УДК 004.423

Буй Д. Б, д.ф.-м.н., проф.,
Колегаєв О. М., аспірант.

Формальні методи розробки програмного забезпечення Z, B, VDM: порівняльний аналіз

Київський національний університет імені Тараса Шевченка, 03187, м. Київ, пр-т. Глушкова 4д
e-mail: buy@unicyb.kiev.ua
e-mail: alexkolegaev@mail.ru

D. B. Buy, the doctor of physical and mathematical sciences, the professor,
O. M. Kolegaev, postgraduate.

Formal methods in software development Z, B and VDM: comparison analysis

Taras Shevchenko National University of Kyiv, 03187, Kyiv, Glushkova st., 4d,
e-mail: buy@unicyb.kiev.ua
e-mail: alexkolegaev@mail.ru

У статті представлений порівняльний аналіз найбільш поширених формальних методів розробки програмного забезпечення: Z, B методу та Vienna Development Method. Розглянуті основні властивості зазначених формальних методів, вказані їх загальні характеристики. Проведений порівняльний аналіз. Мета порівняльного аналізу: визначити можливості вищевказаних формальних методів для створення абстрактних моделей та відтранслювання їх до програмного коду, що задовольняє умовам узгодженості первинної абстрактної моделі. Також запропоновані можливі формули переходу між абстрактними моделями, що створені за допомогою нотацій B та VDM. На основі розглянутих формул зроблений висновок щодо взаємозалежності даних методів.

Ключові слова: формальні методи розробки програм, порівняльний аналіз, Z нотація, B метод, VDM.

This paper contains comparative analysis of the most common formal methods for software development: Z, B method and Vienna Development Method. Basic properties of these formal methods are considered, their general characteristics are given. This paper contains comparative analysis of VDM, B and Z methods. The purpose of the comparative analysis: identify opportunities of these formal methods for creation abstract models and translation them into code that satisfies the conditions of the consistency for abstract models. Differences between these methods are placed in three tables (general comparative, OOP support and programming tools for development process) Also some formulas for transition between abstract models created with the notation B and VDM are proposed. On the basis of these formulas conclusion about interdependence of formal methods are given.

Key Words: formal methods in software development, comparison analysis, Z notation, B method, VDM.

Статтю представив д.ф.-м.н., проф. А.В. Анісімов

Вступ

У відповідності до [1] формальні методи можна визначити як такі, що використовуються в розробці комп'ютерних програмних систем, та базуються на математичних методах, зокрема, численні предикатів та теорії множин. Формальні методи застосовуються для розробки програмних систем: від математично строгої моделі майбутньої програми до власне програмного коду.

Відповідно до класифікації Джона Рашбі (John Rushby) з точки зору застосування формалізації методи створення програм можна умовно розділити на чотири групи або рівня: з нульового по третій. Третій, найвищий рівень, передбачає використання повністю формалізованих мов специфікацій і дозволяє перевіряти всі програмні конструкції створюваної абстрактної моделі засобами математичної логіки. Усі докази на цьому рівні можуть бути проведені або перевірені автоматично за допомогою спеціальних програмних засобів. В статті розглянуто формальні методи, що належать саме до другого та третього рівнів.

Проходячи декілька етапів уточнення (refinement) абстрактна модель зазначених методів зберігає свою коректність та узгодженість (consistency). Наприклад, B

метод, описаний у [2], передбачає декілька стадій переходу від абстрактної моделі до програмної реалізації, наведених на рис.1.

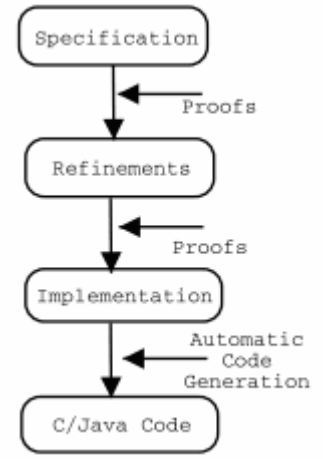

Рис. 1. Стадії переходу від абстрактної моделі до програмного коду

Слід зазначити, що використання формальних методів не гарантує оптимальності (по певному критерію) програмного коду або відсутності помилок в ньому [3]. Коректність визначається як відповідність обмежень (в іншій термінології – аксіом), заданих для початкової (абстрактної) моделі та програмної реалізації. Як зазначав E. Хехнер в [4] саме в цьому полягає одна з складнощів для застосування формальних методів у розробці прикладного програмного забезпечення.

Вибір для порівняння саме Z, B та VDM обумовлений можливостями їх використання в індустрії (проекти FeliCa Networks та TradeOne в Японії, Automatic Train Protection у Франції тощо).

Загальні характеристики Z, B, VDM

Z метод

Z нотація (Z notation) буда вперше запропонована Абріалем [5], пізніше доповнена Programming Research Group [6]. Основна структура, що використовується для створення Z-специфікацій, є схемою [7], яка складається з декларативної частини (об'явлення змінних) та предикативної частини (накладання обмежень на зазначені змінні). Зв'язок між іменами та типами змінних, вказаних в декларативній частині, задається за допомогою сигнатури. Наприклад [8]:

$$books : \mathrm{P}\ Book$$

$$person : \mathrm{P}\ People$$

$$lent\_to : \mathrm{P}\left(Book \times Person\right)$$

Загалом поняття схеми потрібно для визначення множини станів абстрактної моделі, перехід між якими здійснюється за допомогою операцій, що задаються як умови між початковим і кінцевими станами.

Одним з суттєвих недоліків Z нотації є складність її застосування в усталених програмних конструкціях подібно до мови охоронюваних команд Дейкстри [9]. Окрім цього, Z не містить в явному вигляді обробки виключень та, власне кажучи, нотація Z більше спрямована на розуміння моделі людиною, аніж на її програмну реалізацію. Зазначені недоліки були усунені Абріалем при розробці концепції B методу.

B метод

B метод є модельно-орієнтованим формальним методом, заснованим на численні предикатів та типізованій теорії множин, застосування і характеристики якого повністю наведені Абріалем в [2].

Семантика B визначається мовою GSL (Generalized Substitution Language). За допомогою цієї мови визначаються основні елементи мови машинних конструкцій AMN (Abstract Machine Notation), яка використовується для створення абстрактних B-моделей. Для кожної конструкції GSL Абріал вказує найслабшу передумову (користуючись термінологією Дейкстри). Наприклад, для конструкції $\mathrm{Q}\,|\,\mathrm{S}$, що виконує оператор S у разі істинності предиката Q (аналог охоронюваної команди у Дейкстри), має місце наступна найслабша передумова:

$$\left[\mathrm{Q}\,|\,\mathrm{S}\right]\mathrm{P} = \mathrm{Q} \wedge \left[\mathrm{S}\right]\mathrm{P}\,.$$

Зазначені абстрактні моделі, для запису яких використовується мова AMN, складаються с декількох частин, узгоджених між собою. Наприклад, початкові значення змінних, які задаються в INITIALIZATION, мають задовольняти обмеження, вказані в CONSTRAINTS (змінні, що передаються до моделі) та PROPERTIES (умови для констант).

Центральною ідеєю класичного B методу є концепція операції, виконання якої можливе лише за умови істинності обмежень абстрактної моделі та передумов самої операції. Розвиток цієї концепції виявив себе у відгалуженні Event-B [10].

Event-B, як і класичний B метод, призначений для створення та аналізу абстрактних моделей. Однак класичний і Event методи були розроблені для різних цілей: B метод призначений для створення коректних програмних систем, у той час як призначенням Event-B є моделювання повних систем (програмні системи разом з управлінням апаратною частиною). Event-B модель складається з контексту і машин. Контекст включає в себе статичну частину моделі: константи і аксіоми (описують властивості згаданих констант). Машини оперують динамічною частиною моделі, змінюючи її стан, що визначається за допомогою змінних. Змінні, як і константи, зіставляються з такими математичними об'єктами як множини, бінарні відношення, функції, числа і т.д. Вони обмежені умовами, що задаються в пункті INVARIANTS. Ці умови повинні бути дотримані для всіх наборів значень, які можуть набуватись змінними. Зміни стану моделі задаються переліком подій (event), кожна з яких складається з охорони і дії.

Застосування інваріантів, що склеюють (glueing invariants) [11], дозволяє переносити обмеження абстрактних моделей на їх деталізації. Кінцевою метою розробки постає створення доказаної моделі (proved model) [11], яка може бути втілена за допомогою обраної мови програмування (C, Java).

VDM

Vienna Development Method (VDM) [12, 13] є попередником як Z, так і B методів. У відповідності до [13] VDM визначається як сукупність технологій для моделювання та розробки програмних систем. Є декілька подвидів VDM, як, наприклад, VDM-SL або VDM++ (об'єктно-орієнтоване розширення VDM).

Функції VDM можуть бути визначені двома способами: як явним, так і неявним. Наприклад, явне визначення функції, яка кожному числу ставить у відповідність його абсолютне значення, матиме вигляд:

$$\text{abs} : Z \to N$$

$$\text{abs}(i)\ \Delta\ if\ \ i < 0 \text{ then } -i \text{ else } i$$

За допомогою перед- та післяумов можна задати неявну функцію. Наприклад, функція для знаходження максимального значення з масиву, елементами якого є натуральні числа:

$$\text{maxs}(s : N - set)\ r : N$$

$$\text{pre } s \neq \{\ \}$$

$$\text{post } r \in s \wedge \forall i \in s \cdot i \leq r$$

В перевірці істинності перед- та післяумов і полягає несуперечливість абстрактної моделі. Аналогічно до B методу визначене поняття уточнення (reification, [13, p. 179]), аналогом якої в B методі є поняття refinement.

Таблиця 1. Порівняння Z, B та VDM

Ознака порівняння	Z	B	VDM
Стиль формального методу	Модельно-орієнтований	Модельно-орієнтований	Модельно-орієнтований
Синтаксис	Використання замкнених структур – схем (schema), які включають до себе перелік змінних, предикатів, та операцій	Використання абстрактних машин, які записуються за допомогою ключових слів (PRE, THEN, INVARIANT тощо), що включаються до мови AMN	Використання ключових слів (pre, if, then тощо), які застосовуються для запису функцій (явно або неявно), перед- та післяумов і інваріантів
Визначення множини станів (перед і після операції)	Перед виконанням операції множина станів визначається набором вхідних (undecorated) змінних, після виконання операції – вихідних (primed) змінних [16, p. 98]		Перед виконанням: підключені (hooked) змінні; після виконання відключені (unhooked) змінні. Відключені змінні мають задовольняти післяумові [15]

Ідентифікація вхідних та вихідних даних	Змінні вхідних даних містять символ «?» наприкінці імені. Змінні вихідних даних містять символ «!» наприкінці імені.	Вхідні та вихідні параметри визначаються в заголовку операції за наступним синтаксисом: $Out <-- Name(Input)$, де *Name* позначає ім'я операції, а *Out* та *Input* – набори вихідних та вхідних даних	
Генерація програмного коду	Абстрактна модель не може бути автоматично переведена до програмного коду. Уточнення абстрактної моделі відбувається в полуавтоматичному режимі.		

Таблиця 2. Підтримка ООП

Z	B	VDM
Object Z підтримує поліморфізм, наслідування, інкапсуляцію [16]		VDM++ Підтримує поліморфізм, наслідування, інкапсуляцію [17]

Інструментарій для розробки програмних моделей:

Таблиця 3. Інструментарій розробки

Z	B	VDM
Z Word; Z/Eves Fastest	AteluerB ProB Rodin	SpecBox Overture VDM tools

Трансляція конструкцій VDM в B

Слід зазначити, що незважаючи на синтаксичні відмінності, абстрактні моделі, записані за допомогою одного з розглянутих вище формальних методів, можуть бути записані за допомогою конструкцій іншого методу. Для прикладу розглянемо перехід від VDM до B нотації. Зразок такого переходу для прикладу мережевого протоколу наведений в [18]. Розглянемо деякі аспекти такого переходу, для чого використаємо функцію наступного виду:

$\text{TranslationFunction} : [VDM\_Term] \to AMN\_Term$

Базовими типами VDM є наступні.

Базовий тип $= 'nat' | 'nat1' | 'int' | 'rat' | 'real' | 'char' | 'bool' | 'token'$

Зазначені типи являють собой множини, наприклад, тип *nat*1 позначає множину натуральних чисел, а тип *nat* доповнює тип *nat*1 числом 0.

Базовим типам VDM можна поставити у відповідність наступні типи даних B:

$Type[nat] \to N;$

$Type[nat1] \to N_1;$

$Type[char] \to STRING.$

Звісно, що наявна семантична різниця між типами в VDM та B. Зокрема, роль типів B методу відіграють множини, а всі множини в B скінченні. Типи bool та int, наявні в VDM, потребують об'явлення в бібліотеці машин:

$Type[\text{bool}] \to \text{INCLUDE } Bool\_Type$

$Type[\text{int}] \to \text{INCLUDE } Int\_Type$

Типи *rat* та *real* можуть бути представлені аналогічним чином. Синоніми, на зразок $Time = N$, можуть бути представлені за допомогою констант. Перераховні типи в VDM визначаються наступним чином:

$type\ definition = identifier, '=', name, \{| name\},$

де кожне ім'я є унікальним. Їх можна представити за допомогою множин-констант:

$TypeDef[name = t_1 | ... | t_k] \to$

MACHINE *name*'_*type*'

SETS

$t_1'\_set' = \{t_1\},$

...

$t_n'\_set' = \{t_n\},$

*name*_'*set*'

PROPERTIES

$name'\_set' = t_1'\_set' \cup ... \cup t_n'\_set' \wedge t_1 \neq t_2 ... t_{n-1} \neq t_n$

END

Наприклад, перераховний тип $\text{Result} = YES | NO$ буде відтрансльований наступним чином:

MACHINE Result_type

SETS

$Yes\_set = \{YES\},\ No\_set = \{NO\},$ Result_set

PROPERTIES

$\text{Result\_set} = Yes\_set \cup No\_set \ \wedge Yes \neq No$

END

Приклад трансляції конкретної абстрактної машини, створеної за допомогою VDM, в конструкції B методу вказана в [18].

Так як Z та B нотації мають одного розробника, то перехід між ними більш спрощений. Приклад такого переходу на основі B-ToolKit вказаний в [19].

Головні результати та висновки

Зроблений порівняльний аналіз дозволяє стверджувати, що формальні методи мають єдину структуру абстрактних моделей, що дозволяє здійснювати взаємні трансляції специфікацій. З трьох методів найбільші можливості для індустріального використання має B-метод (зокрема його відгалуження Event-B), тому розглянута можливість трансляції абстрактних моделей в AMN. З розглянутого випливає логічний зв'язок методів, представлений на рис.2.

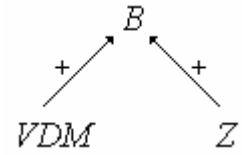

Рис. 2. Діаграма трансляцій специфікацій

Подальша робота полягає в уточненні базових понять B методу на основі композиційного підходу.

Список використаних джерел

1. *Morris D.* Concise encyclopedia of software engineering / D. Morris, B. Tamm. – Pergamon Press Ltd, 1993. – 400 p.
2. *Abrial J.-R.* The B Book: Assigning Programs to Meanings / J.-R. Abrial. – Cambridge University Press, 1996. – 816 p.
3. *Hall A.* Seven Myths of Formal Methods / A. Hall // IEEE Software. – 1990. – Vol. 5. – P. 11-19.
4. *Hehner E.* What's wrong with formal programming methods? / Eric C. R. Hehner // Advances in Computing and Information. – 1991. – Vol. 497. – P. 1-23.
5. *Abrial* J.-R. Data Semantics / J.-R. Abrial. – IFIP Working Conference Data Base Management – 1974. – P. 1-60.
6. *Spivey J.* The Z notation: a reference manual / J. M. Spivey. – Prentice Hall, 1992. – 150 p.
7. *McKeag R.* On the Construction of Programs / R. M. McKeag, A.M. MacNaghten. – Cambridge University Press, 1980. – 426 p.
8. *Wordsworth J.* Software Development With Z: A Practical Approach to Formal Methods in Software Engineering / J. Wordsworth – Addison-Wesley Pub, 1992. – 336 p.

References

1. MORRIS, D. (1993) Concise encyclopedia of software engineering. Pergamon Press Ltd.
2. ABRIAL, J. (1996) The B Book: Assigning Programs to Meanings. Cambridge University Press.
3. HALL, A. (1990) Seven Myths of Formal Methods. *IEEE Software*. 5. pp. 11-19.
4. HEHNER, E. (1991) What's wrong with formal programming methods? Advances in Computing and Information. 497. pp. 1-23.
5. ABRIAL, J. (1974) Data Semantics. IFIP Working Conference Data Base Management. pp.1-60.
6. SPIVEY, J. (1992) The Z notation: a reference manual. Prentice Hall.
7. MCKEAG, R. (1980) On the Construction of Programs. Cambridge University Press.
8. WORDSWORTH, J. (1992) Software Development With Z: A Practical Approach to Formal Methods in Software Engineering. Addison-Wesley Pub.
9. DIJKSTRA, E. (1976) A Discipline of Programming. Prentice-Hall.
10. ABRIAL, J. (2010) Modeling in Event-B System and Software Engineering. Cambridge University Press.
11. CANSELL, D. (2003) Foundations of the B method. Computing and Informatics.22. pp.221-256.

12. *Дейкстра* Э. Дисциплина программирования: [пер. с англ.] / Э. Дейкстра. – М.: Мир, 1978. – 274 с.
10. *Abrial* J.-R. Modeling in Event-B System and Software Engineering / J.-R. Abrial. – Cambridge University Press, 2010. – 612 p.
11. *Cansel D.l* Foundations of the B method / D. Cansell, M. Cansell // Computing and Informatics – 2003. – Vol. 22. – P. 221-256.
12. *Оллонгрен А.* Определения языков программирования интерпретирующими автоматами : [пер. с англ.] / А. Оллонгрен. – М.: Мир, 1977. – 289 с.
13. *Jones C.* Systematic Software Development Using VDM / C. Jones. – Prentice Hall, 1990. – 350 p.
14. *Fitzgerald J.* The Typed Logic of Partial Functions and the Vienna Development Method / J. Fitzgerald // Logics of Specification Languages. – 2008. – P. 453-487.
15. *Stolen K*. A method for the development of totally correct shared-state parallel programs / K. Stolen // 2nd International Conference on Concurrency Theory. – 1991. – Vol. 527. – P. 510-525.
16. *Graeme S.* The Object-Z Specification Language / S. Graeme. – Springer US. – 2000. – 146 p.
17. *Durr* E. VDM++, a formal specification language for object-oriented designs / E. Durr, J. Katwijk // Computer Systems and Software Engineering. – 1992. – P. 214-219.
18. *Bicarregui J.* Invariants, frames and postconditions: a comparison of the VDM and B notations / J. Bicarregui, B. Ritchie // Lecture Notes in Computer Science. – 2005. – Vol. 670. – P. 168-182.
19. *Hoare J.* Applying the B technologies to CICS / J. Hoare, Jeremy Dick, David Neilson, Ib Sorensen // Industrial benefit and advances in formal methods. – 1996. P. 74-84.

12. OLLONGREN, A (1974) Definition of programming languages by interpreting automata.Academic Press.
13. JONES, C. (1990) Systematic Software Development Using Vdm. Prentice Hall.
14. FITZGERALD, J. (2008) The Typed Logic of Partial Functions and the Vienna Development Method. Logics of Specification Languages. pp.453-487.
15. STOLEN, K. (1991) A method for the development of totally correct shared-state parallel programs. 2nd International Conference on Concurrency Theory. 527. pp.510-525.
16. GRAEME, S. (2000) The Object-Z Specification Language. Springer US.
17. 15 DURR, E. (1992) VDM++, a formal specification language for object-oriented designs. Computer Systems and Software Engineering. pp.214-219
18. BICARREGUI, J. (2005) Invariants, frames and postconditions: a comparison of the VDM and B notations. Lecture Notes in Computer Science. 670. pp.168-182
19. HOARE, J. (1996) Applying the B technologies to CICS. Industrial benefit and advances in formal methods. pp.74-84.

Надійшла до редакції 14.12.2015