инженерия программного обеспечения

Компьютерные инструменты в образовании, 2020
№ 2: 66–79
УДК: 004.052
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-2-66-79

СОЗДАНИЕ ИЕРАРХИЧЕСКОЙ МОДЕЛИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПРОГРАММНЫХ СРЕДСТВ УЧЕБНОГО НАЗНАЧЕНИЯ

Гуров В. В.[1], кандидат технических наук, доцент, ✉ vvgurov@gmail.com, orcid.org/0000-0003-3503-1571

[1]Национальный исследовательский ядерный университет «МИФИ», Каширское ш., д. 31, 115409, Москва, Россия

Аннотация

В настоящее время в учебный процесс всех высших учебных заведений широко внедряются программные средства учебного назначения различного типа: от средств поддержки проведения лекций, практических и лабораторных занятий до оценки знаний студентов. Помимо программ, разрабатываемых и распространяемых (на платной или бесплатной основе) крупными компаниями, в каждом вузе широко распространена практика, когда ряд подобных программ пишется относительно небольшими коллективами собственных разработчиков, которые учитывают сложившуюся в данном вузе методику преподавания тех или иных дисциплин и в состоянии быстро откликаться на постоянно меняющиеся требования к учебному процессу.

В последнем случае перед разработчиками встает двуединая задача. С одной стороны, они должны создать необходимый продукт как можно быстрее, но, с другой стороны, он должен отвечать необходимым требованиям по качеству, в том числе по надежности.

Для оценки этого параметра применяются различные модели надежности. В частности, на ранних этапах создания программного модуля может быть использована модель Миллса. Одним из ее недостатков в рассматриваемой области является то, что для оценки достоверности результата, даваемого данной моделью, желательно знать ожидаемое начальное количество ошибок в программе.

Данное значение может быть получено с использованием простой интуитивной модели надежности программного обеспечения, которая не требует ведения сложного журнала наблюдений за ходом тестирования и не требует сложных вычислений.

В работе показывается, каким образом возможно совместить применение этих моделей в одну иерархическую модель, которая может быть эффективно использована в рассматриваемой предметной области.

Ключевые слова: *программные средства учебного назначения, надежность программного обеспечения, модели надежности, модель Миллса, простая интуитивная модель.*

Цитирование: Гуров В. В. Создание иерархической модели надежности программного обеспечения для программных средств учебного назначения // Компьютерные инструменты в образовании. 2020. № 2. С. 66–79. doi: 10.32603/2071-2340-2020-2-66-79

1. ВВЕДЕНИЕ

В современных условиях программные средства учебного назначения (ПСУН) для вузов играют чрезвычайно важную роль. Эта роль все более возрастает в связи с расширением перехода на дистанционную форму обучения, а также в связи со все большим привлечением к обучению (особенно в передовых вузах) иностранных и иногородних студентов. Такие программные средства включают в свой состав средства для проведения различного вида занятий и контроля усвоения знаний. Учебный процесс строится по так называемому «принципу ЗУН»: студент должен получить набор определенных *знаний* по данной дисциплине, у него должны сформироваться *умения* пользоваться этими знаниями, которые должны перерасти в *навыки*. ЗУН закладываются в процессе чтения лекций, проведения практических и лабораторных занятий и в последующем должны быть проверены соответствующими средствами контроля.

Программные системы, обеспечивающие проведение лекционных занятий, а также системы автоматизированного и удаленного контроля качества усвоения пройденного материала разрабатываются, как правило, крупными компаниями с большим штатом сотрудников и на протяжении нескольких лет [1–3]. Здесь же следует отметить ряд систем, разрабатываемых мировыми производителями программного обеспечения, которые, как правило, проводят обучение в своих сертифицированных центрах [4, 5].

Компьютерные обучающие программы, используемые на практических и лабораторных занятиях, как правило, отражают специфику дисциплины и используемую преподавателем методику. Поэтому они должны быть созданы быстро и с помощью того штата программистов, которым располагает данный научный коллектив (как правило, это базовая ячейка вуза — кафедра). Такая научная группа обычно возглавляется преподавателем, ответственным за данную дисциплину, и включает в себя аспирантов, магистров, студентов старших курсов. Цель работы такой группы — создать за заданное время программный продукт с приемлемым уровнем качества.

Качество программы определяется нормативными документами [6–8], включающими различный набор параметров и показателей качества, но все они содержат такой параметр, как надежность программы. Надежность во многом определяется количеством ошибок, содержащихся в программе к моменту ее сдачи в эксплуатацию. Получить полностью безошибочную программу невозможно. Даже в программах ответственного назначения при их сдаче в эксплуатацию остается 0,04...0,15 ошибок на 1000 строк кода [9, 10]. Это означает, что в таких программах, состоящих из миллионов кодовых строк, при передаче их в эксплуатацию остается несколько сотен ошибок. Не свободны от наличия ошибок и программные средства учебного назначения, даже если они разрабатываются известными компаниями [11].

Программные средства учебного назначения имеют меньшие объемы, но и их невозможно полностью избавить от ошибок. К тому же в случае такого рода программ большую роль играет фактор длительности их разработки. При отладке программ исправление ошибки обходится тем дороже, чем позже она выявлена: от 140 долларов на этапе разработки спецификаций до 140 000 долларов на этапе сопровождения [12]. Поэтому важную роль играет возможность исправления как можно большего количество ошибок на ранних этапах жизненного цикла программ.

Оценка количества ошибок, остающихся в программе после очередного этапа ее тестирования, может быть проведена на основе модели Миллса. Однако для её первоначального запуска желательно знать ожидаемое количество ошибок в тестируемой программе.

С этой целью в работе предлагается использовать простую интуитивную модель надежности программного обеспечения, которая, с одной стороны, достаточно проста с вычислительной точки зрения, а с другой стороны, позволяет оценить количество необходимых тестовых прогонов, для того чтобы в последующем использовать более сложные модели надежности.

2. МЕТОДИКА

2.1. Специфика программных средств учебного назначения

В ГОСТ 28195-89 «Оценка качества программных средств. Общие положения» [8] отмечается, что выбор номенклатуры показателей качества для конкретного ПС осуществляется с учетом его назначения и требований областей применения. Программные средства учебного назначения отнесены к группе «Прочие программные средства», выбор показателей качества для которых осуществляется в зависимости от их назначения с учетом требований областей применения. Именно из этого положения и будем исходить.

Программные средства характеризуются рядом параметров и показателей, но лишь четыре из них носят количественный характер:

— показатель устойчивости к искажающим воздействиям;
— оценка по среднему времени восстановления;
— вероятность безотказной работы;
— оценка по продолжительности преобразования входного набора данных в выходной.

Другие параметры и показатели оцениваются, как правило, экспертными методами.

Для программных средств учебного назначения наиболее важными количественными показателями, на наш взгляд, являются два последних.

Оценка по продолжительности преобразования входного набора данных в выходной для программ учебного назначения, в частности, означает, что определенные виды работ должны быть выполнены за отводимое под эти работы время, например за два академических часа. Возможность учета данного положения была подробно рассмотрена в [13].

Оценка вероятности безотказной работы этого класса программ должна учитывать следующее:

— разработка ПСУН должна быть выполнена в возможно более короткий срок, пока она не потеряла своей актуальности для данной дисциплины;
— программы учебного назначения, ориентированные на поддержку учебного процесса по определенной дисциплине или комплексу смежных дисциплин, создаются, как правило, коллективами разработчиков самих вузов;
— следует быть готовым к тому, что в программе может остаться определенное число ошибок, хотя она, в то же время, должна обеспечивать заданные требования по вероятности безотказной работы за определенный период времени;
— создаваемые ПСУН имеют относительно небольшой объем; так комплексы компьютерных обучающих программ по различным дисциплинам и система тестирования с политомическими заданиями и множественным вариантом правильных ответов, разработанные на кафедре «Компьютерные системы и технологии» НИЯУ МИФИ, имеют объем от 5 до 25 мегабайт каждый [14–17].

Ряд особенностей программных средств учебного назначения рассмотрен в [18, 19].

Эти положения приводят к тому, что коллективу разработчиков ПСУН должны быть предоставлены средства, позволяющие уже на ранних этапах проектирования программных средств оценить их надежность и тем самым сократить сроки ввода таких программ в эксплуатацию. Проблемам выбора модели надежности программного обеспечения, ориентированной на максимальное ускорение тестирования программ с целью их скорейшего ввода в эксплуатацию, посвящена, в частности, работа [20]. Однако в ней предлагается слишком сложный подход к решению этой задачи, который вряд ли может быть использован небольшим коллективом из аспирантов, магистров и студентов вуза. Некоторые подходы, более применимые на практике, рассматривались автором в работе [21].

2.2. Модель Миллса надежности программного обеспечения

Одной из наиболее простых моделей для восприятия и оценки таких важных показателей, как первоначальное количество ошибок в исходной программе и мера доверия к полученному результату, считается модель Миллса [22, 23]. Её использование предполагает следующую последовательность действий.

В исходную программу перед тестированием вносится L ошибок, наиболее характерных для данного класса задач и уровня квалификации программистов, написавших её.

Тестирование проводит другая группа программистов, которая не знает ни количества, ни характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Она находит внесённые ошибки, а заодно и некоторые из собственных ошибок программы, и по этим данным определяет количество ошибок, присутствовавших в программе перед началом тестирования (N), и меру доверия к полученному результату (C).

Предполагается, что вероятность нахождения всех ошибок (как изначально присутствовавших в программе, так и искусственно внесенных) одинакова.

В этом случае модель Миллса определяется следующими соотношениями:

$$N = \frac{L}{V} \times S, \tag{1}$$

$$C = \begin{cases} 1, & \text{если} \quad S \geqslant N, \\ \dfrac{L}{L+N+1}, & \text{если} \quad S < N, \end{cases} \tag{2}$$

где L — известное количество ошибок, вносимых в программу перед началом тестирования; V — количество внесённых ошибок, обнаруженных за время тестирования; S — количество первоначально присутствовавших в программе ошибок, которые были обнаружены за время тестирования.

Соотношения (1) и (2) образуют полезную модель ошибок: выражение (1) предсказывает возможное количество первоначально имевшихся в программе ошибок, а выражение (2) определяет меру доверия к модели, то есть вероятность правильного предсказания величины N, которая используется для установления доверительного уровня прогноза.

Пример

Пусть количество внесённых ошибок $L = 6$, количество найденных во время тестирования ошибок в исходной программе $S = 12$. Требуется определить предполагаемое

исходное число ошибок в программе N и меру доверия C при различном количестве найденных собственных ошибок V.

Полученные результаты сведены в таблицу 1.

Таблица 1

V	N	C
3	24	0,19
4	18	0,24
5	14	0,29

Обычно такие простейшие формулы, определяющие модель Миллса, ни в статьях на эту тему, ни в каких-либо пособиях не анализируются подробно. Считается, что особых вопросов они не вызывают. Однако, посмотрев на них более внимательно, увидим несколько интересных моментов, которые требуют более пристального изучения.

Очевидно, что если найдены все внесённые ошибки ($V = L$), то, в силу положенного в основу модели Миллса принципа пропорциональности, случай, когда все внесённые ошибки обнаружены, означает, что обнаружены и все собственные ошибки программы, то есть $N = S$. И весь вопрос состоит только в мере доверия к такой оценке.

Из формулы (2) видно, что значение C будет возрастать с увеличением значения L, то есть с увеличением количества искусственно внесённых ошибок.

В [24] нами было показано, что

— получение высокой достоверности требует первоначального внесения в программу очень большого количества искусственных ошибок;
— с целью практического использования данной модели для прогнозной оценки количества ошибок в исходной программе достаточно получить эту оценку с достоверностью $C = 0,5$;
— для получения значения первоначального количества ошибок в программе достаточно выявить не все «посеянные» ошибки, что может потребовать чрезвычайно большого времени тестирования, а лишь $L-1$ ошибку. Это обеспечивает приблизительно такую же достоверность результата, но практически вдвое сокращает время тестирования;
— модель Миллса с одной ненайденной «посеянной» ошибкой может эффективно использоваться на первоначальном этапе тестирования для оценки общего количества ошибок в исходной программе;
— предположение о начальном количестве ошибок в программе может существенно повлиять на конечный результат, так как при некотором его предположении мы в определённый момент можем получить прогноз N с достоверностью $C = 1$, в то же время при другом первоначальном прогнозе достоверность результата в данный момент будет существенно ниже;
— предположение о начальном значении количества ошибок в программе может проводиться чисто субъективно или на основе других моделей надёжности, однако влияние этого фактора на конечный результат требует дополнительного изучения.

В данной работе для первоначальной оценки ожидаемого количества ошибок в программе предлагается использовать простую интуитивную модель надежности программного обеспечения [22, 25].

2.3. Простая интуитивная модель надежности программного обеспечения

Простая интуитивная модель относится к статическим видам моделей надежности ПО, так как в ней не используются параметры времени тестирования и учитываются только результаты испытаний.

В этой модели используется тестирование двумя группами программистов (или двумя программистами в зависимости от величины программы), независимыми друг от друга.

В процессе тестирования каждая из групп фиксирует все найденные ею ошибки.

Пусть первая группа обнаружила n_1 ошибок, вторая n_2 ошибок, n_{12} — число ошибок, обнаруженных как первой, так и второй группами тестировщиков.

Обозначим через N неизвестное количество ошибок, присутствующих в программе до начала тестирования. Тогда можно эффективность тестирования каждой из групп определить как

$$E_1 = \frac{n_1}{N},$$

$$E_2 = \frac{n_2}{N}.$$

Эффективность тестирования можно интерпретировать как вероятность того, что ошибка будет обнаружена. Таким образом, можно считать, что первая группа обнаруживает ошибку в программе с вероятностью $p_1 = \frac{n_1}{N}$, вторая — с вероятностью $p_2 = \frac{n_2}{N}$. Тогда вероятность p_{12} того, что ошибка будет обнаружена обеими группами, можно принять равной $p_{12} = \frac{n_{12}}{N}$. С другой стороны, так как группы действуют независимо друг от друга, то $p_{12} = p_1 \cdot p_2$.

Получаем:

$$\frac{n_1}{N} \cdot \frac{n_2}{N} = \frac{n_{12}}{N}.$$

Это выражение позволяет нам получить оценку первоначального числа ошибок программы:

$$N = \frac{n_1 \cdot n_2}{n_{12}}. \tag{3}$$

Пример

В процессе тестирования программы 1-я группа нашла 15 ошибок, 2-я группа нашла 25 ошибок, общих ошибок было найдено 5. Определить исходное количество ошибок в программе по простой интуитивной модели.

Решение: $n_1 = 15; n_2 = 25; n_{12} = 5$. Отсюда:

$$N = \frac{n_1 \cdot n_2}{n_{12}} = \frac{15 \cdot 25}{5} = 75.$$

Несмотря на свою простоту, интересными представляются некоторые ситуации, возникающие при использовании данной модели, которые обычно не рассматриваются в литературе.

Рассмотрим некоторые из них.

1. Как определить первоначальное количество ошибок в программе, если первая группа тестировщиков выявила n_1 ошибок, вторая — n_2 ошибок, но общих ошибок выявлено не было?

В данном случае возможны два пути оценки результата.

Прямое использование формулы (3) здесь невозможно, так как $n_{12} = 0$.

В этом случае мы получаем чисто математически бесконечное число ошибок в исходной программе.

Другой реальный путь говорит о том, что первоначально в программе было не менее чем $N = (n_1 + n_2)$ ошибок. Хотя этот вывод, на первый взгляд, и кажется очевидным, он позволяет использовать полученное значение в качестве первоначального приближения, например, в модели Миллса.

Также в этом случае, возможно, тестирование следует продолжить до тех пор, пока тестировщиками не будет выявлено достаточное количество общих ошибок. При этом встает вопрос достаточности, который требует отдельного рассмотрения.

Очевидно, что график предполагаемого исходного количества ошибок в программе в зависимости от числа общих найденных ошибок двумя группами тестировщиков имеет вид, представленный на рис. 1.

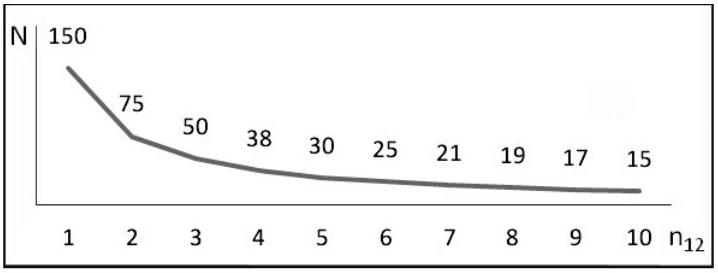

Рис. 1. График зависимости количества первоначальных ошибок в программе N от количества общих ошибок n_{12}, найденных двумя группами тестировщиков ($n_1 = 10, n_2 = 15$)

В общем виде процесс изучения надежности программы по простой интуитивной модели сводится к последовательному нахождению ошибок в ней двумя коллективами тестировщиков, их устранению, а затем новому тестированию программы с исправленными ошибками, найденными на предыдущем этапе. При этом будем считать, что при исправлении найденных ошибок новые ошибки в программу не вносятся.

Количество ошибок, оставшихся в программе после исправления всех найденных на первом этапе тестирования ошибок, составит $N^{(2)} = N^{(1)} - (n_1 + n_2) + n_{12}$.

Считая, что тестировщики будут работать на втором этапе с той же эффективностью, что и на первом, получим, что на втором этапе первая группа найдет

$$n_1^{(2)} = E_1 \cdot N^{(2)} = \frac{n_1}{N^{(1)}} \cdot \left(N^{(1)} - (n_1 + n_2) + n_{12}\right) \text{ ошибок,}$$

а вторая

$$n_2^{(2)} = E_2 \cdot N^{(2)} = \frac{n_2}{N^{(1)}} \cdot \left(N^{(1)} - (n_1 + n_2) + n_{12}\right) \text{ ошибок.}$$

При этом общих ошибок будет найдено

$$n_{12}^{(2)} = E_1 \cdot E_2 \cdot N^{(2)} = \frac{n_1 \cdot n_2}{N^{(1)} \cdot N^{(1)}} \cdot \left(N^{(1)} - (n_1 + n_2) + n_{12}\right).$$

Тогда можно сказать, что согласно формуле (3) перед началом второго этапа тестирования в программе находилось

$$N^{(2)} = \frac{n_1^{(2)} \cdot n_2^{(2)}}{n_{12}^{(2)}} = \frac{\frac{n_1}{N^{(1)}} \cdot N^{(2)} \cdot \frac{n_2}{N^{(1)}} \cdot N^{(2)}}{\frac{n_1 \cdot n_2}{N^{(1)} \cdot N^{(1)}} \cdot N^{(2)}} = N^{(1)} - (n_1 + n_2) + n_{12} =$$

$$= \frac{n_1 \cdot n_2}{n_{12}} - (n_1 + n_2) + n_{12} = \frac{(n_1 - n_{12}) \cdot (n_2 - n_{12})}{n_{12}} \text{ ошибок.}$$

После исправления всех найденных на втором этапе тестирования ошибок в программе останется

$$N^{(2)} = N^{(2)} - (E_1 + E_2) \cdot N^{(2)} + E_1 \cdot E_2 \cdot N^{(2)} = N^{(2)} \cdot \left(1 - \frac{n_1 + n_2}{n_{12}} + \frac{n_1 \cdot n_2}{n_{12}^2}\right) = \frac{(n_1 - n_{12})^2 \cdot (n_2 - n_{12})^2}{n_{12}^2}$$

ошибок.

Рассмотрев этот процесс далее, получим, что после выполнения i-го шага тестирования в программе согласно простой интуитивной модели останется:

$$N^{(i)} = \frac{(n_1 - n_{12})^{i-1} \cdot (n_2 - n_{12})^{i-1}}{n_{12}^i} \tag{4}$$

ошибок.

Простая интуитивная модель не требует ведения сложных журналов наблюдений и сложных вычислений. Но, исходя из полученной формулы, можно получить несколько важных и полезных результатов.

Во-первых, можно вычислить необходимое количество шагов (i) для определения значения N, которое будет использоваться другими более сложными, моделями надежности программного обеспечения (например моделью Миллса) в качестве начального приближения. Решая полученное уравнение относительно i, получим:

$$i = \frac{\ln\left(N \cdot (n_1 - n_{12}) \cdot (n_2 - n_{12})\right)}{\ln \frac{(n_1 - n_{12}) \cdot (n_2 - n_{12})}{n_{12}}}.$$

Зная результаты первого этапа тестирования (n_1, n_2, n_{12}) и необходимое значение N для запуска более точной модели оценки надежности ПО, можно определить количество этапов тестирования по простой интуитивной модели.

Так, если на первом этапе тестирования получены значения $n_1 = 15$, $n_2 = 10$, $n_{12} = 5$, то, считая необходимым получить значение $N = 100$, для того чтобы использовать его в дальнейшем, получим, что для этого потребуется примерно 4 этапа тестирования с оценкой результатов по простой интуитивной модели.

Во-вторых, можно определить необходимое количество шагов i, при котором $n_{12} \geqslant 1$, что является необходимым условием применения простой интуитивной модели. Для этого следует преобразовать выражение (4) относительно n_{12}. Выполнить это аналитически затруднительно, что возникает необходимость использования численных методов решения.

Рассмотрим еще один случай использования простой интуитивной модели, который тоже обычно не рассматривается в литературе.

Пусть в процессе тестирования программы задействовано не две, а три независимые группы тестировщиков, каждая из которых выявила n_1, n_2 и n_3 ошибок соответственно. Причем первая и вторая, первая и третья, вторая и третья группы выявили n_{12}, n_{13} и n_{23} общих ошибок соответственно, а общих ошибок, выявленных всеми тремя группами, было n_{123}. Как в данном случае интерпретировать полученный результат?

Очевидно, что если вычислять N по результатам работы 1-й и 2-й, 1-й и 3-й, а также 2-й и 3-й групп, то их значения, в общем случае, не совпадут.

Пример

Пусть $n_1 = 12, n_2 = 8, n_3 = 10, n_{12} = 4, n_{13} = 4, n_{23} = 5, n_{123} = 2$.

Тогда $N_{12} = n_1 \cdot n_2 / n_{12} = 24$ — количество ошибок, рассчитанное на основе общих ошибок 1-й и 2-й групп.

$N_{13} = n_1 \cdot n_3 / n_{13} = 30$ — количество ошибок, рассчитанное на основе общих ошибок 1-й и 3-й групп.

$N_{23} = n_2 \cdot n_3 / n_{23} = 16$ — количество ошибок, рассчитанное на основе общих ошибок 2-й и 3-й групп.

$N_{123} = n_1 \cdot n_2 \cdot n_3 / n_{123} = 22$ — количество ошибок, рассчитанное на основе общих ошибок 1-й, 2-й и 3 й групп.

Величину N_{123} определяли, исходя из рассуждений, аналогичных приведенным выше:

$$\frac{n_1}{N} \cdot \frac{n_2}{N} \cdot \frac{n_3}{N} = \frac{n_{123}}{N}.$$

Отсюда следует, что

$$N = \sqrt{\frac{n_1 \cdot n_2 \cdot n_3}{n_{123}}}. \quad (5)$$

Очевидно, что подобный подход может быть применен и к большему количеству групп. При этом в полученной формуле (5) изменится лишь степень корня для получения значения N.

В то же время нужно учитывать, что при тестировании уже было в данном случае найдено $n_1 + n_2 + n_3 - n_{12} - n_{13} - n_{23} = 12 + 8 + 10 - 4 - 4 - 5 = 17$ ошибок, и количество ошибок в исходной программе не может быть меньше этой величины.

Следовательно, для дальнейших расчетов нужно исходить из того, что в исходной программе имелось количество ошибок, равное максимальному значению из получившихся значений N, но не меньшее, чем количество ошибок, обнаруженных при тестировании.

В рассматриваемом случае эта величина будет равна $N = N_{13} = 30$.

Полученное значение величины N может быть использовано в последующем в качестве начального приближения для анализа надёжности программы с помощью других моделей надежности программного обеспечения.

3. РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

В данной работе были рассмотрены особенности программных средств учебного назначения и процесса их создания силами коллективов высших учебных заведений.

Определены основные особенности программ такого типа, к главным из которых отнесены относительно небольшой объем, создание их силами сотрудников и студентов высшего учебного заведения в максимально короткие сроки при приемлемом уровне их надежности.

Такие положения требуют оценки надежности создаваемых программ на наиболее ранних этапах их жизненного цикла. Для этого могут быть использованы различные модели надёжности программного обеспечения. Одна из них, модель Миллса, позволяет оценить количество ошибок в программе и достоверность этой оценки, но для ее нормального функционирования требуется предварительная оценка ожидаемого количества ошибок в программе.

В работе предлагается для получения этой оценки использовать простую интуитивную модель надежности ПО. Проведенный анализ этой модели позволил прояснить те места, на которые обычно обращают мало внимания при ее рассмотрении. Определено количество этапов тестирования разрабатываемой программы, которое следует провести, анализируя результаты по простой интуитивной модели с тем, чтобы определить количество ошибок в программе для начального запуска анализа надежности с помощью модели Миллса. Также даны рекомендации по определению минимального количества общих ошибок, которые должны быть выявленны двумя группами программистов, проводящих независимое тестирование программы, при котором данная модель применима.

Развитие работ по исследованию простой интуитивной модели надежности программного обеспечения предполагает исследование на возможность использования ее как первоосновы в иерархическом построении моделей надежности в случаях, когда на следующих уровнях иерархии используются более сложные модели надёжности ПО: модель Шумана, модель Джелинского-Моранды и другие.

Также следует дополнительно рассмотреть вопрос, касающийся возможного внесения новых ошибок при устранении тех ошибок, которые были выявлены на очередном этапе тестирования.

Благодарности. Автор благодарит заведующего кафедрой «Компьютерные системы и технологии» НИЯУ «МИФИ» доктора технических наук профессора Иванова Михаила Александровича за многолетнюю помощь в проводимых автором научных исследованиях в области исследования и обеспечения качества программного обеспечения.

Список литературы

1. 35 крупнейших EdTech-компаний России: рейтинг РБК [Электронный ресурс]. URL: https://www.rbc.ru/trends/education/5d68e8fb9a7947360f1e2e52 (дата обращения: 20.05.2020).
2. Ассоциация «Национальная платформа открытого образования» [Электронный ресурс]. URL: https://openedu.ru/ (дата обращения: 20.05.2020).
3. Академия Росатома [Электронный ресурс]. URL: http://rosatom-academy.ru/svedeniia-ob-obrazovatelnoi-organizatsii/obrazovanie/ (дата обращения: 20.05.2020).
4. Центры обучения Microsoft [Электронный ресурс]. URL: https://www.microsoft.com/ru-ru/learning/training.aspx (дата обращения: 20.05.2020).
5. Образовательные услуги HP в России [Электронный ресурс]. URL: https://education.hpe.com/ru/ru/training/index.html (дата обращения: 20.05.2020).
6. ГОСТ Р ИСО/МЭК 9126-93 Оценка программной продукции. Характеристики качества и руководства по их применению [Электронный ресурс]. URL: https://gostbank.metaltorg.ru/gost/7323/ (дата обращения: 20.05.2020).
7. ГОСТ 28806-90. Качество программных средств. Термины и определения. [Электронный ресурс]. URL: https://gostbank.metaltorg.ru/gost/7316/ (дата обращения: 20.05.2020).
8. ГОСТ 28195-89. Оценка качества программных средств. Общие положения. [Электронный ресурс]. URL: http://docs.cntd.ru/document/1200009135 (дата обращения: 20.05.2020).
9. *Липаев В. В.* Анализ и сокращение рисков проектов сложных программных средств. М.: Синтег. 2005. 208 с.
10. *Нупур Д, Хамфри У., Редвайн С., Цибульски Г. и др.* Процессы разработки безопасного программного обеспечения // Открытые системы. 2004. № 8. С. 49–58.
11. 7 самых удобных платформ для онлайн-обучения [Электронный ресурс]. URL: https://zen.yandex.ru/media/id/5cb4718903be7000b456ae54/7-samyh-udobnyh-platform-dlia-onlainobucheniia-5cd108d800fcb600afbec599 (дата обращения: 20.05.2020).
12. *Кирьянчиков В. А., Опалева Э. А.* Качество и надежность программного обеспечения Санкт-

Петербургский государственный электротехнический университет «ЛЭТИ» Санкт-Петербург 2002 [Электронный ресурс]. URL: https://studfile.net/preview/913486/page:23 (дата обращения: 20.05.2020).
13. *Gurov V. V.* The probability estimation of the electronic lesson implementation taking into account software reliability // AIP Conference Proceedings. 2017. Vol. 1797. № 1. P. 030005. doi: 10.1063/1.4972444 (7 стр. p.p. 030005-1 – 030005-7).
14. *Гуров В. В., Гуров Д. В., Кузнецова П. В., Михайлов Д. М.* Программа «Интерактивная система тестирования на основе компьютерных технологий ИСТОК». Свидетельство об официальной регистрации программы для ЭВМ № 2006613218. Зарегистрировано в реестре программ для ЭВМ 13.09.2006.
15. *Гуров В. В., Гуров Д. В., Кузнецова П. В., Михайлов Д. М.* Пакет "Physics-МЕРНI"компьютерных обучающих программ по курсу «Общая физика». Свидетельство об официальной регистрации программы для ЭВМ № 2006613219. Зарегистрировано в реестре программ для ЭВМ 13.09.2006.
16. *Гуров В. В., Гуров Д. В., Криводаев А. В., Матюшенков Н. В.* Пакет компьютерных обучающих программ по курсу «Организация ЭВМ». Свидетельство об официальной регистрации программы для ЭВМ № 2007611495. Зарегистрировано в реестре программ для ЭВМ 10.04.2007.
17. *Гуров В. В., Вологдин Е. В., Кузнецова П. В., Михайлов Д. М.* Пакет компьютерных обучающих программ по курсу «Защита информации в компьютерных системах и сетях». Свидетельство об официальной регистрации программы для ЭВМ № 2007611496. Зарегистрировано в реестре программ для ЭВМ 10.04.2007.
18. *Гуров В. В.* Исследование системы тестирования на основе усовершенствованной модели частичного оценивания // Современные технологии и задачи управления, автоматики и обработки информации: Труды XVI Международного научно-технического семинара. 2007. Тула: ТулГУ, 2007. С. 254–255.
19. *Гуров В. В., Гуров Д. В.* Использование электронных образовательных ресурсов в учебном процессе технического вуза // Информационные технологии в обеспечении нового качества высшего образования. Сборник научных статей. Книга 3. Труды Всероссийской научно-практической конференции с международным участием. М.: Исследовательский центр проблем качества подготовки специалистов. 2010. С. 53–58. [Электронный ресурс]. URL: https://bookre.org/reader?file=803395 (дата обращения: 20.05.2020).
20. *Li X., Xie M., Ng S. H.* Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points // Applied Mathematical Modelling, Vol. 34. Is. 11.P. 3560–3570. doi: 10.1016/j.apm.2010.03.006
21. *Гуров В. В.* Оценка надёжности программного обеспечения на ранних этапах жизненного цикла // Вестник НИЯУ МИФИ, ООО МАИК «НАУКА/ИНТЕРПЕРИОДИКА»., 2012. Т. 1. Вып. 2. С. 245–250.
22. *Майерс Г.* Надежность программного обеспечения [Текст] М.: Мир, 1980.
23. Оценка количества ошибок в программе. Модель Миллса. [Электронный ресурс]. URL: https://habr.com/ru/post/122912/ (дата обращения: 20.05.2020).
24. *Гуров В. В.* Практические особенности использования моделей надежности программного обеспечения // Вестник НИЯУ МИФИ. 2017. Т. 6. № 5. С. 458–465. doi: 10.1134/S2304487X17050030
25. Оценка количества ошибок в программе. Парная оценка. [Электронный ресурс]. URL: http://habrahabr.ru/post/123473/ (дата обращения: 20.05.2020).

Поступила в редакцию 20.05.2020, окончательный вариант — 22.06.2020.

Гуров Валерий Валентинович, кандидат технических наук, доцент кафедры «Компьютерные системы и технологии» Института интеллектуальных кибернетических систем НИЯУ «МИФИ», ✉ vvgurov@gmail.com

Computer tools in education, 2020
№ 2: 66–79
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-2-66-79

Creating a Hierarchical Model of Software Reliability for Educational Software

Gurov V. V.[1], PhD, associate professor, ✉ vvgurov@gmail.com,
orcid.org/0000-0003-3503-1571

[1]National research nuclear University "MEPhI", 31, Kashirskoe hwy, 115409, Moscow, Russia

Abstract

Currently, various types of educational software are widely introduced into the educational process of all higher education institutions, from lecture support tools, practical and laboratory classes to assessing students' knowledge. In addition to programs developed and distributed (for a fee or free of charge) by large companies, each University has a widespread practice when a number of such programs are written by relatively small teams of their own developers, who take into account the existing methods of teaching certain disciplines in this university and are able to respond quickly to constantly changing requirements for the educational process.

In the latter case, developers face a two-fold task. On the one hand, they need to create the necessary product as quickly as possible, but on the other hand, it must meet the necessary quality requirements, including reliability.

Various reliability models are used to evaluate this parameter. In particular, the Mills model can be used at the early stages of creating a software module. One of its disadvantages in this area is that in order to assess the reliability of the result given by this model, it is desirable to know the expected initial number of errors in the program.

This value can be obtained using a simple intuitive software reliability model that does not require a complex log of monitoring the progress of testing and does not require complex calculations. The paper shows how it is possible to combine the use of these models into a single hierarchical model that can be effectively used in the subject area under consideration.

Keywords: *educational software, software reliability, Mill's error seeding model, simple intuitive software reliability model.*

Citation: V. V. Gurov, "Creating a Hierarchical Model of Software Reliability for Educational Software," *Computer tools in education*, no. 2, pp. 66–79, 2020 (in Russian); doi:10.32603/2071-2340-2020-2-66-79

References

1. D. Ryzhkova, M. Aranovskaya, I. Reikhard, and S. Vysokikh, "35 largest EdTech companies in Russia: RBC rating," in *RBC.ru*, 2019 (in Russian). [Online]. Available: https://www.rbc.ru/trends/education/5d68e8fb9a7947360f1e2e52
2. *Natsional'naya platforma otkrytogo obrazovaniya* [National Platform for Open Education] (in Russian). [Online]. Available: https://openedu.ru/https://openedu.ru/

3. State Atomic Energy Corporation Rosatom, “Information about the educational organization,” in *Rosatom Academy* (in Russian). [Online]. Available: http://rosatom-academy.ru/svedeniia-ob-obrazovatelnoi-organizatsii/obrazovanie/
4. Microsoft Corp., “Browse Instructor-led Courses,” in *Microsoft Education* (in Russian). [Online]. Available: https://docs.microsoft.com/en-us/learn/certifications/courses/browse/
5. Hewlett Packard Ent., “HPE Education Services — Russia,” in *Education HPE* (in Russian). [Online]. Available: https://education.hpe.com/ru/ru/training/index.html
6. V. P. Ogurtsov et. al., eds., “GOST P ISO/MEK 9126-93, Otsenka programmnoi produktsii. Kharakteristiki kachestva i rukovodstva po ikh primeneniyu” [Evaluation of software products. Quality characteristics and guidelines for their application], in *Metaltorg* (in Russian). [Online]. Available: https://gostbank.metaltorg.ru/gost/7323/
7. “GOST 28806-90, Kachestvo programmnykh sredstv. Terminy i opredeleniya” [The quality of software. Terms and Definitions], in *Metaltorg* (in Russian). [Online]. Available: https://gostbank.metaltorg.ru/gost/7316/
8. Yu. P. Galustian et. al., eds., “GOST 28195-89, Otsenka kachestva programmnykh sredstv. Obshchie polozheniya” [Assessment of the quality of software. General Provisions], in *Tehexpert* (in Russian). [Online]. Available: http://docs.cntd.ru/document/1200009135
9. V. V. Lipaev, *Analiz i sokrashchenie riskov proektov slozhnykh programmnykh sredstv* [Analysis and reduction of risks of complex software projects], Moscow: Sinteg, 2005 (in Russian).
10. N. Davis, W. Humphrey, S. Redwine, G. Zibulski, and G. Mcgraw, “Processes for Producing Secure Software,” *Open Systems. DBMS*, no. 8, pp. 49–58, 2004 (in Russian).
11. User Info.Smail, “The 7 most convenient online learning platforms,” in *Yandex.Dzen* (in Russian). [Online]. Available: https://zen.yandex.ru/media/id/5cb4718903be7000b456ae54/7-samyh-udobnyh-platform-dlia-onlainobucheniia-5cd108d800fcb600afbec599
12. V. A. Kir’yanchikov and E. A. Opaleva, “Kachestvo i nadezhnost’ programmnogo obespecheniya” [Software quality and reliability], in *StudFiles*, 2002 (in Russian). [Online]. Available: https://studfile.net/preview/913486/page:23
13. V. V. Gurov, “The probability estimation of the electronic lesson implementation taking into account software reliability,” in *AIP Conference Proc.*, vol. 1797, no. 1, 2017, p. 030005; doi: 10.1063/1.4972444
14. V. V. Gurov, D. V. Gurov, P. V. Kuznetsova, and D. M. Mikhailov, *Interaktivnaya sistema testirovaniya na osnove komp’yuternykh tekhnologii ISTOK*. [Soft]. [Certificate of official registration of the computer program No. 200613218. Registered in the register of computer programs 13.09.2006].
15. V. V. Gurov, D. V. Gurov, P. V. Kuznetsova, and D. M. Mikhailov, *Package “Physics-MEPHI” of computer training programs for the course “General Physics”*. [Soft]. [Certificate of official registration of the computer program No. 2006613219. Registered in the register of computer programs 13.09.2006].
16. V. V. Gurov, D. V. Gurov, A. V. Krivodaev, and N. V. Matyushenkov, *Package of computer training programs for the course “Organization of Computer”*. [Soft]. [Certificate of official registration of the computer program No. 2007611495. Registered in the register of computer programs 10.04.2007].
17. V. V. Gurov, E. V. Vologdin, P. V. Kuznetsova, and D. M. Mikhailov, *Package of computer training programs for the course “Information protection in computer systems and networks”*. [Soft]. [Certificate of official registration of the computer program No. 2007611496. Registered in the register of computer programs 10.04.2007].
18. V. V. Gurov, “Issledovanie sistemy testirovaniya na osnove usovershenstvovannoi modeli chastichnogo otsenivaniya” [Investigating a Testing System Based on an Improved Partial Grading Model], *Sovremennye tekhnologii i zadachi upravleniya, avtomatiki i obrabotki informatsii: Trudy XVI Mezhdunarodnogo nauchno-tekhnicheskogo seminara*, Tula State University, Tula, Russia, 2007, pp. 254–255 (in Russian).
19. V. V. Gurov and D. V. Gurov, “Ispol’zovanie elektronnykh obrazovatel’nykh resursov v uchebnom protsesse tekhnicheskogo vuza” [The use of electronic educational resources in the educational process of a technical university], in *Sbornik nauchnykh statei. Trudy Vserossiiskoi nauchno-prakticheskoi konferentsii s mezhdunarodnym uchastiem ’Informatsionnye tekhnologii v obespechenii novogo kachestva vysshego obrazovaniya’. Book 3*, Moscow: Issledovatel’skii tsentr problem kachestva podgotovki spetsialistov, 2010, pp. 53–58 (in Russian). [Online]. Available: https://bookre.

org/reader?file=803395
20. X. Li, M. Xie, and S. H. Ng, “Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points,” *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3560–3570, 2010; doi: 10.1016/j.apm.2010.03.006
21. V. V. Gurov, “Otsenka nadezhnosti programmnogo obespecheniya na rannikh etapakh zhiznennogo tsikla” [Assessment of software reliability in the early stages of the life cycle], *Vestnik natsional’nogo issledovatel’skogo yadernogo universiteta “MIFI”*, vol. 1, no. 2, pp. 245–250, 2012 (in Russian).
22. G. J. Myers, *Software reliability*, Moscow: Mir, 1980 (in Russian).
23. User WatsOne, “Estimation of the number of errors in the program. Mills model,” in *Habr* (in Russian). [Online]. Available: https://habr.com/ru/post/122912/
24. V. V. Gurov, “Prakticheskie osobennosti ispol’zovaniya modelei nadezhnosti programmnogo obespecheniya” [Practical features of using software reliability models], *Vestnik natsional’nogo issledovatel’skogo yadernogo universiteta “MIFI”*, vol. 6, no. 5, pp. 458–465, 2017 (in Russian); doi: 10.1134/S2304487X17050030
25. User WatsOne, “Estimation of the number of errors in the program. Pair estimate,” in *Habr* (in Russian). [Online]. Available: https://habr.com/ru/post/123473/https://habr.com/ru/post/123473/

Received 20.05.2020, the final version — 22.06.2020.

Valery V. Gurov, PhD, associate professor of the Department “Computer systems and technologies” Institute of intelligent cybernetic systems, national research nuclear University “MEPhI”, ✉ vvgurov@gmail.com