УДК 681.3.06

Г.Б. МОРОЗ, Г.И. КОВАЛЬ, Т.М. КОРОТУН

КОНЦЕПЦИЯ ПРОФИЛЕЙ В ИНЖЕНЕРИИ НАДЕЖНОСТИ ПРОГРАММНЫХ СИСТЕМ

1. Введение

Проблема прогнозирования, обеспечения и оценки надежности программных систем (ПС) всегда была в центре внимания разработчиков, о чем свидетельствует большое количество посвященных ей монографий и статей. Более того, с годами она становится все острее и сложнее. Причиной тому является проникновение компьютерных технологий в критические сферы человеческой деятельности, все возрастающая сложность ПС, стремление фирм-разработчиков к захвату как можно более широких сегментов рынка и, как следствие, все обостряющаяся конкуренция между ними. Поэтому в последние годы происходит интенсивный поиск новых концепций, подходов и методов инженерии надежности программного обеспечения, которые гарантировали бы разработку высококачественных ПС [1]. Одна из таких концепций, названная нами концепцией профилей, рассматривается в данной работе.

Основная идея предлагаемой концепции – максимальный учет при разработке ПС всех аспектов ее будущего применения с целью создания таких ПС, которые бы полностью удовлетворяли предъявляемым к ним требованиям по надежности со стороны конкретных пользователей (или категорий пользователей).

Истоки этой концепции восходят к работе [2], где впервые была высказана идея «взглянуть на надежность ПС глазами пользователя». В 90-е годы эта идея развивалась и продолжает развиваться в работах Мусы [3 – 5], Вайттекера [6, 7], Мансона [8, 9], Ранисана [10] и других авторов практически во всех направлениях, обозначенных в работе [2].

Однако, изучая различные аспекты применения ПС, упомянутые авторы невольно используют одни и те же термины для определения разных понятий. Кроме того, они расходятся во взглядах на необходимую степень детализации изучаемого процесса применения ПС.

В данной работе предпринята попытка обобщить современные подходы к определению, построению и применению профилей. В ней предложена концепция профилей как система взглядов на процесс применения ПС с позиций определения количественных характеристик элементов этого процесса, учет которых позволяет повысить эффективность разработки ПС и улучшить ее качество.

Выбор процесса применения ПС в качестве объекта исследования отличает данную концепцию от других концепций повышения качества и надежности ПС, акцентирующих внимание преимущественно на свойствах программного продукта и характеристиках процессов его разработки [11]. Мы считаем, что этот выбор оправдан, поскольку знание особенностей будущего применения ПС позволит правильно распределить ограниченные ресурсы разработки, лучше протестировать ПС и снизить затраты на ее сопровождение при эксплуатации.

2. Некоторые особенности разработки и использования современных ПС

В предлагаемой концепции нашли свое отражение следующие особенности разработки и использования современных ПС.

Во-первых, разработка крупных и сложных программных систем в условиях ограниченных ресурсов требует применения дифференцированного (избирательного) подхода к обеспечению надежности ПС, распределению трудовых и материальных ресурсов по функциям и компонентам, построению графика проекта разработки и ввода ПС в эксплуатацию.

Во-вторых, анализу собственно процесса применения ПС обычно предшествует анализ (маркетинговые исследования) областей возможного применения (приобретения) ПС в различных отраслях потребления, определение классов потенциальных пользователей ПС в этих отраслях и др. Особенно это касается ПС широкого потребления. Такого рода анализ позволит более точно установить требования к функциональным и техническим характеристикам ПС, применяемым в конкретных деловых процессах организаций-потребителей ПС.

В-третьих, достоверность оценок надежности ПС зависит от степени соответствия процесса тестирования и моделей надежности ПС реальному процессу применения ПС с учетом различных категорий пользователей, режимов и условий эксплуатации ПС.

В-четвертых, надежность ПС зависит от того, какое именно подмножество программного кода выполняется в течение рассматриваемого интервала времени. Если всегда выполняется одна и та же бездефектная часть кода, то ПС может работать неограниченное время без какой-либо вероятности отказа даже тогда, когда в ней имеются части кода, содержащие дефекты. Поэтому точность предсказания надежности ПС будет ограничена в той степени, в которой достигнуто понимание будущих сценариев ее функционирования.

В-пятых, поскольку выполнение функций ПС всегда связано с работой четко определенного подмножества ее кода, нужно исследовать не просто надежность ПС как программного объекта, а надежность отдельных функциональных возможностей, предоставляемых пользователю путем выполнения кода. Важно определить те элементарные функции, которые отказывают, и оценить их надежность.

В-шестых, чем больше существует категорий пользователей данной ПС, функций и режимов ее эксплуатации в различных средах, тем сложнее обеспечить ее надежное функционирование. Пользователи, применяющие одну и ту же ПС различным образом, скорее всего, будут иметь разные мнения о ее надежности. Поэтому, знание и учет на всех стадиях жизненного цикла (ЖЦ) ПС характеристик процесса ее применения будет способствовать улучшению качества спецификаций требований к ПС, процессов разработки и тестирования ПС, а также управления программными проектами, поскольку обеспечит регулярную «обратную связь» с заказчиками и пользователями и своевременную реакцию на изменение требований к ПС.

Таким образом, учитывая вышеперечисленные особенности разработки и использования ПС, мы считаем, что надежность является динамической характеристикой качества, которая зависит не только от свойств самой программной системы, но и от того, каким образом она применяется пользователями в ходе эксплуатации, то есть от характеристик процесса применения ПС.

Процесс применения ПС рассматривается как совокупность взаимосвязанных действий, направленных на решение определенных задач в некоторой проблемной области с помощью разрабатываемой ПС. Составляющими элементами процесса применения ПС, количественные характеристики которых необходимо определять, являются:

– субъекты – потребители и непосредственные пользователи ПС;

– объекты – элементы функциональной, информационной и программной архитектуры ПС (функции, данные, исполнительный код);

– действия – внешние операции, образующие операционные сценарии применения ПС субъектами процесса;

– ограничения – условия среды и режимы применения ПС.

В качестве количественной характеристики каждого из этих элементов процесса применения будем использовать соответствующий профиль.

3. Основные виды и структура системы профилей

Под профилем понимается полное множество альтернатив (например, множество альтернативных категорий пользователей, функций и т.п.), для каждой из которых существует определенная вероятность появления. Полнота множества альтернатив означает, что сумма вероятностей их появления должна быть равна единице. Построение профилей заключается в выявлении соответствующего множества альтернатив для элементов процесса применения ПС и их вероятностной оценке.

В зависимости от сферы применения ПС, ее объемов и сложности, а также принятых подходов к проектированию и разработке может быть построена та или иная многоуровневая система профилей (на уровне системы, подсистем, отдельных компонентов и т.п.). Кроме того, независимо от уровня определения профилей, они могут образовывать иерархии, в которых каждый профиль нижнего уровня иерархии является детализацией (отображением) соответствующего профиля верхнего уровня иерархии. Профили могут определяться на всех стадиях жизненного цикла ПС, начиная с определения концепции автоматизации определенного вида деятельности в некоторой проблемной области. Преобразование одного профиля в другой выполняется по мере продвижения от одной стадии жизненного цикла ПС к другой.

В ходе анализа проблемной области, определения требований к ПС и технологии программирования, выделения множества решаемых задач и декомпозиции системы на составляющие ее программные компоненты или модули могут строиться различные виды профилей, основными из которых являются следующие:

– потребительский профиль ПС (C-профиль);

– пользовательский профиль ПС (U-профиль);

– профиль системных режимов ПС (SM-профиль);

– функциональный профиль (F-профиль);

– операционный профиль (O-профиль);

– исполнительный (или модульный) профиль (K-профиль) и др.

Присутствие каждого из видов профилей в системе профилей, используемых для описания элементов процесса применения ПС, необязательно.

Далее мы дадим краткую характеристику основных элементов процесса применения ПС, приведем формальные определения некоторых из перечисленных видов профилей и покажем, каким образом они взаимодействуют.

4. Потребительский профиль

Потребитель ПС – это лицо, группа лиц или организация, которая приобретает ПС (отдельно или в составе компьютерной системы) так же, как и любой другой продукт.

Для формального определения потребительского профиля разобьем множество всех возможных потребителей на *n* независимых групп, в рамках каждой из которых предполагается относительно единообразное использование ПС. Примером ПС с различными группами потребителей может быть любой программный компонент Microsoft Office, например, Excel. Можно ожидать, что каждая группа потребителей будет использовать разные функциональные возможности электронных таблиц, с разной частотой и продолжительностью.

Пусть $\mathrm{N_c} = \{c_1, c_2, ...c_\mathrm{n}\}$ – полное множество возможных групп потребителей (заказчиков) данного ПС, а p_{ci} – вероятность того, что ПС может быть приобретено i-потребителем (т.е. потребителем i-ой группы), $i = 1,..., n$. Множество пар $\Pi_\mathrm{c} = \{(c_\mathrm{i}, p_{\mathrm{ci}}), i = 1,..., n\}$ называется *потребительским профилем ПС.* Иными словами, потребительский профиль – это список всех возможных независимых групп потребителей, которые могут приобрести ПС, и вероятностей ее приобретения этими группами.

5. Пользовательский профиль

Любая группа потребителей может быть разделена на категории пользователей. Пользователь – это лицо, группа лиц или организация, которые непосредственно *используют* (в отличие от потребителя, который *приобретает*) ПС. Множество пользователей, которые будут выполнять схожие действия (операции) в ПС, образуют ту или иную, отличную от других, категорию пользователей.

Пользовательский профиль может быть построен для каждого элемента потребительского профиля, например, следующим образом. Обозначим через $\mathrm{N_{ui}} = \{u_{\mathrm{i}1}, u_{\mathrm{i}2}, ...u_{\mathrm{ik_i}}\}$ полное множество возможных категорий пользователей для i-потребителя и пусть g_{ij} – вероятность наличия j-пользователей (т.е. пользователей j-ой категории) среди потребителей i–ой группы. Тогда множество пар $\Pi_{ui} = \{(u_{ij}, g_{ij}), j = 1,...k_i\}$ образуют *пользовательский профиль i–потребителя.*

*Пользовательский профиль П*С – это множество пар $\Pi_u = \{(u_i, p_{ui}), j = 1,...m\}$, где $u_i \in N_u = \bigcup_{k=1}^{n} \mathrm{N_{uk}}$, а p_{ui} определяются по формуле $\mathrm{p_{ui}} = \sum_{j=1}^{m} \sum_{i=1}^{n} p_{ci} g_{ij}$.

Иными словами, *пользовательский профиль ПС* – это список возможных категорий пользователей ПС и связанных с ними вероятностей использования ПС определенным образом.

Как видим, при построении пользовательского профиля ПС следует исходить из того, что, если в различных группах потребителей встречаются одни и те же категории пользователей, то они, во-первых, должны быть представлены в списке категорий пользователей одним элементом, то есть этот список не должен содержать повторяющихся элементов. Во-вторых, вероятность появления каждой категории пользователей в группе потребителей должна умножаться на вероятность появления группы потребителей с целью получения общей вероятности для данной категории пользователей. И, в-третьих, когда категории пользователей комбинируются по группам потребителей, обобщенные вероятности появления каждой категории пользователей должны суммироваться для получения общей вероятности появления категории пользователей безотносительно к группе потребителей.

Абстрактный пример построения простейшего пользовательского профиля представлен в табл.1. В этом примере для ПС выделено две группы потребителей, в каждой из которых (с разной вероятностью) присутствуют пользователи двух категорий.

Таблица 1. Пользовательский профиль ПС

Категории пользователей	Группа потребителей 1		Группа потребителей 2		Суммарная вероятность по группам =1
	Вероятность группы = 0,3		Вероятность группы = 0,7		
	Вероятность категории в группе	Вероятность категории с учетом вероятности в группе	Вероятность категории в группе	Вероятность категории с учетом вероятности в группе	Общая вероятность для категории
Категория пользователей 1	0,9	0,9*0,3 = 0,27	0,6	0,42	0,27+0,42=0,69
Категория пользователей 2	0,1	0,1*0,3 = 0,03	0,4	0,28	0,03+0,28=0,31
	Сумма по категориям в группе = 1		Сумма по категориям в группе = 1		Сумма по категориям = 1

6. Профиль системных режимов

Выделение системных режимов ПС может быть обусловлено следующими причинами:

– системные режимы можно определить необходимостью обслуживания различных категорий пользователей (например, режим администрирования; режим сопровождения; рабочий режим (обслуживание пользователей));

– наличием периодов времени в работе ПС, предполагающих различную нагрузку на систему (например, дневной режим, ночной режим);

– наличием различных состояний функционирования ПС (например, режим перезагрузки; режим стабильного функционирования; режим инициализации; режим восстановления);

– наличием различной аппаратно-программной среды (например, режим командной строки; пакетный режим; диалоговый режим);

– наличием критических условий (например, режим отключения (для атомной электростанции в опасной ситуации));

– наличием пользователей с различным опытом работы (например, режим работы новичка; режим работы опытного пользователя и др.).

Системные режимы можно определить (интерпретировать) как независимые сегменты функционирования системы или различные направления ее использования.

С точки зрения анализа поведения системы, в ходе ее работы, *системный режим* – это множество функций ПС или выполняемых пользователями ПС операций, сгруппированных для удобства анализа поведения ПС в ходе ее функционирования в определенной среде эксплуатации «на площадках» потребителей.

Большинство систем имеют более, чем один режим функционирования. Например, полет самолета (оснащенного бортовыми компьютерами) обеспечивается работой ПС в режимах разгона, подъема, полета на заданном уровне, спуска и приземления.

Система может переключаться между режимами последовательно или допускать одновременную работу в нескольких режимах, используя совместно одни и те же ресурсы.

Профиль системных режимов – это множество пар $\Pi_s = \{(s_i, p_{si}), i = 1,...,r\}$, где s_i – i-й режим ПС; p_{si} – вероятность пребывания ПС в i-ом режиме.

Подход к построению профиля системных режимов тот же, что и к построению пользовательского профиля. Должны определяться вероятности режимов работы каждой категории пользователей с учетом вероятностей появления каждого режима, каждой категории пользователей и каждой группы потребителей.

Цель определения режимов системы – максимальный учет особенностей поведения ПС в различных контекстах применения, выделение независимых программных комплексов и их программных компонентов, которые целесообразно разрабатывать и тестировать отдельно, локализация изменений или уточнений требований к ПС по мере ее разработки и т.п.

Ограничений на количество режимов системы не существует. Однако, поскольку для каждого режима системы далее могут разрабатываться операционные и другие профили, нужно соблюдать баланс между трудоемкостью разработки самих профилей и реальной пользой, которую они могут принести в ходе разработки и тестирования системы.

После того как установлены режимы, в которых должна работать ПС, необходимо определить перечень функций ПС в рамках каждого режима и далее предпринять тот же подход к построению функционального профиля ПС.

7. Функциональные профили

Изначально, на стадии проектирования, потребитель и пользователь воспринимают ПС через призму предоставляемых ею функций, отраженных в техническом задании (ТЗ) на разработку ПС в виде спецификации функциональных требований верхнего уровня. Это положение вещей зафиксировано в стандартах ДСТУ, системах стандартов ГОСТ 34 и ГОСТ 19.

На всех последующих стадиях жизненного цикла ПС осуществляется декомпозиция каждой функции верхнего уровня на более мелкие функции. В результате образуется иерархическая

структура функциональных уровней (функциональная пирамида). Количество этих уровней различно для различных ПС и зависит от сложности ПС, опыта разработчиков, требований к надежности и т. п. При этом самый нижний функциональный уровень представляют независимые элементарные функции, реализуемые программным кодом. Заметим, что в современной концепции измерения объема функциональности ПС, отраженной в серии проектов стандартов ISO/IEC 14143, элементарные функции классифицируются, например, как функции ввода, вывода, запроса или др., в зависимости от класса ПС [12].

Множество всех возможных функций условно можно разбить на две категории: внешние и внутренние. Внешние – это те функции верхних уровней функциональной пирамиды, которые определены в техническом задании и не находят непосредственного отражения в программном коде. Внутренние функции – это функции нижнего уровня функциональной пирамиды, которые описаны в конструкторской документации и реализованы программным кодом.

Для каждого функционального уровня может быть построен свой функциональный профиль. *Функциональный профиль i-го уровня* представляет собой множество пар $F^i = \{(f_k^i, p_k^i), k = 1,...,n_i\}$, где n_i – количество функций *i*–го уровня; f_k^i – *k*-я функция *i*-го уровня; p_k^i – вероятность выполнения f_k^i в процессе функционирования ПС.

Функциональные профили могут быть явными и неявными. В случае *явного функционального профиля* каждая функция должна быть полностью описана значениями ключевых входных переменных.

Ключевая входная переменная – внешний параметр, определяющий выбор пути, по которому пойдет работа (и, возможно, разработка) программной системы в зависимости от различных принимаемых переменной значений. Ключевые входные переменные позволяют дифференцировать функции системы (то есть распознавать их и далее анализировать отдельно).

Явный функциональный профиль содержит спецификацию набора *одновременно всех* значений ключевых переменных с указанием вероятности принятия этих значений.

Неявный функциональный профиль может быть представлен только как *совокупность подпрофилей*, в которых соответствующим ключевым внешним переменным присваиваются вероятности, связываемые с *выбором* значений на определенном уровне.

Неявный профиль может использоваться только в тех случаях, когда ключевые входные переменные не зависят друг от друга в смысле вероятности появления их значений. Если же они коррелируют, необходимо строить явный функциональный профиль.

Для лучшего понимания различия между явными и неявными функциональными профилями рассмотрим простой пример.

Пусть разрабатываемая гипотетическая система - архиватор файлов, о котором известно следующее:

– он должен работать в DOS и Windows (системные режимы **D** и **W**);

– должен предоставлять минимальные услуги – *помещения* файлов в архив (**a**), *извлечение* файлов из архива (**e**). При этом файлы могут быть организованы в *каталоги* и их нужно обрабатывать совместно (т.е. каталог и все его подкаталоги) (**r**);

– архив может занимать *несколько томов* (например, архивирование может производиться с переходом с одной дискеты на другую) (**v**);

– существующие в архиве файлы могут *обновляться* (**u**).

Функции архиватора (как программной системы) можно определить по-разному, в зависимости, например, от того, будут ли архиватор программировать несколько исполнителей или один. Если исполнитель один, – это может быть одна функция с параметрами: *архиватор (команда, переключатель…)*. Если исполнителя два, – две функции: *архивация (переключатель…)* и *деархивация (переключатель…)*.

Максимальное число исполнителей определяется сочетанием **A(a,r), A(a,v), A(a,u), A(e,r), A(e,v), A(e,u).**

Примеры неявного и явного функциональных профилей приведены в табл. 2 и 3. В них независимые ключевые переменные – это *команда* (**К**), которая должна выполняться архиватором и может иметь два значения: К1**=a** или К2**=e**, и *переключатель* (**Р**) режима, принимающий три значения: Р1**=r** или Р2**=v,** или Р3**=u**. Комбинируя значения ключевых переменных, получим шесть функций.

Таблица 2. Неявный функциональный профиль

Подпрофиль 1		Подпрофиль 2	
Значение ключевой входной переменной	Вероятность появления	Значение ключевой входной переменной	Вероятность появления
К1 К2	0,6 0,3	Р1 Р2 Р3	0,7 0,2 0,1

Таблица 3. Явный функциональный профиль

Значения ключевых входных переменных	Вероятность появления
К1Р1	0,42
К1Р2	0,12
К1Р3	0,06
К2Р1	0,21
К2Р2	0,06
К2Р3	0,03

Главное преимущество использования неявного профиля состоит в том, что нужно описывать существенно меньшее количество элементов, всего лишь сумму числа уровней выбора ключевых входных параметров, то есть вектор ключевых переменных (в данном примере – это 2+3 элемента).

Явный профиль имеет число элементов, равное *произведению* количеств принимаемых значений каждой ключевой переменной (в данном примере – это 2×3 элемента). При этом вероятность появления i-ой функции профиля может быть получена по формуле

$$p_{fi} = \frac{v_i}{\sum_{i=1}^{n} v_i},$$

где v_i – частота появления *i*–й функции в час; n – общее количество функций, в которых нуждаются все пользователи.

Заметим, что в большинстве случаев нет необходимости строить явный профиль, поскольку он легко получается из неявного профиля.

Недостаток неявного профиля состоит в том, что можно не учесть некоторых редко используемых, но критических операций при тестировании системы, если не записать всех комбинаций ключевых входных переменных, как это делается при построении явного профиля. Однако неявный функциональный профиль удобен при выделении функций систем, основанных на обработке транзакций, когда атрибуты транзакций независимы и вероятности их появления известны.

Разработка функционального профиля является одной из работ стадии определения требований к ПС и в общем случае включает следующие шаги:

– построение начального списка функций;

– определение внешних переменных среды;

– построение окончательного списка функций;

– назначение вероятностей появления.

Начальный список функций должен основываться на спецификации функциональных требований к ПС в ТЗ на ее разработку. Если строится неявный профиль, этот список может быть образован функциями, связываемыми с каждой ключевой входной переменной. Если работа по извлечению требований была проведена надлежащим образом, функции должны идентифицироваться достаточно легко. Если функции сложно идентифицировать, то требования оказались неполными или неясными.

Следующий шаг состоит в установлении переменных среды и областей их значений, которые определяют различные режимы поведения системы и сегрегируют (разделяют на части) разработку системы. Переменные среды характеризуют *условия*, оказывающие влияние на выбор пути работы программной системы. Пример переменной среды для архиватора – операционная система – DOS (**D**) или Windows (**W**)).

Перед тем, как получить окончательный список функций, необходимо проанализировать зависимость ключевых переменных среды и функциональных переменных. Список должен быть настолько ортогональным, насколько это возможно. Если одна переменная существенно зависит от другой, она может быть удалена из окончательного списка функций. Частичные зависимости могут вызвать затруднения, поскольку придется перечислять все возможные комбинации уровней зависимых переменных. Те взаимодействия, которые не могут возникнуть и которые имеют несущественную вероятность появления, должны быть удалены из списка. Для рассматриваемого примера нет невозможных взаимодействий, так как архиватор должен выполнять одни и те же функции в любой среде.

Окончательное число функций в списке вычисляется как произведение числа функций в начальном списке и числа уровней переменных среды минус число комбинаций начальных функций и значений переменных среды, которые не могут появиться.

Пусть вероятность работы архиватора в среде DOS равна 0,3, а в среде Windows – 0,7. Окончательный список функций состоит из функций и соответствующих им переменных среды (табл. 4). Вероятности появления каждой функции можно оценить экспертным путем, хотя наилучший способ их получения – это анализ самого позднего выпуска данной или подобной по функциям системы. Эти данные могут содержаться в системных журналах или записях электронных средств регистрации событий при работе системы.

Таблица 4. Пример окончательного списка функций

Функция	Системный режим	Вероятность появления
К1Р1	D	0,42 * 0,3
	W	0,42 * 0,7
К1Р2	D	0,12* 0,3
	W	0,12* 0,7
К1Р3	D	0,06* 0,3
	W	0,06* 0,7
К2Р1	D	0,21* 0,3
	W	0,21* 0,7
К2Р2	D	0,06* 0,3
	W	0,06* 0,7
К2Р3	D	0,03* 0,3
	W	0,03* 0,7

8. Спецификации использования и операционный профиль ПС

Как отмечалось ранее, надежность – это динамическая характеристика, которая имеет смысл только тогда, когда она связывается с определенным использованием системы. Различные пользователи сталкиваются с разной надежностью, поскольку используют ПС различным образом. Для того, чтобы получить оценку надежности ПС, предназначенной для конкретной группы пользователей, до начала ввода ее в эксплуатацию необходимо иметь модель предполагаемого использования ПС, представленную в виде *спецификации использования*. Эта спецификация должна отражать два момента, а именно: 1) максимально полное описание того, как пользователи могут использовать ПС; 2) вероятностные характеристики различных вариантов использования.

Модель использования специфицирует структуру использования программной системы и может определяться в терминах внешне наблюдаемых состояний системы или инициируемых пользователем событий (операций), с которыми связаны переходы между состояниями.

В литературных источниках последних лет предложено несколько подходов к представлению спецификации использования ПС, а именно: основанные на марковских моделях [2, 6, 7], операционных сценариях [3 – 5, 8], иерархических моделях состояний [10], моделях технологических процессов [9]. Выбор того или иного подхода может быть обусловлен особенностями проблемной области, для которой разрабатывается ПС, а также используемых подходов к анализу и проектированию программных систем (в частности, структурного или объектно-ориентированного подходов). Однако на практике наибольшее применение находят первые два подхода. Поэтому остановимся на них несколько подробней.

Марковская модель и модель иерархии состояний определяют использование ПС в терминах извне наблюдаемых состояний системы и событий, которые возникают при переходе из одного состояния в другое. Разница между ними состоит в возможности последней моделировать

многопользовательские ПС, в то время как марковская модель пригодна для моделирования только однопользовательских ПС.

Если моделирование использования ПС осуществляется цепью Маркова, то в пространстве состояний ПС выделяется единственное стартовое состояние $\mathbf{S_0}$ (состояние активизации работы ПС), единственное конечное состояние $\mathbf{S_n}$ (состояние завершения работы ПС) и множество промежуточных пользовательских состояний $\mathbf{S_i}$ (рис. 1).

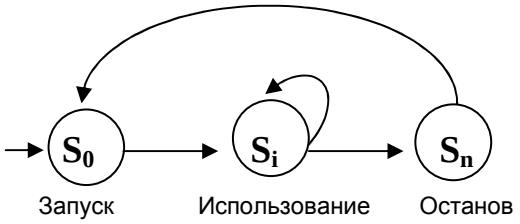

Рис. 1. Модель верхнего уровня использования ПС

Множество состояний $S = \{S_0, \{S_i\}, S_n\}$ упорядочивается вероятностным отношением перехода $(S \times [0,1] \times S)$. Для каждой дуги $D_{i,i+1}$, определенной этим отношением, следующее состояние S_{i+1} не зависит от всех прошлых по отношению к S_i состояний.

Каждое новое состояние S_{i+1} порождается для каждой операции указанной дугой, исходящей от текущего состояния S_i (начиная с состояния запуска). Выходные дуги от вновь созданного состояния образуются путем анализа операций, которые возможно выполнить в текущем состоянии. Эти дуги могут быть направлены на предыдущие состояния или приводить к созданию новых состояний. В результате образуется направленный граф, состояния которого связаны со входами из пространства входов ПС. Дуги графа определяют операции над входами из пространства входов и таким образом упорядочивают входы во входном пространстве.

Пользовательская цепь Маркова имеет двухфазовую конструкцию. В структурной фазе образуются состояния и дуги, а затем в статистической фазе назначаются вероятности перехода из одного состояния цепи в другое, и таким образом строится модель использования ПС. При этом обычно используют один из трех подходов к определению вероятностей переходов между состояниями системы. Если в наличии нет информации, позволяющей сделать обоснованный выбор вероятностей, строится однородная модель использования, в которой все вероятности перехода равны. Если существует информация о реальных последовательностях и частоте появления состояний, строится неоднородная модель использования. Это возможно при наличии прототипа ПС или ранней версии ПС (при условии их инструментальной оснастки средствами оценки частоты появления состояний). Могут строиться также ожидаемые (прогнозируемые экспертами) верхний и нижний пределы возможного отклонения вероятностей перехода.

Во многих случаях марковская модель может быть преобразована в операционную модель использования ПС, которая в силу своей простоты находит все более широкое практическое применение.

Операция – инициированный извне шаг в процессе выполнения функции или решения задачи пользователя, выполняемый программной системой незримо.

Операционный профиль представляет собой множество пар $O = \{(o_i, p_{oi}), i = 1,..., n_o\}$, где n_o – количество операций; o_i – *i*-я операция; p_{oi} – вероятность появления операции o_i в процессе функционирования ПС.

Операционный профиль специфицирует использование ПС в терминах операций в системе и вероятностей их появлений.

Операционный сценарий – последовательность операций (шагов) процесса выполнения функции или решения задачи.

Множество операционных сценариев образуют *операционную модель использования ПС* пользователем.

На основе операционного профиля может быть определен *профиль операционных сценариев* использования ПС, который представляет собой множество пар $S = \{(s_i, p_{si}), i = 1,..., n_s\}$, где n_s – количество всех операционных сценариев; s_i – і-й операционный сценарий; p_{si} – вероятность появления s_i. Вероятность появления (выбора) определенного операционного сценария вычисляется как произведение всех вероятностей перехода в этом операционном сценарии.

При работе пользователя по конкретному операционному сценарию происходит выполнение операций, которые параметризованы соответствующими входами из пространства входов. Таким образом, каждый сценарий представляет разбиение входного пространства ПС. Выполнение экземпляра сценария с конкретными параметрами образует *прогон*. Это понятие обычно используется при тестировании ПС.

Следует обратить внимание на то обстоятельство, что *спецификации использования*, по сути, устанавливают взаимосвязь между внутренними и внешними функциями ПС. С одной стороны, каждая внешняя функция прямо или опосредованно (через внешние функции более низкого уровня) может быть представлена одной или несколькими операциями. С другой стороны, каждая операция может быть реализована одной или несколькими элементарными внутренними функциями. Поэтому операционный профиль является функцией функционального профиля *внешних функций*, в то время как функциональный профиль *внутренних функций* является функцией операционного профиля.

Подчеркнем также, что существующие отечественные нормативные документы, в соответствии с которыми обычно разрабатываются ПС, не требуют включения в спецификацию требований к ПС указания возможных сценариев ее использования наряду с определением перечня функциональных возможностей и технических характеристик, что, несомненно, является их недостатком. Предшествующие испытаниям ПС функциональное и статистическое тестирование также обычно не основаны на предварительном анализе всевозможных (или, по крайней мере, наиболее вероятных) сценариев работы пользователей. При этом результаты тестирования и оценки надежности ПС страдают недостоверностью. С этой точки зрения, для практики программной инженерии, безусловно, важен пользовательский взгляд на операции, операционные сценарии и операционные профили ПС.

9. Исполнительный профиль

Алгоритм выполнения функции или операции реализуется в ПС в виде определенного подмножества программного кода, распределенного по программным модулям (при модульном программировании) или программным объектам-обработчикам событий (при программировании, управляемом событиями, в объектно-ориентированных средах).

Стоит остановиться на различиях в логике построения и выполнения программ, обусловленных особенностями операционных систем, в среде которых программы должны функционировать.

В однопользовательских однозадачных операционных системах с текстовым интерфейсом, таких, как, например, MS DOS, программы выполняются *линейно*, от модуля к модулю, ожидая от пользователя ввода той или иной команды или ответа на запрос и блокируя нежелательные в этот момент действия.

В однопользовательских многозадачных операционных системах с графическим интерфейсом, таких, как, например, MS Windows, логика работы программ (*приложений* Windows) называется логикой, управляемой событиями. Под событием понимается обнаружение в очереди сообщений к приложению того или иного сообщения. Основная работа, которую выполняет приложение, – это обслуживание собственной очереди сообщений, для чего в приложении организован цикл обработки сообщений. Приложение Windows после инициализации переходит в состояние постоянного опроса собственной очереди сообщений. При появлении сообщения приложение начинает его обработку, выполняя определенный фрагмент кода, а после обработки вновь возвращается к опросу собственной очереди сообщений.

Исполнительный профиль – множество программных действий, связанных с реализацией внутренних функций, с указанием доли этих действий, приходящихся на каждый программный модуль или фрагмент кода обработчика события (далее просто модуль).

Для лучшего понимания сути исполнительного профиля остановимся кратко на некоторых моментах процессов разработки и функционирования ПС.

Обычно на ранних стадиях жизненного цикла ПС определяется множество внешних функциональных требований, которые специфицируются в ТЗ и находят отражение во множестве пользовательских операций O_p. В ходе детального проектирования и программирования ПС вначале множество пользовательских операций отображается во множество внутренних элементарных функций F, а затем осуществляется назначение этим функциям определенных программных модулей m из множества всех модулей M. То есть процесс детального проектирования ПС может рассматриваться как процесс определения множества отношений $НАЗНАЧИТЬ$ на $F \times M$ таких, что $НАЗНАЧИТЬ(f, m)$ верно, если элементарная функция f реализуется модулем m.

По существу, множество программных модулей M может быть разбито на два непересекающихся подмножества M_c и M_F, где M_c – множество общих программных модулей, которые будут выполняться независимо от элементарной функции, осуществляемой ПС, а M_F –

множество программных модулей, которые вызываются только в ответ на выполнение определенной элементарной функции. Множество общих модулей $M_c \subset M$ можно определить следующим образом:

$$M_c = \{m : M \mid f \in F \cdot НАЗНАЧИТЬ(f, m)\}.$$

В то же время при выполнении ПС определенной функции f возможны два случая:

1) каждый раз, когда выполняется функция f, будет вызываться одно и то же множество программных модулей $M_i^{(f)}$. В этом случае будем говорить, что эти модули полностью охватываются элементарной функцией f. Это множество безусловно включаемых модулей для элементарной функции f можно определить следующим образом:

$$M_i^{(f)} = \{m : M_F \mid \forall f \in F \cdot НАЗНАЧИТЬ(f, m) \Rightarrow p(f, m) = 1\},$$

где $p(f, m)$ – частота (доля) вызова модуля $m \in M$ во время осуществления элементарной функции $f \in F$;

2) при выполнении функции f могут вызываться различные программные модули из некоторого множества $M_p^{(f)} \subset M_F$. Об этих модулях говорят, что они являются *потенциально вызываемыми модулями*. Множество потенциально вызываемых модулей есть

$$M_p^{(f)} = \{m : M_F \mid \exists f \in F \cdot НАЗНАЧИТЬ(f, m) \wedge 0 < p(f, m) < 1\}.$$

В результате детального проектирования ПС будет получено множество программных модулей $M_f = M_p^{(f)} \cup M_i^{(f)}$, которые могут использоваться при выполнении заданной элементарной функции f.

С точки зрения проекта ПС, реальные проблемы в понимании динамического поведения ПС связаны с множеством потенциально вызываемых модулей M_p. Чем больше мощность этого множества, тем менее определенно можно говорить о поведении ПС, осуществляющей соответствующую функцию. Для любого экземпляра выполнения (прогона) этой элементарной функции может вызываться различное количество модулей в M_p.

По мере осуществления своего функционального назначения ПС пропорционально разделяет свои действия между множеством модулей, переходя от модуля к модулю в последовательности вызовов-возвратов. Каждый вызываемый модуль в этой последовательности имеет соответствующую частоту вызова. Когда ПС подвергается серии функциональных обращений (прогонов), отработка каждой пользовательской операции разная, вследствие чего выполняется различное множество элементарных функций, которые, в свою очередь, вызывают, возможно, разные множества программных модулей.

Каждая внутренняя элементарная функция j имеет отличное от других множество модулей M_{fj}, которые она может вызвать к действию. В любой произвольный момент времени в ходе

интерпретации f_i ПС будет выполнять модуль $m_i \in M_{fj}$ с некоторой вероятностью P_{ij}. Из этого распределения условной вероятности для всех внутренних элементарных функций мы можем получить исполнительный профиль для ПС как функцию внутреннего функционального профиля системы. Такие исполнительные профили могут быть хорошей основой для разработки более точных и адекватных моделей прогнозирования и оценки надежности ПС.

10. Уровни определения и преимущества использования профилей в программной инженерии

Профили применения могут определяться для ПС, ее подсистем и/или отдельных программных компонентов, а при необходимости и для сетевой среды, содержащей компоненты прикладного программного обеспечения на рабочих станциях.

Решение о том, для какого уровня системы целесообразно разрабатывать профили, зависит не только от масштабов системы, но и от особенностей области ее будущего применения, а также целей управления проектом, в рамках которого разрабатывается система.

Если, например, система создается для широкого круга потенциальных потребителей, то разработку профилей нужно начать с анализа возможных направлений использования системы и сред ее функционирования. Если же разработка ПС выполняется по заказу для конкретной категории пользователей, работающих в одном режиме, тогда можно ограничиться анализом перечня функций системы и частоты их использования.

С точки зрения управления проектом, если ПС невелика, разрабатывается небольшим коллективом, относительно независима от особенностей операционной системы, сети и других программных систем и не предполагается специального тестирования ее внешних интерфейсов, может быть разработан профиль для системы как единого целого, без выделения профилей программных компонентов, обеспечивающих эти интерфейсы. Если управление проектом должно осуществляться в условиях раздельной разработки (по времени разработки, месту разработки, коллективам разработчиков), а также раздельного тестирования, внедрения и сопровождения, то построение иерархических профилей – хороший способ локализации частей проекта.

Со времени своего появления профили получили распространение во многих областях как программной, так и системной инженерии в качестве основы для принятия решений в ходе выполнения процессов жизненного цикла программного обеспечения и систем. Применение профилей способствует:

– правильной расстановке приоритетов разработки, внедрения, эксплуатации и сопровождения ПС;

– систематизации исследования автоматизируемой предметной области и деловых процессов в организации-потребителе ПС;

– повышению степени соответствия спецификации как функциональных, так и технических требований к ПС, действительным потребностям пользователей;

– построению системы тестов и тестовых сценариев, максимально приближенных к реальным условиям среды функционирования ПС;

– определению рациональных конфигураций, версий и этапов внедрения компонентов ПС, начиная с наиболее востребованных пользователями;

– выполнению анализа производительности и надежности функционирования ПС, а также определению необходимых технических характеристик среды функционирования ПС (например, числа серверов сети), способных обеспечить требуемую производительность и надежность системы. Определению оптимальной системной архитектуры ПС;

– выделению критических функций и режимов работы и применению к ним надлежащих методов разработки и тестирования;

– совершенствованию структуры и порядка обучения потенциальных пользователей навыкам работы с ПС, улучшению документированности ПС;

– снижению стоимости разработки путем сокращения числа мало используемых функций, укрупнению операций, снижению числа маршрутов в операционных сценариях и обеспечению их прозрачности;

– рациональному распределению усилий по верификации и валидации компонентов проекта ПС;

– облегчению сопровождения ПС путем локализации функций и соответствующих операций в системе;

– совершенствованию процесса управления проектом ПС путем:

а) улучшения коммуникации между заказчиками и разработчиками, своевременного учета риска проекта ПС;

б) получения на предпроектных стадиях разработки ПС достоверных оценок размера и сложности ПС, а также затрат на ее разработку (путем применения технологий оценивания, основанных на учете пользовательского взгляда на функции ПС);

в) рационального распределения человеческих, финансовых и программно-технических ресурсов разработки.

Применение профилей позволяет не только повысить производительность разработки и надежность программных продуктов, но и снизить продолжительность их тестирования в 2 раза, сроки внедрения – на 30%, а стоимость сопровождения – в 10 раз [13].

Стоимость разработки профилей различна. По данным фирмы Hewlett-Packard, в среднем усилия на создание профилей для проекта из 10 разработчиков, содержащего 100 тыс. строк исходного кода и продолжительностью 18 месяцев, составляют 1 человек/месяц [4]. Затраты растут с ростом размера ПС, однако тот факт, что информация, собираемая для построения профилей, может быть использована по ходу разработки многообразно, оправдывает эти затраты.

11. Выводы

Таким образом, анализ состояния технологии и практики программной инженерии показывает, что профили применения ПС можно считать неотъемлемым элементом понятийного аппарата, лежащего в основе современной парадигмы качества программного обеспечения. Сфера их использования далеко не ограничивается рамками, определяемыми исходными целями создания:

обеспечение адекватности тестирования компьютерных систем реальным процессам эксплуатации и достижение максимальной достоверности оценок надежности компьютерных программ.

Представленная концепция профилей создает основу для дальнейшего упорядочения деятельности по разработке ПС, регламентируемой базовыми стандартами серии ГОСТ 34 и ГОСТ 19, а также новым стандартом ДСТУ 3918-99 «Процессы жизненного цикла программного обеспечения». Она позволяет структурировать разработку ПС в условиях ограниченных ресурсов с учетом условий применения как ПС в целом, так и ее отдельных компонентов, и концентрировать усилия разработчиков на наиболее часто используемых или уязвимых компонентах.

СПИСОК ЛИТЕРАТУРЫ

1. Мороз Г. Б., Коваль Г. И., Коротун Т. М. Определение целей и задач инженерии надежности программного обеспечения // Проблемы программирования. – 1997. – № 1. – С. 98 – 106.
2. Cheung R. A User-oriented Software Reliability Model // IEEE Trans. Soft. Eng. – 1980. – SE-6, N 2. – P. 11 – 125.
3. Musa J., Ianino A., Okumoto K. Software Reliability Measurement, Prediction, Application // McGraw-Hill. – 1987. – 370 p.
4. Musa J. Operational Profiles in Software Reliability Engineering // IEEE Software. – 1993. – Vol. 10, N 2. – P. 14 – 32.
5. Musa J. Sensitivity of field failure intensity to operational profile errors // Proc. 5th International Symposium on Software Reliability Engineering. – Monterey, CA. – 1994. – P. 334 – 337.
6. Whittaker J., Poore J. Markov Analysis of Software Specifications // ACM Trans. On Soft. Eng. Methodology. – 1993. – N 2. – P. 93 – 106.
7. Whittaker J., Thomason M. Markov chain model for statistical Software testing // IEEE Trans. Soft. Eng. – 1994. – SE-20, N 10. – P. 812 – 824.
8. Munson J. Software measurement: problems and practice // Annals of Software Engineering. – 1995. – Vol. 1, N1. – P. 255 – 285.
9. Munson J., Elbaum S. Software reliability as a function of user execution patterns // Proc. 32nd International Conference on System Sciences. – Hawaii. – 1999. – P. 1 – 12.
10. Runesson, P., Wohlin C. Usage Modelling: The Basis for Statistical Quality Control // Proc. 10th Annual Software Reliability Symposium. – Denver, Colorado. – 1992. – P. 77 – 84.
11. Андон Ф.И., Коваль Г.И., Коротун Т.М., Суслов В.Ю. Основы инженерии качества программных систем. – К.: Академпериодика, 2002. – 504 с.
12. Коваль Г. И. Методы определения размера ПО // Проблемы программирования. – 1999. – № 1. – С. 63 – 71.
13. Abramson S. Customer Specification-Based Product Development // Proc. International Switching Symp. – Yokohama, Japan. – 1992. – Vol. 2. – P. 65 – 69.