УДК 004.424

Д. Б. БУЙ, А. М. КОЛЕГАЕВ

Киевский национальный университет имени Тараса Шевченко, Украина

ФОРМАЛЬНЫЕ МЕТОДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье рассматриваются основные концепции, которые легли в основу B метода и Event-B. Описан процесс построения абстрактных машин и определения их характеристик с использованием структур B метода. Рассмотрены два направления B метода: классический и Event-B. Указаны подходы различных авторов к проблемам детализации и завершимости абстрактных моделей. Указано программное обеспечение, с помощью которого можно создавать абстрактные машины в нотации AMN (Abstract Machine Notation).

Ключевые слова: *формальный метод разработки программного обеспечения, абстрактная машина, B метод, слабейшее предусловие, детализация, Event-B, AMN.*

Введение

Разработка современного программного обеспечения требует надежных методов контроля за корректностью алгоритма, положенного в основу программируемой конструкции. Такой контроль становится возможным благодаря применению формальных методов в программировании. В соответствии с классификацией Джона Рашби (John Rushby) [1] с точки зрения применения формализации методы создания программ можно условно разделить на четыре группы.

1. Нулевой уровень, на котором применение формальных методов не предусмотрено.

2. Первый уровень, предполагающий замену конструкций обычного языка, с помощью которого выражены общие требования к программе, терминами и обозначениями теории множеств, математической логики и т.д. Все доказательства и проверки на этом уровне проводятся вручную.

3. Второй уровень, на котором предполагается использование языков формальных спецификаций с поддержкой определенными средствами автоматизации (например, Z метод [2]).

4. Третий уровень, на котором используются полностью формализованные языки спецификаций, позволяет проверять все программные конструкции создаваемой абстрактной модели средствами математической логики. Все доказательства могут быть проведены или проверены автоматически с помощью специальных программных средств. Именно к третьему уровню относится B метод, разработчиком которого является Жан-Раймонд Абриал (Jean-Raymond Abrial) ([3]). Указанный метод применим ко всему циклу создания программы: от абстрактной модели, которая реализуется с помощью AMN (Abstract Machine Notation) до генерации программного кода, что обусловило популярность B метода и создания на его основе разнообразного программного обеспечения как для верификации абстрактных моделей, так и для их программной реализации.

1. Предпосылки создания B метода

Одна из первых попыток создания аксиоматики и правил проверки конструкций упрощенно-схематического алгоритмического языка была предпринята Хоаром (C.A.R. Hoare) в статье [4]. Там же было введено понятие, известное сейчас как «тройка Хоара», имеющая следующий вид: $P\{Q\}R$, где P (предусловие) и R (постусловие) являются предикатами, а Q – командой. Если команда Q выполняется при условии P, то предикат R обращается в истину после выполнения команды Q. Ставшее впоследствии классическим, понятие тройки Хоара применяется в формулировке предложенных Хоаром четырёх аксиом, одна из которых (аксиома присваивания) практически без изменений используется и в дальнейших формализмах: языке охраняемых команд Дейкстры [5] или B нотации Абриала [3]:

$$\vdash P < x \setminus E > \{x := E\} P, \qquad (1)$$

где запись $P < x \setminus E >$ означает замену всех вхождений переменной x в предикат P выражением E.

Однако как отмечал в [4] сам Хоар, один из недостатков его аксиоматического подхода состоял в вопросе возможного незавершения алгоритма, что проявлялось в аксиоме D3 (правило цикла).

Следующий шаг был сделан Дейкстрой в [5] и несколько уточнён Д. Грисом в [6]. Предложенная Дейкстрой модель «языка охраняемых команд» предусматривала введение понятия «слабейшего преду-

словия» (weakest precondition), которое практически без изменений перешло в B метод. Данный термин определял условие, характеризующее множество всех тех начальных состояний, для которых запуск команды обязательно приведёт к корректному завершению алгоритма, и при этом система перейдет в конечное состояние, удовлетворяющее заданному постусловию. Например, для конструкции присваивания данное предусловие определялось следующим образом:

$$\mathrm{wp}\left(\mathrm{x} := \mathrm{E}, \mathrm{Q}\right) = \mathrm{Q} < \mathrm{x} \setminus \mathrm{E} > . \qquad (2)$$

Или в более общей форме для группового оператора присваивания $\mathrm{x}_1,...,\mathrm{x}_n := \mathrm{E}_1,...,\mathrm{E}_n$, семантика которого состоит в том, что операторы $\mathrm{x}_1 := \mathrm{E}_1,...,\mathrm{x}_n := \mathrm{E}_n$ выполняются параллельно:

$$\begin{aligned}&\mathrm{wp}\left(\mathrm{x}_1,\mathrm{x}_2,...,\mathrm{x}_n := \mathrm{E}_1,\mathrm{E}_2,...,\mathrm{E}_n,\mathrm{Q}\right) = \\ &= \mathrm{Q} < \mathrm{x}_1,\mathrm{x}_2,...,\mathrm{x}_n \setminus \mathrm{E}_1,\mathrm{E}_2,...,\mathrm{E}_n > .\end{aligned} \qquad (3)$$

Несколько иная трактовка тройки Хоара $\mathrm{P}\{\mathrm{Q}\}\mathrm{R}$ и модели Дейкстры была предложена С.А. Абрамовым [7, 8]. В качестве P и Q Абрамов предложил рассматривать не предикаты на множестве состояний V, а бинарные отношения, являющиеся подмножествами $\mathrm{V}\times\mathrm{M}$, где M может быть произвольным множеством.

Другой подход быв предложен Морганом (C. Morgan) в [9]. Вместо определения семантики программных конструкций посредством слабейшего предусловия (как это было сделано Дейкстрой) он рассмотрел семантику в терминах предусловий, постусловий и структуры (frame). Запись $\omega:[\mathrm{P},\mathrm{Q}]$, которую использовал Морган, включала в себя список переменных ω (структуру), предусловие P и постусловие Q, которого необходимо достичь при условии истинности P. Критерием выполнимости (feasible) у Моргана служит условие $\omega:[\mathrm{pre},\mathrm{post}]$ являющееся истинным в том, и только в том случае, когда существует фрейм ω типа T, для которого истинно условие $\mathrm{pre}\Rrightarrow(\exists\omega:\mathrm{T}\cdot\mathrm{post})$. Запись $\mathrm{A}\Rrightarrow\mathrm{B}$ у Моргана означает следующее: если предикат A истинный, то предикат B также является истинным. Морган также ввёл понятие детализации (refinement, $\sqsubseteq$), предусматривающее два подхода: усиление постусловия и ослабление предусловия. Первый подход гласил, что если $\mathrm{post}'\Rrightarrow\mathrm{post}$, то $\omega:[\mathrm{pre},\mathrm{post}]\sqsubseteq\omega:[\mathrm{pre},\mathrm{post}']$. Ослабление предусловия означало, что если $\mathrm{pre}\Rrightarrow\mathrm{pre}'$, то $\omega:[\mathrm{pre},\mathrm{post}]\sqsubseteq\omega:[\mathrm{pre}',\mathrm{post}]$.

Само понятие детализации было введено (и сохранило аналогичный смысл в B методе) для установления связи между абстрактной моделью и программной реализацией алгоритма. По сути, переход от абстрактной модели S к реализации I представлял собой цепочку промежуточных детализаций R_i:

$$\mathrm{S}\sqsubseteq\mathrm{R}_1\sqsubseteq\mathrm{R}_2\sqsubseteq...\sqsubseteq\mathrm{R}_\mathrm{n}\sqsubseteq\mathrm{I}. \qquad (4)$$

В отличие от Моргана, использовавшего пары предикатов для формирования своего языка формальных спецификаций, подход, предложенный в [10] Хи Джифенгом (He Jifeng) и Хоаром предполагал использование отношений (relation) вида $(\alpha\mathrm{P},\mathrm{P})$, включавших в себя множество переменных $\alpha\mathrm{P}$ (алфавит), допустимых к использованию в предикате P. В трактовке Хоара и Джифенга алфавит представляется объединением множеств переменных, задающих начальное и конечное состояние модели: $\alpha\mathrm{P}=\mathrm{in}\alpha\mathrm{P}\cup\mathrm{out}\alpha\mathrm{P}$. Для рассмотрения вопроса завершимости программы вводится так называемая *структура* (design, $\mathrm{P}\vdash\mathrm{Q}$), определяемая как предикат, который может быть записан в следующей форме:

$$\mathrm{P}\wedge\mathrm{ok}\Rightarrow\mathrm{Q}\wedge\mathrm{ok}', \qquad (5)$$

где булевы переменные ok и ok′ соответствуют успешному старту и завершению программы. Последняя запись означает, что программа, запущенная при условии истинности посылки P, будет завершена в состоянии, удовлетворяющем истинность обязательства (commitment) Q. Авторы приводят четыре благоприятных условия (healthiness conditions), накладываемых на элементы структуры для гарантии успешного завершения алгоритма (см. [10], Appendix 3).

Изложенные выше идеи Моргана, Хоара и Дейкстры в несколько измененном виде воплотились в разработках Абриала по созданию Z [2] и B [3] нотаций.

2. B метод

В разработке современных абстрактных моделей ведущая роль принадлежит созданной Абриалом концепции B и Z методов. Помимо создания предусмотрены средства преобразования в программный код непротиворечивых абстрактных моделей. B метод основан на исчислении предикатов первого порядка и типизированной теории множеств ([3], Appendix C). Семантика B определяется языком GSL (Generalized Substitution Language). С помощью этого языка определяются основные элементы языка машинных конструкций AMN (Abstract Machine Notation), который используется для создания абстрактных B моделей. Для каждой конструкции GSL Абриал указывает наислабейшее предусловие (пользуясь терминологией Дейкстры). Например, для конструкции $\mathrm{Q}\,|\,\mathrm{S}$, выполняющей опе-

ратор S в случае истинности предиката Q (аналог охраняемой команды у Дейкстры), имеет место следующее наислабейшее предусловие:

$$[Q \mid S]P = Q \wedge [S]P. \qquad (6)$$

Указанная выше формула означает, что для того, чтобы конечное состояние обращало в истину предикат P, требуется истинность предусловия Q и, кроме того, выполнение оператора S должно гарантировать истинность P в достигнутой точке пространства состояний абстрактной модели. При этом соответствующая конструкция AMN такова: **PRE** P **THEN** S **END**. Полный список соответствий конструкций языков GSL и AMN вместе с наислабейшими предусловиями указан в [3] (Appendix C.12).

Схема разработки программной модели в B методе включает в себя следующие этапы: создание абстрактной модели или машины (abstract machine), её детализация (refinement) и трансляция в программный код на каком-либо языке программирования (например, CASE-система Atelier B предлагает трансляцию в язык C). Сама абстрактная модель имеет строго определенную структуру с заданными связями между элементами.

MACHINE $N(p)$
CONSTRAINTS C
SETS St
CONSTANTS k
PROPERTIES B
VARIABLES v
INVARIANT I
INITIALISATION T (7)
OPERATIONS
$y \leftarrow op(x) \mathrel{\hat{=}}$
PRE P
THEN S
END;
...
END

Пункт **MACHINE** содержит уникальное имя абстрактной модели. Здесь же можно указать список параметров, которые передаются данной модели. В пункте **CONSTRAINTS** предполагается размещение ограничений или значений параметров, которые передаются в данную модель. Раздел **SETS** содержит перечень множеств, предполагаемых к использованию. При этом допускается задание множеств без указания типов или задание множеств посредством перечисления элементов. Пункт **CONSTANTS** содержит список констант, ограничения и тип которых задаются в **PROPERTIES**. В пункте **VARIABLES** содержится перечень локальных переменных абстрактной модели, ограничения и тип которых указаны в пункте **INVARIANT**. Всем локальным переменным нужно присвоить начальные значения в пункте **INITIALISATION**. И, наконец, функции, исполняемые абстрактной моделью, указаны в разделе **OPERATIONS**. Каждая из операций имеет (опционально) предусловие, выраженное предикатом в пункте **PRE**.

Для каждой абстрактной модели необходимо доказать истинность определённых утверждений, истинность которых гарантирует согласованность (consistency) упомянутой модели. Например, для **CONSTRAINTS** имеем: $\exists p.C$, т.е. должны существовать значения параметров, удовлетворяющие предикатам **CONSTRAINTS**. Для **PROPERTIES** подобное ограничение имеет вид: $C \Rightarrow \exists St, k.B$ (в случае истинности предикатов **CONSTRAINTS** должен существовать набор множеств и констант, удовлетворяющий требованиям **PROPERTIES**); для **INVARIANT**: $B \wedge C \Rightarrow \exists v.I$ (т.е. при условии истинности предыдущих пунктов должен существовать хотя бы один набор переменных, удовлетворяющий требованиям **INVARIANT**); для **INITIALISATION**: $B \wedge C \Rightarrow [T]I$ (начальные значения переменных удовлетворяют требованиям **INVARIANT**); для каждой операции: $(B \wedge C \wedge I \wedge P) \Rightarrow [S]I$. Запись $[S]I$, как и ранее, обозначает наислабейшее предусловие.

Детализация абстрактной машины выполняется аналогично процессу, предложенному Морганом, т.е. осуществляется пошаговая замена конструкций AMN с той целью, чтобы полученная в итоге исполняемая (implementation) модель содержала лишь ограниченный набор операторов, а именно:

1. Присваивание $x := E$.
2. Последовательную композицию операторов $S;T$.
3. Условия вида **IF** E **THEN** S **ELSE** T **END**. Допускаются вариации, например, отсутствие **ELSE**.
4. Операторы выбора, использующие конструкцию **CASE**.
5. Конструкция цикла:
WHILE E **DO** S **INVARIANT** I
VARIANT v **END.**
6. Использование локальных переменных: **VAR** x **IN** S **END**.

Непосредственное задание абстрактных моделей и переход от абстрактных к алгоритмическим, а затем и программным структурам зависит от CASE-средств, поддерживающих B метод, напри-

мер, B-Toolkit, B4free, Atelier B и Rodin. Два последних средства из этого списка кроме классического B предполагают также создание так называемых Event-B моделей.

3. Event-B

Event-B, как и классический B метод, предназначен для создания и анализа абстрактных моделей. Однако классический и Event методы были разработаны для различных целей: B метод предназначен для создания корректных *программных* систем, в то время как предназначением Event-B является моделирование полных систем (программные системы вместе с управлением аппаратной частью). Event-B модель состоит из контекста и машин [11, 12]. Контекст включает в себя статическую часть модели: константы и аксиомы (описывающие свойства упомянутых констант). Машины располагают динамической частью модели, оперируя её состоянием, определяемым с помощью переменных. Переменные, как и константы, сопоставляются с такими математическими объектами как множества, бинарные отношения, функции, числа и т.д. Они ограничены условиями, задаваемыми в пункте **INVARIANTS.** Эти условия должны быть соблюдены для всех наборов значений, принимаемых переменными. Изменения состояния модели задаются перечнем событий (event), каждое из которых состоит из охраны и действия. Под охраной подразумевается условие, истинность которого позволяет перейти к действию.

Каждое событие E может быть выражено в таком виде:

$$E \triangleq \textbf{any } t \textbf{ where } P(t, v) \textbf{ then } S(t, v) \textbf{ end}, \quad (8)$$

где $P(t, v)$ – предикат (охрана); t – локальные параметры события, $S(t, v)$ – действие. Переменные, не являющиеся локальными, обозначены как v. Допускаются и сокращённые формы записи событий – например, отсутствие локальных переменных или охраны:

$$E \triangleq \textbf{when } P(v) \textbf{ then } S(v) \textbf{ end} \quad (9)$$

$$E \triangleq \textbf{begin } S(v) \textbf{ end}$$

Правила доказательства непротиворечивости моделей Event-B и их детализация указаны в [11, 12]. Для прикладной разработки, доказательства корректности и детализации Event-B систем используется CASE-средство Rodin [13, 14]. Примеры применения Rodin для моделирования программных систем приводятся в [15, 16].

Заключение

В статье рассмотрены основные принципы построения формальных моделей программных разработок. Указаны подходы различных авторов к проблемам детализации и завершимости абстрактных моделей. Особое внимание уделено современным методам формализации: B методу и Event-B, а также CASE-средствам поддержки упомянутых формальных методов разработки программного обеспечения.

Литература

1. Rushby, J. Formal Methods and the Certification of Critical Systems [Text] / J. Rushby // Computer Science Laboratory. Technical Report CSL-93-7. – 1993. – 319 p.

2. ISO/IEC standard 13568:2002. International Organization for Standartization [Text]. - Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics.

3. Abrial, J.-R. The B Book: Assigning Programs to Meanings [Text] / J.-R. Abrial // Cambridge University Press, 1996. – 816 p.

4. Hoare, C. A. R. An axiomatic basis for computer programming [Text] / C. A. R. Hoare // Communications of the ACM. – 1962. – P. 576 – 583.

5. Дейкстра, Э. Дисциплина программирования [Текст] : пер. с англ. / Э. Дейкстра. – М. : Мир, 1978. – 274 с .

6. Грис, Д. Наука программирования [Текст] : пер. с англ. / Д. Грис. – М. : Мир, 1984. – 416 с.

7. Абрамов, С. А. Анализ программ и бинарные отношения [Текст] / С. А. Абрамов // Вычисл. матем. и матем. физ. – 1983. – Т. 23, № 2. – С. 440 – 452.

8. Абрамов, С. А. Соотношения в множествах семантических отображений и бинарных отношений [Текст] / С. А. Абрамов // Вычисл. матем. и матем. физ. – 1982. – Т. 22, № 1. – С. 197 – 207.

9. Morgan, C. Programming from Specifications. Second Edition [Text] / C. Morgan. – Prentice Hall International, 1998. – 344 p.

10. Hoare, C. A. R. Unifying Theories of Programming [Text] / C. A. R. Hoare He Jifeng. – Prentice Hall International Series in Computer Science, 1998. – 320 p.

11. Abrial, J.-R. Modeling in Event-B System and Software Engineering [Text] / J.-R. Abrial // Cambridge University Press, 2010. – 612 p.

12. Abrial, J.-R. Refinement, Decomposition and Instantiation of Discrete Models: Application to Event-B [Text] / J.-R. Abrial, S. Hallerstede // Fundamenta Informaticae. – 2007. – Vol. 77. – P. 1 – 28.

13. Rodin: an open toolset for modelling and reasoning in Event-B [Text] / J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, L. Voisin // International Journal on Software Tools for Technology Transfer. – 2010. – Vol. 12. – P. 447 – 466.

14. An open extensible tool environment for Event-B [Text] / J.-R. Abrial, M. Butler, S. Hallerstede, L. Voisin // 8th International Conference on Formal Engineering Methods. – 2006. – P. 588 – 605.

15. Применение Event-B для создания систем на программируемой логике [Текст] / Ю. Прохорова, С. Остроумов, О. Трубицына, Л. Лайбинис // Радіоелектрон. і комп'ют. системи. – 2009. – № 6. – С. 245 – 250.

16. Formal Development and Quantitative Assessment of a Resilient Multi-robotic System [Text] / A. Tarasyuk, I. Pereverzeva, E. Troubitsyna, L. Laibinis // Software Engineering for Resilient Systems. – 2013. – P. 109 – 124.

Поступила в редакцию 24.02.2014, рассмотрена на редколлегии 24.03.2014

Рецензент: д-р техн. наук, проф., Б. М. Конорев, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Харьков, Украина.

ФОРМАЛЬНІ МЕТОДИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Д. Б. Буй, О. М. Колегаєв

У статті розглядаються загальні концепції, що лягли в основу B методу та Event-B. Описано процес побудови абстрактних машин та визначення їх характеристик з використанням структур B методу. Розглянуто два напрямки B методу: класичний та Event-B. Вказані підходи різних авторів до проблем деталізації і завершеності абстрактних моделей. Вказано програмне забезпечення, за допомогою якого можна створювати абстрактні моделі в нотації AMN (Abstract Machine Notation).

Ключові слова: формальний метод розробки програмного забезпечення, абстрактна машина, B метод, найслабкіша передумова, деталізація, Event-B, AMN.

FORMAL METHODS IN SOFTWARE DEVELOPMENT

D. B. Buy, A. M. Kolegaev

General conceptions and foundation of B method and Event-B are considered in the article. The process of creating abstract machines and determination of their properties using B method is described. Two trends of B method are considered: classic B and Event-B. Approaches of various authors to the problems of detail and completeness of abstract models are indicated .Attention is paid to the list of CASE tools for creating abstract models in AMN (Abstract Machine Notation).

Keywords: formal methods in software development, abstract machine, B method, weakest precondition, refinement, Event-B, AMN.

Буй Дмитрий Борисович – д-р физ.-мат. наук, профессор, профессор кафедры теории и технологии программирования факультета кибернетики Киевского национального университета им. Тараса Шевченко, Киев, Украина, e-mail: buy@unicyb.kiev.ua.

Колегаев Алексей Михайлович – аспирант кафедры теории и технологии программирования Киевского национального университета им. Тараса Шевченко, Киев, Украина, e-mail: alexkolegaev@mail.ru.