УДК 519.681.2

А. В. Копыльцов, В. В. Цехановский
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Применение теории динамических систем для оценивания качества программного обеспечения

Предлагается подход оценивания качества программного обеспечения с помощью средств теории динамических систем. Если программы, созданные с помощью алгоритмических языков программирования, рассматривать как динамические системы, то можно оценить информационную энтропию и фрактальную размерность траектории в фазовом пространстве переменных программы. Фрактальную размерность и энтропию можно рассматривать как показатели качества, как меру сложности программного продукта. Если на основе одного и того же алгоритма написано несколько программ, то, определив их информационные энтропии и фрактальные размерности, можно эти программы сравнить между собой и определить, какая из программ, реализующих алгоритм, сложнее. Этот подход может быть распространен на оценивание качества как программных, так и аппаратных средств, а также языков программирования и других средств программного обеспечения.

Качество программного обеспечения, динамическая система, фрактальная размерность, информационная энтропия

В последнее время стоимость программного обеспечения (ПО) значительно больше стоимости компьютеров. Требования к качеству ПО также возрастают и предлагаются различные подходы для оценивания и особенно для обеспечения качества ПО [1]–[12]. Однако качество ПО неразрывно связано с качеством аппаратных средств. Поэтому предлагается подход на основе теории динамических систем, который позволит связать качество аппаратных и программных средств с качеством языков программирования и других средств.

Динамические системы. К фон-неймановским или алгоритмическим языкам можно отнести Фортран, Кобол, ПЛ/1, Алгол, Паскаль, Си, Ада и другие менее распространенные языки. Суть алгоритмического программирования можно свести к двум основным понятиям: оператор присваивания и передача управления [13]. Пусть программа написана на каком-нибудь алгоритмическом языке и осуществляет вычисления на конечном подмножестве M области вещественных чисел. Пусть количество элементов множества M равно MM. Пронумеруем множество M в порядке возрастания чисел номерами от 1 до MM. Если вещественное число x принадлежит M, то его номер равен N_x. Если в программе содержится одна вещественная переменная, то оператор присваивания A, заданный на множестве M, действует следующим образом: $A{:}x1 \to x2$, т. е. переводит вещественное число $x1$ в вещественное число $x2$, причем $x1$ и $x2$ принадлежат M.

Множество всех операторов присваивания $\{A_i\}$, заданных на множестве M мощности MM, допускает разбиение на классы эквивалентности. Два оператора A_x: $x1 \to x2$ и A_y: $y1 \to y2$ принадлежат одному классу, если $N_{x2} - N_{x1}$ равно $N_{y2} - N_{y1}$ по модулю MM, т. е. $N_{x2} - N_{x1} = N_{y2} - N_{y1} \pmod{MM}$. Очевидно, что число классов равно MM.

Пусть NM – максимальное число, используемое на компьютере. Рассмотрим пространство с мерой (M, σ, μ), где M – пространство, элементами которого являются вещественные числа, используемые на компьютере, т. е. $M = \{-NM, ..., 0, ..., NM\}$, σ есть σ-алгебра всех подмножеств M, μ – мера, нормированная на M, т. е. $\mu(M) = 1$ и все точки M равновероятны.

Пусть T – циклическая подстановка на M. Тогда T – эргодическое, не перемешивающее преобразование, и T порождает однопараметрическую группу автоморфизмов $\{T^t\}$ пространства с мерой (M, σ, μ), т. е. динамическую систему на (M, σ, μ) [14], [15]. Таким образом, можно утверждать, что для множества операторов присваивания $\{A_i\}$, заданных на множестве M, воз-

можно разбиение на классы эквивалентности. На этих классах эквивалентности может быть определена групповая структура, т. е. однопараметрическая группа автоморфизмов или, иными словами, динамическая система, которая является эргодической, но не перемешивающей [3], [14], [15].

Если количество переменных N в программе больше одного, то предыдущее утверждение можно переформулировать следующим образом. Если количество переменных N в программе больше одного, то множество операторов присваивания, которые заданы на множестве $M \times \ldots \times M$ (N сомножителей), можно разбить на классы эквивалентности. На этих классах можно определить динамическую систему, которая определяется как прямое произведение динамических систем, каждая из которых действует на пространстве (M, σ, μ). Эта динамическая система является эргодической, но не перемешивающей [3], [14], [15].

Поскольку программу рассматриваем как последовательность операторов присваивания и управления [13], то имеем следующие следствия. Во-первых, каждому прогону программы соответствует единственный автоморфизм пространства (M, σ, μ) и, во-вторых, каждому автоморфизму пространства (M, σ, μ) соответствует некоторое число прогонов программы, которое является не более чем счетным.

Из изложенного следует, что программы можно рассматривать как динамические системы и, таким образом, для анализа ПО могут быть применены методы и результаты теории динамических систем.

Применение теории динамических систем в программировании позволяет, в частности, осуществлять тестирование программ. Для анализа программ, рассматриваемых как динамические системы, используем методы спектрального анализа. Каждому прогону программы сопоставляется траектория в N-мерном фазовом пространстве переменных программы. С помощью методов спектрального анализа можно выявлять периодические, квазипериодические, хаотические и иные особенности поведения траектории в N-мерном фазовом пространстве переменных. При анализе траекторий часто используют фрактальные размерности, в частности, емкостную, поточечную, корреляционную и информационную [16].

Информационная размерность и энтропия. Рассмотрим информационную размерность. Строим покрытие дискретной траектории N-мерными кубами с ребром длины ε. Находим число точек N_i в каждой из N_1 ячеек покрытия и оцениваем вероятность P найти точку в i-й ячейке

$$P_i = \frac{N_i}{N_0},$$

где N_0 – общее число точек на траектории, $N_0 \neq N_1$, сумма всех P равна 1. Важным показателем является информационная энтропия

$$I(\varepsilon) = -\sum_{i=1}^{N} P_i \log_2 P_i,$$

где $I(\varepsilon)$ измеряется в битах. Информационная размерность

$$d = \lim_{\varepsilon \to 0} \frac{I(\varepsilon)}{\log_2 (1/\varepsilon)}.$$

В качестве показателей качества программного продукта (ПП) можно использовать фрактальные размерности и информационную энтропию. Фрактальная размерность может рассматриваться как степень сложности программы, т. е., чем сложнее программа, тем размерность больше [3], [17]. С помощью фрактальной размерности можно ввести отношения порядка (больше (>), меньше (<)) на множестве всех программ, т. е., если имеются программы A и B, то A < B, если фрактальная размерность A меньше фрактальной размерности B. Этот подход позволяет классифицировать программы. Например, к первому классу можно отнести программы с фрактальной размерностью от 0 до 1, ко второму классу – от 1 до 2, к третьему – от 2 до 3 и т. д.

Информационную энтропию, измеряемую в битах, тоже можно рассматривать как меру сложности программы. Расчет таких характеристик, как фрактальная размерность и энтропия, может осуществляться на аттракторах (предельных множествах в фазовом пространстве), или, если не рассматривать предельные множества, на обработке достаточно длинной конкретной временной реализации анализируемой программы [18].

Если две разные программы написаны по одному и тому же алгоритму, то, определив их фрактальные размерности и энтропии, можно их между собой сравнить, т. е. определить, какая из двух реализаций алгоритма более сложная (фрактальная размерность больше). Таким образом, фрактальная размерность и энтропия могут быть рассмотрены как показатели качества программного продукта.

Информационную энтропию *IE* можно рассматривать как меру количества случайности в эксперименте, состоящем в однократном прогоне программы, и называемую энтропией этого эксперимента. Информационная энтропия *IE* измеряет также количество неопределенности, содержащейся в этом эксперименте, т. е. количество неопределенности до прогона программы относительно того, каков будет ее результат [18].

Поскольку при определении фрактальных размерностей d нужно брать предел при N_0 и ε, стремящихся к бесконечности и нулю соответственно, а N_0 в программах обычно конечная величина, то использовать d нужно осторожно (только при достаточно больших N – порядка тысячи и более). Информационная энтропия *IE*, как показали расчеты, отличается большей стабильностью при варьировании входных параметров, чем фрактальные размерности. Поэтому при малых значениях N_0 использование *IE* более надежно, чем *d*. Если известна область изменения входных параметров программы, то можно сделать случайную выборку параметров из этой области и усреднить *IE* по области изменения параметров. Таким образом, можно получить усредненное значение *IE*.

Расчеты были проведены для программы, осуществляющей решение системы n алгебраических уравнений по методу Гаусса (для n равного 2 и 3) [19]. Так, для системы двух уравнений вида

$$a_1x + b_1y = c_1,\ a_2x + b_2y = c_2$$

имеем 22 вещественных переменных и 74 точки в 22-мерном фазовом пространстве переменных программы. Поскольку число точек в фазовом пространстве невелико (равно 74, что меньше тысячи), то использовать фрактальную размерность как качественный показатель нежелательно (хотя программа позволяет ее оценить, но доверять ей не следует). Поэтому использовали информационную энтропию *IE*, которая составила около 5 бит при $a_1 = 2$, $a_2 = 1$, $b_1 = 1$, $b_2 = –2$, $c_1 = 9$, $c_2 = –2$. При других значениях входных параметров получаются близкие по величине значения *IE* и составляет 4…5 бит. Аналогично в случае системы 3-х уравнений:

$$a_1x + b_1y + c_1 = d_1,$$
$$a_2x + b_2y + c_2 = d_2,$$
$$a_3x + b_3y + c_3 = d_3,$$

количество переменных в программе равно 39, а количество точек в 39-мерном фазовом пространстве составляет 192, что также меньше тысячи. Поэтому по тем же причинам производилась оценка *IE*, которая составила около 7 бит при $a_1 = 7$, $a_2 = 6$, $a_3 = 8$, $b_1 = 6$, $b_2 = 3$, $b_3 = 6$, $c_1 = 8$, $c_2 = 7$, $c_3 = 7$, $d_1 = 14$, $d_2 = 3$, $d_3 = 12$. При других значениях коэффициентов получаются близкие значения, и *IE* составляет 6…7 бит. Таким образом, *IE* для системы трех алгебраических уравнений больше, чем для системы двух уравнений и, следовательно, первая программа сложнее, чем вторая.

Аппаратное обеспечение. Заключение о сложности (возможностях) компьютера можно сделать по прогонам программы и найти важный показатель качества компьютера (аппаратного обеспечения) с точки зрения программирования. Пусть R – множество всех вещественных чисел – и есть алгоритм, который реализован на двух компьютерах C_1 и C_2 в виде двух одинаковых программ PR_1 и PR_2, p – количество переменных в программах и $R^p = R \times R \times ... \times R$ (p сомножителей). Тогда программы PR_1 и PR_2 можно рассматривать как отображения PR_1: $R^P \to R^P$, $PR2$: $R^P \to R^P$. Пусть M – множество вещественных чисел, равное объединению множеств вещественных чисел, которые используются на компьютерах. Пусть количество элементов в множестве M равно N. Рассмотрим пространство с мерой (M, σ, μ), где M – пространство, элементами которого являются вещественные числа, используемые на компьютере, т. е. $M = \{–NM, ..., 0, ..., NM\}$; σ есть σ-алгебра всех подмножеств M; μ – мера нормированная на M, т. е. $\mu(M) = 1$ и все точки M равновероятны.

Преобразуем M следующим образом. Вещественную ось сожмем до выпуклой оболочки множества, состоящего из объединения множества M и точки A, причем расстояние между M и точкой A равно минимальному вещественному числу, представимому на компьютерах. Пусть M^* есть объединение M и этой точки. Рассмотрим вместо отображения PR: $M \to M$ расширение этого отображения i_1: $M^* \to R$, i_2: $R \to M^*$, где отображение i_1 задается программой, а i_2 действует таким образом, что, если его образ содержится в M, то отображение не изменяется, а если вне M, то отображается в выбранную точку A. Таким образом, вместо отображения $M \to M$ рассматриваем отображение $M^* \to M^*$. Если рассмотреть последовательность отображений

$$PR_1(M^*), PR_1(PR_1(M^*)), ..., PR_1^j(M^*), ...,$$
$$PR_2(M^*), PR_2(PR_2(M^*)), ..., PR_2^j(M^*), ...,$$

то после каждого отображения количество оставшихся точек не возрастает, т. е., если ввести меру μ*, такую, что μ*(*M**) = 1, то $\mu^*(PR_j^i(M^*)) \geq \mu^*(PR_j^{i+1}(M^*))$ для всех $i > 0$, $j = 1, 2$.

Поскольку M конечно, то существуют числа j_1 и j_2 такие, что

$$\mu_1 = \mu^*(PR_1^{j_1}(M^*)) \geq \mu^*(PR_1^{jj}(M^*)) \text{ для } jj > j_1,$$
$$\mu_2 = \mu^*(PR_2^{j_1}(M^*)) \geq \mu^*(PR_2^{jj}(M^*)) \text{ для } jj > j_2.$$

Числа j_1, j_2 и μ_1, μ_2 являются характеристиками компьютеров C_1 и C_2, а не характеристиками алгоритма или программы (алгоритм и программа одинаковые). Различия между j_1 и j_2 или μ_1 и μ_2 могут быть объяснены, в частности, тем, что разрядная сетка и максимальное вещественное число, представимое на компьютерах, не одинаковые. Что касается μ_1 и μ_2, то ввиду больших затрат машинного времени реально их оценить практически трудно. Величины j_1 и j_2 оцениваются легче, так как программы одинаковые и, если взять одинаковые входные данные, то можно получить величины j_1 и j_2. Таким образом, j_1, j_2 или μ_1, μ_2 могут быть рассмотрены как показатели качества аппаратного обеспечения с точки зрения программирования (программы и входные данные одинаковые). Если $\mu_1 > \mu_2$ (или $j_1 > j_2$), то это означает, что сложность (возможности) машины C_1 больше, чем C_2.

В качестве примера использовалась программа вычисления корня n-й степени из комплексного числа K [19]. Расчеты проводились на двух разных компьютерах и показали, что в случае $K = 8$ для первого компьютера j_1 равно 26 для $n = 3$ и 13 для $n = 10$, а для второго j_2 равно 34 и 17 соответственно. Таким образом, $j_1 < j_2$, т. е. сложность (возможности) второго компьютера больше, чем первого.

Подход оценивания качества программного обеспечения с точки зрения теории динамических систем перспективен. Взяв за основу этот подход, в дальнейшей работе предполагается связать качество аппаратных и программных средств с качеством языков программирования, а также с качественными показателями человека, проектирующего, разрабатывающего и сопровождающего ПО. В итоге этот подход позволит связать в одно целое производителя (человека), средства производства (компьютеры, языки программирования и т. д.) и продукт труда (ПО) и на основе этого дать объективную оценку качества ПО в процессе как его разработки, так и эксплуатации.

СПИСОК ЛИТЕРАТУРЫ

1. Воробьев В. И., Копыльцов А. В., Юсупов Р. М. К оценке надежности программных средств // 3-й Всесоюзн. семинар «Качество программного обеспечения». Дагомыс, 1991. С. 120–121.

2. Оптимальный метод предварительной оценки качества программного продукта / А. О. Андреев, В. И. Воробьев, А. В. Копыльцов, Б. П. Пальчун // 1-я Межгор. конф. «Надежность, живучесть и безопасность технических систем» / ЛДНТП. СПб., 1992. С. 33–36.

3. Методы и модели оценивания качества программного обеспечения / В. И. Воробьев, А. В. Копыльцов, Б. П. Пальчун, Р. М. Юсупов; ЛДНТП. СПб., 1992. 34 с.

4. Копыльцов А. В. Оценка качества программного обеспечения с учетом человеческого фактора // 4-я Междунар. конф. «Качество программного обеспечения информационных технологий». Дагомыс, 1992. С. 25–27.

5. Kopyltsov A. V., Boyko A. V. Human factor under creation and exploitation of information technology software // Proc. of Intern. Conf. on Informational Technology and People (ITAP 93). Moscow, 1993. Part 2. P. 121–124.

6. Kopyltsov A. V. Human factor in software quality estimation // Proc. of Intern. Conf. on CAD\CAM, Robotics and Factories of the Future. Nauka Publ. St. Petersburg, 1993. Vol. 2. P. 539–542.

7. Kopyltsov A. V., Vorobiev V. I. Mathematical modeling of software quality estimation // 2-я Санкт-Петерб. междунар. конф. «Региональная информатика-93». СПб., 1993. Ч. 1. С. 120.

8. Копыльцов А. В. Теоретическое моделирование оценки качества программных продуктов // Надежность, отказоустойчивость и производительность информационных систем: сб. тез. Межгор. науч.-техн. семинара. Туапсе, 1993. С. 55.

9. Kopyltsov A. V., Vorobiev V. I. On a software quality estimation // Intern. Congress on Computer Systems and Applied Mathematics. Abstract. St. Petersburg, 1993. P. 178–179.

10. Копыльцов А. В. Об оценке качества программного обеспечения // 3-я Санкт-Петерб. междунар. конф. «Региональная информатика-94». СПб., 1994. С. 127.

11. Kopyltsov A. V. On software quality estimation // 3-я Санкт-Петерб. междунар. конф. «Региональная информатика-94». СПб., 1994. С. 181–185.

12. Копыльцов А. В. Об оценке качества программного обеспечения // Проблемы информатизации (теоретич. и науч.-практ. журн.). СПб., 1994. Вып. 3–4. С. 46–49.

13. Валиев М. К., Кругляков С. В., Юрченко А. А. Функциональное программирование. М.: Знание, 1989.

14. Billingsley P. Ergodic Theory and Information. New York: John Wiley and Sons, 1965.

15. Корнфельд И. П., Синай Я. Г., Фомин С. В. Эргодическая теория. М.: Наука, 1980.

16. Moon P. O. Chaotic Vibrations. Inc. New York: John Wiley and Sons, 1987.

17. Колмогоров А. Н. Алгоритм, информация, сложность. М.: Знание, 1991.

18. Grassberger P., Proccacia I. Characterization of Strange Attractors // Phys. Rew. Lett. 1983. Vol. 50. P. 346–349.

19. Агеев М. И., Алик В. П., Марков Ю. И. Библиотека алгоритмов. М.: Сов. радио, 1976.

A. V. Kopyltsov, V. V. Tsehanovsky
Saint Petersburg Electrotechnical University «LETI»

APPLICATION OF THE THEORY OF DYNAMIC SYSTEMS FOR SOFTWARE QUALITY ESTIMATION

An approach for software quality estimation by using the tools of the theory of dynamic systems is proposed. If the programs created using algorithmic programming languages are considered to be dynamic systems, then it is possible to estimate the informational entropy and the fractal dimension of the trajectory in the phase space of the program variables. The fractal dimension and entropy can be taken as indicators of quality, as a measure of complexity of a software product. If there are several programs written on the basis of the same algorithm, the values of their informational entropies and fractal dimensions can be used to compare the programs and find out which of the programs implementing the algorithm is more complicated. This approach can be extended to assess the quality of software and hardware, as well as programming languages and other software tools.

Software quality, dynamic system, fractal dimension, informational entropy

УДК 681.5

Ю. В. Ильюшин
Санкт-Петербургский горный университет

М. Ю. Шестопалов
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Применение модифицированного критерия Найквиста для анализа импульсных распределенных систем

Классические результаты теории автоматического управления получены в большинстве случаев применительно к системам с сосредоточенными параметрами, поведение которых однозначно характеризуется изменением управляемых величин только во времени и описывается чаще всего обыкновенными дифференциальными уравнениями, что не дает возможности расширения области применения таких систем. В статье рассматривается модификация критерия абсолютной устойчивости Найквиста для применения его в теории систем с распределенными параметрами. Исследованы предельные характеристики параметров, влияющих на вид и форму пространственных годографов типовых распределенных звеньев. Разработан модифицированный критерий абсолютной устойчивости нелинейных распределенных систем управления. Разработан метод анализа абсолютной устойчивости класса нелинейных распределенных систем управления. На примере рассмотрено построение пространственно-распределенного годографа и построена область устойчивости рассматриваемой системы.

Системный анализ, управление, распределенные системы, абсолютная устойчивость

В современных условиях динамически развивающегося рынка минерально-сырьевого сектора все большую роль играет возможность снижения себестоимости продукции. Добыча нефти является одной из таких отраслей. Системы автоматизации такого технологического процесса создава-