О.Г. СТАРУСЕВ, канд. техн. наук, НТУ "ХПИ" (г. Харьков)

ИСПОЛЬЗОВАНИЕ МЕТОДА ФОРМАЛЬНОЙ СПЕЦИФИКАЦИИ Z ДЛЯ ОПИСАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

У статті розглянуто підхід до опису інформаційної системи за допомогою одного з методів формальної специфікації – Z. Наведений опис методу та вказано його використання при описі реальної технічної системи, а також місце цього методу серед інших методів формального опису систем.

In this article the approach to information system description was considered with Z formal method specification using. In this article to give a method description and illustrate using it with real technical system description, and was methods place among all system formal specification methods set.

Постановка проблемы. Традиционные технические дисциплины обычно легко адаптируют лучшие математические методы. На сегодняшний день проведено достаточно много исследований по применению математических методов в процессе создания информационных систем (ИС) и программного обеспечения (ПО), но использование этих методов ограничено, так как многие специалисты не считают применение этих методов экономически выгодным [1].

Термин «формальные методы» подразумевает ряд операций, в состав которых входят создание формальной спецификации, анализ и доказательство спецификации, реализация системы на основе преобразования спецификации в программы и верификация программ. Все эти действия зависят от формальной спецификации, т.е. от системной спецификации, написанной на языке, словарь, синтаксис и семантика которого заданы формально. Формальное определение спецификации базируется на дискретной математике, в частности, на теории множеств и алгебре логики [2].

При проектировании ИС и ПО формальные методы могут применяться на трех уровнях:

1. Формальное описание. Формальная спецификация позволяет сделать спецификации более полными и однозначными, позволяют разрешить неоднозначности и структурировать как подход к проблеме, так и подход к реализации.
2. Формальная разработка и проверка. Полная разработка с использование формальных методов на сегодняшний день встречается крайне редко. Она предполагает формальную спецификацию системы, доказательство того, что система обладает необходимыми свойствами и не обладает нежелательными, а также выполнение уточняющих итераций, которые приводят систему из абстрактного вида к исполняемому коду.

3. Доказательства, проверяемые на ЭВМ. С появлением программного инструментария поддержки доказательства программ на непротиворечивость стала возможна автоматическая проверка непротиворечивости и обоснованности доказательств. Такая проверка особенно важна для систем с повышенными требованиями к безопасности. Такие требования выдвигаются большинством правительственных организаций, например Европейским космическим агентством.

Существует два основных подхода к разработке формальной спецификации, которые используются для написания детализированных спецификаций сложных информационных систем:

a) алгебраический подход, при котором система описывается в терминах операций и их отношений;
b) подход, ориентированный на моделирование, при котором модель ИС строится с использованием математических инструкций, таких, как множества и последовательности, а операции определяются тем, как они изменяют состояние системы.

Таким образом, необходимо провести анализ состояния вопроса и рассмотреть, какие конкретно методы формальных спецификаций предложены на сегодняшний день

Анализ литературы. В настоящее время создано достаточно много языков формальных спецификаций. Стоит отметить, что, к сожалению, в отечественной литературе вопрос отражен крайне слабо. Классификация языков формальных описаний согласно предложенному правилу следующая.

Алгебраические языки. Для описания последовательных систем используются такие языки как Larch [3] с компонентом Larch Prover, OBJ [4], а для описания параллельных (concurrent) систем язык LOTOS [5].

Языки, основанные на моделях. Для описания последовательных систем используются такие языки как семейство Z [6], VDM [7], B [8], а для описания параллельных систем (concurrent system) язык CSP/CCS [9] и сети Петри [10].

Наиболее перспективным из перечисленных методов, на наш взгляд, представляется метод Z. Во-первых, метод достаточно просто и вместе с тем обладает высокой гибкостью и мощностью, что подтверждают результаты его промышленного использования, например, для описания абонентской системы IBM CICS [11] и для создания узла вычислений с плавающей точкой Inmos для суперЭВМ Transputer T800 [12]. Во-вторых, метод достаточно легко комбинируется с привычными структурными методами, например SSADM или методом «чистой комнаты». В-третьих, для метода Z отработаны системы доказательств в средах EVES, HOL и OBJ, что позволяет сочетать легко читаемую нотацию Z с механической системой проверки доказательств и таким образом повысить надежность разработки.

Цель статьи. Целью этой статьи является получение формальной

спецификации ИС при помощи метода Z.

Использование метода Z для описания ИС. Алгебраические методы подходят для описания интерфейсов, когда операции, связанные с объектом, не зависят от состояния объекта. Тогда результаты любой операции не зависят от результатов предыдущей операции. Если эти предпосылки не выполняются, то необходимо вносить массу условий и ограничений, при этом алгебраические методы становятся громоздкими. Практика показывает, что алгебраические описания поведения систем в большинстве своем искусственны и трудны для понимания.

Альтернативным подходом к созданию формальных спецификаций является использование поведенческих спецификаций, которые используют модели состояний системы. Системные операции определяются посредством изменения состояний системной модели. Это и определяет поведение системы. Наиболее логичным и простым для понимания методом разработки формальных спецификаций систем является метод Z. Спецификации представляют собой неформальный текст, дополненный формальными описаниями. Формальная часть спецификации состоит из небольших блоков (схем), декомпозиционно связанных между собой. Схемы отделяются от остального текста спецификации, и используются для отображения переменных состояний, определения ограничений и операций над состояниями [13]. Общая структура схемы приведена в таблице 1. В методе также предусмотрены операции над схемами, в частности, для построения, переименования и сокрытия схем.

Таблица 1

Структура Z-схемы

Контейнер	Имя схемы
Содержимое:**N** Емкость:**N**	Сигнатура схемы
Содержимое ≤ емкость	Схемные предикаты

Сигнатура схемы определяет сущности, которые составляют состояние схемы, а схемные предикаты – набор условий, которые должны быть всегда истинны для всех сущностей. Если схема определяет операции, предикаты могут представлять пред- и постусловия для этих операций.

Для иллюстрации применения метода Z в разработке формального описания критической системы рассмотрим пример насоса, подающего лекарственный препарат инсулин в условиях отделения интенсивной терапии. У больных, страдающих сахарным диабетом, нарушен естественный метаболизм глюкозы в крови и им требуются инъекции инсулина с постоянным контролем уровня сахара в крови. Уровень сахара в крови проверяется через постоянные промежутки времени – 10 минут. Если при этом уровень сахара увеличился, то больному делается инъекция инсулина,

понижающая уровень сахара.

Прибор для подачи инсулина состоит из следующих частей: 1) набор игл; 2) датчик; 3) насос; 4) контроллер (позволяет не только включать/выключать систему, но и устанавливать дозу инсулина вручную); 5) устройство аварийной сигнализации; 6) индикаторы; 7) таймер.

Построение формальной спецификации для такой системы будет достаточно объемно потому, что необходимо учесть все возникающие аварийные ситуации. В этой статье приведены только некоторые из них.

Основная схема PUMP, моделирующая состояние насоса, приведена в таблице 2.

Таблица 2

Z-схема инсулинового насоса

PUMP	
reading?:**N**	Данные от датчика количества сахара в крови
dose, cumulative_dose:**N**	Доза инсулина и суммарная доза инсулина
capacity**:N**	Текущий объем инсулина в резервуаре
alarm!: {off,on}	Включение/выключение датчика аварии
pump!:**N**	Управляющие сигналы для насоса
display1!, display2!:**STRING**	Индикатор сообщений и введенной дозы инсулина
dose≤capacity ∧ dose≤5 ∧ cumulative_dose≤50 capacity≥40 ⇒ display1!=" " capacity≤39 ∧ capacity≥10 ⇒ display1!="Мало инсулина" capacity≤9 ⇒ alarm!=on ∧ display1!="Очень мало инсулина" r2=reading?	Схемные предикаты, которые должны быть всегда истинны

Работа инсулинового прибора заключается в определении каждые 10 минут количества сахара в крови пациента и введения дозы инсулина, если уровень сахара повышается. Количество инсулина, которое необходимо ввести, вычисляется согласно схеме Дозировка (DOSAGE), которая приведена в таблице 3. Схема Дозировка является дельта-схемой и полностью инкапсулируется в схему PUMP. При этом вводится новый набор величин, имена которых совпадают с именами данной схемы, но к ним добавляется апостроф. Так обозначаются значения переменных состояний, которые были изменены после выполнения операции. В таблице 3 дельта-схема DOSAGE вводит имена capacity', cumulative_dose', *r*0' и *r*1'.

Кроме того, необходимо разработать ряд схем, которые будут моделировать выходные данные инсулинового насоса. Для этого предлагается разработать дельта-схемы DISPLAY (модель индикаторов) и ALARM (условия активизации устройства аварийной сигнализации), которые приведены в таблице 4.

Во всех схемах предикаты должны быть согласованы, т.е. не должно

быть условий в одной схеме, которые противоречат предикатам в другой схеме. Если в спецификации замечены противоречия, то необходимо применить методы анализа Z-спецификаций. Кроме того, необходимо построить модель поведения системы во времени. Хотя лучше всего использовать неформальные описания или метод CCS.

Таблица 3

Z-схема дозировки инсулина

DOSAGE
ΔPUMP
(
dose = 0 ∧ (
((r1≥r0) ∧ (r2=r1)) ∨
((r1>r0) ∧ (r2≤r1)) ∨
((r1<r0) ∧ ((r1-r2) > (r0-r1)))) ∨
dose = 4 ∧ (
((r1≤r0) ∧ (r2=r1)) ∨
((r1<r0) ∧ ((r1-r2) > (r0-r1)))) ∨
dose = (r2 – r1) * 4 ∧ (
((r1≤r0) ∧ (r2>r1)) ∨
((r1>r0) ∧ ((r2-r1)≥(r1-r0)))))
capacity’ = capacity – dose
cumulative_dose’ = cumulative_dose + dose
r0’ = r1 ∧ r1’ = r2

Таблица 4

Z-схема выходных данных

DISPLAY
ΔPUMP
display2’ = NaturalToString(dose) ∧
(reading? < 3 ⇒ display1!’ = «Сахар низкий» ∨
reading? > 30 ⇒ display1!’ = «Сахар высокий» ∨
reading? ≥ 3 and reading? ≤ 30 ⇒ display1!’ = «Ok»)

Z-схема включения сигнала тревоги

ALARM
ΔPUMP
(reading? < 3 ∨ reading? > 30) ⇒ alarm!’ = on ∨
(reading? ≥ 3 ∨ reading? ≤ 30) ⇒ alarm!’ = off

Ко всему написанному выше стоит добавить ряд соображений, связанных с методикой применения выбранного формального метода. Любой формальный метод дорог при интенсивном, а также первом применении. Если учесть начальные затраты на услуги консультирующих организаций и ПО, формальные методы, и в частности Z, обходятся также, как и проекты,

разработанные с использованием привычных подходов. Для большей гибкости разработки необходимо комбинировать формальные методы и традиционные подходы.

Стоит указать и на то, что формальные методы не являются каким-то «сверхподходом» или панацеей. При правильном применении и соблюдении стандартов качества формальные методы позволяют получить описания с высокой степенью целостности. Для этого необходимо правильно оценить необходимый уровень абстракции и воспользоваться корректной моделью жизненного цикла разработки (например, «спиральной» моделью Бёма).

Обязательным шагом в этом случае является дальнейшее применение формальных методов на этапе тестирования. Только постоянный контроль за итерационным процессом разработки позволит избежать ошибок и неточностей, даже если вся разработка базируется полностью на формальных методах.

Выводы. Основным преимуществом использования метода формальной спецификации является возможность выявления противоречий в системных требованиях. Метод формальной спецификации Z представляется наиболее перспективным для этих целей. Это достаточно критично в тех случаях, когда безопасность, отказоустойчивость и защищенность системы являются приоритетными при разработке ИС. При этом выявление ошибок и противоречий в системных требованиях возможно на ранних стадиях разработки ИС и ПО. Исправление ошибок на этих стадиях гораздо дешевле, чем внесение изменений в законченную систему. Также необходимо отметить и то, что формальные спецификации достаточно сложны для прочтения и понимания.

Список литературы: **1.** *Bowen J.P.* Towards Verified Systems. – Elsevier, Real-Time Safety-Critical System series, 1994. – 318 p. **2.** *Рейдуорд-Смит Д.* Теория формальных языков. – М.: Радио и связь. – 1995. – 387 с. **3.** *Wing J.M.* Writing Larch Interface Language Specifications. – Mass. Institute of Technology, 1990. – 611 p. **4**. *Goguen J.A., Winkler T.* Introducing OBJ3. Technical Report SRI-CLS-88-9, SRI, Menlo Park, USA, 1988. – 820 p. **5.** *Bolognesi T., Lagemaat J. V.D., Vissers C.A.* LOTOSphere: Software development with Lotos. – Kluwer Academic Press, 1994. **6.** *Spivey J.M.* The Z Notation: A reference manual. – Prentice Hall International Series of Computer Science, 1989. **7.** *Charatan Q., Kans A.* Formal Software Development. – Palgrave McMillan, 2003. – 256 p. **8.** *Abrial J.R.* The B Book – Assigning Program to Meaning. – Cambridge University Press, 2002. – 926 p. **9.** *Hoare C.A.R.* Communicating Sequential Processes. – Prentice Hall International Series of Computer Science, 1985. **10.** *Питерсон Дж.* Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 284 с. **11.** *Houston I.S.C., King S.* CICS Project Report: Experience and Results from the Use of Z in IBM // Prehn S., Toetenel W.J. VDM'91: Formal Software Development Method. – Springer-Verlag, LNCS 551, 1991. – P. 588-596. **12.** *Phillips M.* CICS / ESA 3.1 Experience // Nicholls J.E. Z User workshop. – Oxford: Springer-Verlag, 1989. – P. 179-185. **13.** *ISO/IEC 13568:2002* Information technology – Z formal specification notation – Syntax, type system and semantic.

Поступила в редакцию 15.03.04