Для цитирования: Парамонов Н.Б., Минин И.В., Янко Д.В., Шаменков Н.А. Проверка функционирования программных комплексов при переходе на новую программно-аппаратную платформу // Вопросы радиоэлектроники. 2017. № 3. С. 100–103.
УДК 681.3

Н.Б. Парамонов[1], И.В. Минин[2], Д.В. Янко[2], Н.А. Шаменков[3]

[1] ПАО «ИНЭУМ им. И.С. Брука», [2] 432 ВП МО, [3] НИИЦ ЦНИИ ВВКО МО

ПРОВЕРКА ФУНКЦИОНИРОВАНИЯ ПРОГРАММНЫХ КОМПЛЕКСОВ ПРИ ПЕРЕХОДЕ НА НОВУЮ ПРОГРАММНО-АППАРАТНУЮ ПЛАТФОРМУ

Рассматриваются подходы к тестированию программного обеспечения на основе формализаций в виде алгебры алгоритмов В.М. Глушкова. Показана применимость такого подхода для случая использования унаследованных программ. Определено множество обоснованных проверок при построении тестов проверок заданного предиката активизации программного комплекса на основе формирования тестовых примеров для логических функций.

Ключевые слова: *информационная технология, вычислительные комплексы, моделирование, программное обеспечение, испытания.*

Введение

Развитие вычислительных средств предполагает проведение испытаний унаследованных программ на новой аппаратной платформе. Если преемственность общесистемного программного обеспечения (ОПО), включая операционные системы, трансляторы, системы начальной загрузки, проверяется в полном объеме, то библиотеки и пакеты программ, входящие в промежуточный слой программного обеспечения, не всегда могут быть испытаны на новой вычислительной базе. Это обусловлено большим спектром пакетов и программных систем, требующихся для разных прикладных задач. При этом рассматриваемые пакеты часто взаимосвязаны, а взаимная зависимость может проявляться в виде достаточно длинных цепочек из выполняемых программ.

Тестирование унаследованного программного обеспечения

Предлагается формальное описание взаимодействующих пакетов программ в виде математической модели сложной системы. Будем считать, что использование последовательности пакетов программ аналогично функциональному объединению моделей элементов и подсистем в единый комплекс программно-реализованных алгоритмов. Имея такую модель, можно осуществлять проверку процессов для всего многообразия входных условий и текущих состояний реальной системы [1]. Для подобных пакетов программ процессы взаимодействия могут быть чрезвычайно сложными, и уверенности в том, что они в совокупности не приведут к ошибкам функционирования нового поколения вычислительной системы, нет. Это может происходить даже тогда, когда для каждого пакета в отдельности обеспечиваются преемственность и работоспособность при переходе на новую программно-аппаратную платформу.

Однако простой переход на обновленный пакет (библиотеку) может привести к невыполнению программного обеспечения (ПО). Причиной невыполнения программы могут служить как недостаточная отлаженность пакетов, так и проверка по версии или контрольным суммам используемых ПО пакетов (библиотек). Такие проверки используются, например, для предотвращения несанкционированного доступа к данным, обрабатываемым ПО.

Другой пример: библиотеки QT4 и QT5 [2] имеют столь сильные различия, что потребовалось сопровождение в составе ОПО обеих версий, т.к. для того, чтобы перевести ПО на QT5, в некоторых случаях необходимо будет изменить исходные тексты ПО (вплоть до переписи отдельных ее частей).

Будем считать, что комплекс программ состоит из совокупности программных компонент и удовлетворяет требованиям, характеризующим комплекс как сложную систему:

- программная система может быть расчленена (не обязательно единственным образом) на конечное число частей, называемых подсистемами; каждая подсистема в свою очередь может быть расчленена на конечное число более мелких подсистем и т.д. до получения в результате

конечного числа шагов – таких частей, называемых элементами сложной системы, относительно которых имеется договоренность, что в условиях данной задачи они не подлежат дальнейшему расчленению на части;
- элементы сложной системы функционируют не изолированно друг от друга, а во взаимодействии, при котором свойства одного зависят от условий, определяемых поведением других элементов;
- свойства сложной системы в целом определяются не только свойствами элементов, но и характером взаимодействия между элементами.

Рассмотрение программных компонент в рамках приведенных ограничений позволяет опереться на математический аппарат системного анализа. Для того чтобы задать комплекс программ как систему, необходимо и достаточно задать описание всех входящих в нее элементов и описать взаимодействие между ними. Элементами описываемой системы являются программные компоненты, представленные в виде специальных библиотек или ранее используемых фрагментов функциональных программ. Элементы в общем случае могут представлять задание агрегата (в агрегативной модели динамической системы) в виде его программной реализации в соответствующей вычислительной среде. Связи между компонентами должны удовлетворять требованиям, обусловленным информационными потоками в комплексе программ, описывающих соответствующие состояния информационной системы [3].

С учетом перечисленных ограничений на класс рассматриваемых программ корректным будет следующее описание.

Пусть A – некоторое непустое множество; $\{j_1^n, j_2^n, j_3^n, ..., j_s^n, ...\}$ – система n-отношений на множестве A. С каждым n-отношением j^n на множестве A можно сопоставить n-местную логическую функцию (предикат) P^n: A^n-{И, Л} так, что $P^n(a_{i1}, a_{i2}, a_{i3}, ..., a_{in}) =$ И тогда и только тогда, когда выполняется n-отношение $j^n(a_{i1}, a_{i2}, a_{i3}, ..., a_{in})$, где И, Л – логические значения истины и лжи соответственно, $(a_{i1}, a_{i2}, a_{i3}, ..., a_{in}) \in A$. Множество предикатов P^n: A^n определяет условия активизации соответствующих компонент.

Такой метод описания моделирующих программ предпроектного анализа информационных систем позволяет опереться на формальный аппарат алгоритмических алгебр В.М. Глушкова.

Отличие описываемой системы задания взаимодействующих компонент через функции предикатов на непустом множестве от традиционных методов описания функциональных программ состоит в том, что:
- не накладываются ограничения на полное задание логической функции предиката (частичная определенность);
- существует возможность задания множества логических функций, адекватно описывающих отображение P^n: A^n-{И, Л} (многоверсионное задание предиката).

В этом случае можно говорить о модели комплекса программ, которые должны быть проверены при переходе на новую программно-аппаратную платформу. Моделью будем называть систему, состоящую из множества A и определенной на данном множестве совокупности предикатов $\Pi = \{p_s^n \mid s = 1, 2, ...\}$.

Множество A будем называть основным множеством данной модели; предикаты, принадлежащие к Π, – ее основными предикатами; последовательность $n_1, n_2, ..., n_s, ...$ – типом модели, а $\Pi = \{p_s^n \mid s = 1, 2, ...\}$ – сигнатурой.

Модель $M_{A1} = <A_1; \Pi_1>$ (где A_1 – непустое подмножество A; $p1^n \in \Pi_1$ – предикат на множестве A_1, индуцированный предикатом $p1^n \in \Pi$) будем называть подмоделью модели $M_A = <A; \Pi>$. Множество однотипных подмоделей образует класс моделей по данному типу.

Для комплексов функциональных программ сложных информационных систем наличие изоморфных подмоделей соответствует режиму избыточности и возможности переиспользования соответствующих фрагментов. Построение модели на основе версий компонент сводится к формированию цепочек преобразований.

Построение функциональных отношений для взаимодействующих компонент моделирующих программ производится в процессе синтеза программы. Порождаемая программа может быть рассмотрена как расширяемое множество компонент, исходное множество которых определяется множеством начальных конструкций языка задания программ. Это могут быть как обычный язык программирования, так и языки формирования сценариев сборки функциональных программ на основе специализированных библиотек. Цепочки операторов над исходным множеством элементов формируют расширяемое множество компонент с их частично определенными функциональными отображениями. Следовательно, множество проверок программ будет определяться таким множеством цепочек выполнения, чтобы обеспечить требуемый уровень покрытия функций отношений на предикатном уровне.

Задание множества проверок требует решения двух взаимосвязанных задач:

1. Определение множества проверок заданного предиката активизации программного комплекса.

2. Определение правила, при котором функциональный компонент считается успешно выполненным.

Вторая задача больше относится к области применения вычислительных средств и решается способами соответствующих подборов признаков успешного завершения и эталонных результатов.

В ходе проведения расчетов встает вопрос об оценке функции чувствительности целевых функций поведения информационных систем к отдельным факторам (множеству факторов). Требование системного подхода к выводу свойств и целей системы из свойств и целей ее элементов вызывает необходимость построения системных целевых функций, чувствительных к свойствам элементов системы [1].

Обозначим функцию цели – y. Будем считать, что абсолютная функция чувствительности цели к изменению параметра x_i системы равна соответствующей частной производной $S_i = \frac{\partial y}{\partial x_i}$; частное отклонение $\Delta y_i = S_i \Delta x_i$ и полное отклонение $\Delta y = \sum_{i=1}^{N} S_i \Delta x_i$, где N – число рассматриваемых элементов в системе (подсистеме, уровне).

Функции чувствительности и отклонения во временно́й области (в динамических системах), соответственно, запишутся в следующем виде:

$$S_i(t) = \frac{\partial y(t)}{\partial x_i}, \quad \Delta y_i(t) = S_i(t)\Delta x_i, \quad \Delta y(t) = \sum_{i=1}^{N} S_i(t)\Delta x_i.$$

В приведенных соотношениях присутствуют лишь производные первого порядка. В ряде случаев эта аппроксимация недостаточна и необходимо учитывать последующие члены ряда Тейлора, используя, в частности, для представления функций чувствительности вторые производные:

$$S_{ij} = \frac{\partial^2 y}{\partial x_i \partial x_j}.$$

Естественно, что вычисление и использование в различных приложениях таких функций чувствительности и отклонений второго порядка является задачей повышенной сложности.

Первую задачу – определение множества проверок заданного предиката активизации программного комплекса – можно решать методами формирования тестовых примеров для логических функций, например, методами классификации моделей программ. В качестве критерия правильности работы комплекса программ может служить сравнение функционирования на предыдущей (A) и проверяемой (B) программно-аппаратных платформах.

В рамках однотипных классов моделей может быть рассмотрен вопрос об отношениях между моделями [3].

Однотипные модели $M_A = <A; \Pi>$ и $M_B = <B; \Pi 1>$ (где $\Pi = \{p_s^n \mid s = 1, 2, ...\}$, $\Pi 1 = \{q_s^n \mid s = 1, 2, ...\}$) будем называть изоморфными, если существует взаимнооднозначное отображение G множества A на множество B такое, что для любого $s = 1, 2, ...$ предикат $p_s^n(a_1, a_2, a_3, ..., a_{ns})$ справедлив тогда и только тогда, когда выполняется предикат $q_s^n((a_1)\ G, (a_2)\ G, ..., (a_{ns})G)$ для произвольных $a_1, a_2, a_3, ..., a_{ns} \in A$. Отображение G в этом случае соответствует изоморфизмам $M_A = <A; \Pi>$ и $M_B = <B; \Pi 1>$.

Для реальных программных комплексов множество $a_1, a_2, a_3, ..., a_{ns} \in A$ является огромным, и обеспечить перебор всех возможных условий в режиме испытаний фактически невозможно. Поэтому воспользуемся свойством иерархичности программных комплексов. С учетом того, что по схеме деления программная система может быть расчленена на конечное число подсистем, получим первый уровень тестовых примеров, где $p_s^n(a_1, a_2, a_3, ..., a_{ns})$ позволяет воспользоваться полным перебором условий.

Разбивая подсистемы на конечное число более мелких подсистем, получим большее количество управляемых при тестировании предикатов. В результате конечного числа шагов по детализации подсистем получим множество примеров, расширение которых будет порождать недопустимое (по критерию выделенных ресурсов) множество проверок.

Учет возможности проведения классификации подмоделей по данному типу (по допустимой последовательности применения различных отношений на A) позволяет определять границы применимости разрабатываемых методов создания и компонентов моделирующих программ и ограничиваться рассмотрением комплексов моделей одного класса.

Приведем наиболее распространенный пример анализа функционального программного комплекса, когда на развитие каждого уровня системы выделяются ограниченные средства и требуется найти их наилучшее использование – оптимальное распределение в пространстве составляющих элементов. При этом функция цены должна зависеть не только от номера элемента, но и от уровня (значения) параметра, его определяющего. Такой характер функционала затрат в наибольшей степени отвечал бы принципу возрастания затрат по мере возрастания характеризующего элемент свойства, а сведе́ние этого свойства до уровня максимального значения было бы практически невозможно (потребовались бы бесконечные ресурсы). При этом появляется возможность найти рациональное распределение свойств элементов некоторого уровня иерархии системы, когда возможно достичь требуемого уровня соответствия цели системы.

Вместе с тем приведенный материал очередной раз показывает невозможность полной проверки сложных функциональных программ при переходе на новую программно-аппаратную платформу. Бесконечная размерность предикатной функции позволяет ставить конструктивно вопрос только о проверках верхних уровней иерархии функциональных программ. Из этого следует, что при переходе на новую программно-аппаратную платформу требуется не только проводить контрольное тестирование на всех режимах взаимодействия привлекаемых библиотек и программных средств проектирования, но и иметь в виду то, что остаточные риски не будут нулевыми после проведенных проверок.

Заключение

В статье рассмотрены подходы к тестированию программного обеспечения на основе формализаций в виде алгебры алгоритмов В.М. Глушкова. Показана применимость такого подхода для случая использования унаследованных программ. Определено множество обоснованных проверок при построении тестов проверок заданного предиката активизации программного комплекса на основе формирования тестовых примеров для логических функций.

Предложен метод оценки функции влияния отдельных компонент на результаты работы системы. Показана возможность замены ряда натурных испытаний на расчет функций влияния отдельных компонент.

СПИСОК ЛИТЕРАТУРЫ

1. Парамонов Н.Б. Испытания при замене вычислительных средств сложных технических систем // Вопросы радиоэлектроники. 2011. Т. 4. № 3. С. 161–172.
2. Ильиных С. Портируем Qt4 приложение на Qt5 [Электронный ресурс]. URL: https://habrahabr.ru/post/164721/
3. Моделирование информационных систем на разнесенном стенде / Н.Б. Парамонов, Ю.В. Морозов, И.В. Минин, Д.А. Токарев, Ю.Н. Парамонов // Вопросы радиоэлектроники. 2014. Т. 4. № 3. С. 160–170.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Парамонов Николай Борисович, д.т.н., профессор, главный научный сотрудник, ПАО «ИНЭУМ им. И.С. Брука», 119334, Москва, ул. Вавилова, д. 24, тел.: 8 (499) 135-44-61, e-mail: paramonov_n_b@rambler.ru.
Минин Иван Валериевич, начальник, 432 ВП МО, тел.: 8 (495) 363-96-65, e-mail: minin_i@ineum.ru.
Янко Денис Викторович, начальник группы, 432 ВП МО, тел.: 8 (495) 363-96-65, e-mail: janko_d@ineum.ru.
Шаменков Николай Александрович, к.т.н., начальник отдела, НИИЦ ЦНИИ ВВКО МО, 129327, Москва, ул. Осташковская, д. 12а, тел.: 8 (903) 287-25-60, e-mail: shna810516@mail.ru.

For citation: Paramonov N.B., Minin I.V., Yanko D.V., Shamenkov N.F. Audit of software complexes when switching to a new software and hardware platform. Voprosy radioelektroniki, 2017, no. 3, pp. 100–103.

N.B. Paramonov, I.V. Minin, D.V. Yanko, N.F. Shamenkov

AUDIT OF SOFTWARE COMPLEXES WHEN SWITCHING TO A NEW SOFTWARE AND HARDWARE PLATFORM

Are examined approaches to testing of software on the basis of the formalizations in the form of the algebra of algorithms V.M. Glushkova. The applicability of this approach for the case of using the hereditary programs is shown. Are determined many substantiated checkings during the construction of the tests of testings of the assigned predicate of the making more active of program set on the basis of the formation of test examples for the logical functions.

Keywords: information technology, computer complexes, simulation, software, testing.

REFERENCES

1. Paramonov N.B. Tests at replacement of computing means of big technical systems. *Voprosy radioelektroniki,* 2011, vol. 4, no. 3, pp. 161–172 (In Russian).
2. Ilyinyh S. [Porting the Qt4 application on Qt5.] Available at: https://habrahabr.ru/post/164721/
3. Paramonov N.B., Morozov Yu. V., Minin I.V., Tokarev D.A. Modeling of information systems on the diverse stand. *Voprosy radioelektroniki,* 2014, vol. 4, no. 3, pp. 160–170 (In Russian).

AUTHORS

Paramonov Nikolay, Dr., professor, chief researcher, PJSC «Brook INEUM», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499) 135-44-61, e-mail: paramonov_n_b@rambler.ru.
Minin Ivan, chief, 432 VP MO, Moscow, Russian Federation, tel.: +7 (495) 363-96-65, e-mail: minin_i@ineum.ru.
Yanko Denis, head of group, 432 VP MO, Moscow, Russian Federation, tel.: +7 (495)363-96-65, e-mail: janko_d@ineum.ru.
Shamenkov Nikolay, PhD, head of department, NIIC CNII VVKO MO, 12a, Ostashkovskaya st., Moscow, 129327, Russian Federation, tel.: +7 (903) 287-25-60, e-mail: shna810516@mail.ru.