мова визначає допустимі типи елементів діаграм. Мова конкретного типу діаграм, що відповідають обраному типу графів, визначається декларативно.

Список літератури

1. Единая система программной документации. Схемы алгоритмов, программ данных и систем. Условные обозначения и правила выполнения (ISO 5807-85) ГОСТ 19.701-90. – [Введен в действие с 1992-01-01]. – М. : Стандартинформ 2010. — 24 с. –(Государственный стандарт Союза ССР).

2. Пароннджанов В.Д. Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации / В.Д. Пароннджанов. –М.: ДМК Пресс, 2012. –520 с.

3. Профессиональное программное обеспечение для построения схем - Microsoft Visio [Електронний ресурс] / Режим доступу : http://office.microsoft.com/ru-ru/visio/ .

4. Microsoft Visio. Custom Patterns [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/office/aa200997(v=office.10).aspx .

5. Microsoft Visio. Custom Properties [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/office/aa200980(v=office.10).aspx .

6. Oliveira N. Domain-Specific Languages: A Theoretical Survey / N. Oliveira, M. J. V. Pereira, P. R. Henriques, D. da Cruz. –Proc. 3rd Int'l Conference Compilers, Programming Languages, Related Technologies and Applications (CoRTA-2009). –2009. –P. 35-46.

7. yEd Graph Editor [Електронний ресурс] / Режим доступу: http://www.yworks.com/en/ products_yed_applicationfeatures.html .

8. 'Palette' Tool Window [Електронний ресурс] / Режим доступу: http://yed.yworks.com/ support/manual/palette.html .

9. DIA diagram editor [Електронний ресурс] / Режим доступу: http://dia-installer.de/ .

10. DIA diagram manual: Chapter 11. Custom Shape Module [Електронний ресурс] / Режим доступу: http://dia-installer.de/doc/en/custom-shapes-chapter.html .

11. Modeling SDK for Visual Studio - Domain-Specific Languages [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/bb126259.aspx .

12. Validation in a Domain-Specific Language. [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/bb126413.aspx .

13. Кузнецов О.П. Дискретная математика для инженера. / О.П. Кузнецов, Г.М. Адельсон-Вельский. – 2-е изд. – М. : Энергоатомиздат, 1988. –480 с.

14. Новиков Ф.А. Автоматный метод определения проблемно-ориентированных языков. Ч. 1 / Ф.А. Новиков, У.Н. Тихонова // Информационно-управляющие системы. – 2009. – № 6. – С. 34-40.

15. Новиков Ф.А. Автоматный метод определения проблемно-ориентированных языков. Ч. 2 / Ф.А. Новиков, У.Н. Тихонова // ИУС. – 2010. – № 2. – С. 31-37

16. Хопкрофт Дж. Введение в теорию автоматов, языков и вычислений / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. –[2-е изд.]. – М. : Вильямс, 2002. –528 с.

17. Fowler M. Language Workbenches: The Killer-App for Domain Specific Languages? [Електронний ресурс] / Режим доступу: http://www.martinfowler.com/articles/languageWorkbench.html .

18. The Compiler Generator Coco/R [Електронний ресурс] / Режим доступу: http://www.ssw.uni-linz.ac.at/Coco/ .

19. Чен П. Модель "сущность-связь" - шаг к единому представлению о данных [Електронний ресурс] / П.-Ш.Чен // СУБД. – 1995. – №3. – Режим доступу: http://www.osp.ru/DBMS/1995/03/271.htm .

Надійшла до редколегії 22.04.2014

Рецензент: д-р техн. наук, проф. С.І. Гоменюк, Запорізький національний університет, Запоріжжя.

МЕТОД ОПРЕДЕЛЕНИЯ ГРАММАТИКИ ПРОБЛЕМНО-ОРИЕНТИРОВАННОГО ЯЗЫКА ДИАГРАММ

Н.Г. Кеберле, Д.В. Чесановський

Возможность создания синтаксически корректных диаграмм, заданных на некотором проблемно-ориентированном языке реализуется либо процедурными средствами – встраиванием в визуальные редакторы инструментов проверки корректности диаграммы, либо декларативными средствами - на уровне граматики языка диаграмм. В работе продложен метод декларативного определения граматики проблемно-ориентированного языка диаграмм. Показано применение метода для языка ER-диаграмм.

Ключевые слова: *диаграмма, проблемно-ориентированный язык диаграмм, контекстно-свободная грамматика, синтаксический анализатор.*

DOMAIN-SPECIFIC DIAGRAM LANGUAGE GRAMMAR DEFINITION METHOD

N.G. Keberle, D.V. Chesanovsky

Ability to create syntactically correct diagrams using some domain-specific diagram language is achieved either procedurally – via embedding of syntax checking means into the visual editors, or declaratively - defining a domain-specific diagram language grammar. The paper introduces one method of declarative definition of domain-specific diagram grammar. Shown is the applicability of the method to the language of ER diagrams.

Keywords: *diagram, domain-specific diagram language, context-free grammar, syntax analyzer.*