# *Intellectual Technologies on Transport*
# *No 1*

# *Интеллектуальные технологии на транспорте*
# *№ 1*

*Санкт-Петербург*
*St. Petersburg*
*2016*

**Интеллектуальные технологии на транспорте**
**№ 1, 2016**

Сетевой электронный научный журнал, свободно распространяемый через Интернет.
Публикует статьи на русском и английском языках с результатами исследований и практических достижений в области интеллектуальных технологий и сопутствующих им научных исследований

Журнал основан в 2015 году

**Учредитель и издатель**

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Петербургский государственный университет путей сообщения Императора Александра I» (ФГБОУ ВПО ПГУПС)



Адрес редакции
190031 Санкт-Петербург, Московский пр., 9, ПГУПС
email: itt-pgups@yandex.ru, сайт: http://itt-pgups.ru/

ISSN 2413-2527

Журнал зарегистрирован Федеральной службой по надзору в сфере связи и массовых коммуникаций, свидетельство ЭЛ №ФС77-61707 от 07 мая 2015 г.

Журнал зарегистрирован в Российском индексе научного цитирования (РИНЦ)

**Intellectual Technologies on Transport**
**Issue № 1, 2016**

Network electronic scientific journal, open access. It publishes articles in Russian and English with the results of research and practical achievements in the field of intelligent technologies and associated research

Founded in 2015

**Founder and Publisher**

Federal State Educational Institution of Higher Professional Education
«Emperor Alexander I Petersburg State Transport University»



Adress
190031, St. Petersburg, Moskovskiy pr., 9, 2–108
email: itt-pgups@yandex.ru, http://itt-pgups.ru/

ISSN 2413-2527

The journal is registered by the Federal Service for Supervision of Communications and Mass Media, EL №FS77-61707 testimony from May 7, 2015

The journal is registered in the Russian Science Citation Index (RSCI)

# Содержание

# Contents

# Стохастический процессор для определения экстремума функции регрессии

Свистунов С. Г.
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия,
ssg47@mail.ru

***Аннотация.*** **В измерительной технике, радиолокации, гидроакустике часто требуется решить адаптивную задачу идентификации, т. е. построить модель по имеющимся данным об объекте. Примерами могут служить задачи статистической оценки параметров, регрессии, робастное оценивание, рекуррентное оценивание, анализ данных и т. д. В статье предлагается использовать для решения указанной задачи адаптивный квазиградиентный алгоритм, реализуемый на стохастическом вычислительном устройстве. Предлагаемый подход дает возможность сократить время решения и упростить используемые аппаратные средства. В статье приведен подробный алгоритм функционирования специализированного вычислительного устройства.**

***Ключевые слова:*** **экстремум функции регрессии, адаптивные алгоритмы стохастической оптимизации, стохастические вычислительные устройства, преобразователь код-вероятность, ПКВ, СтВУ.**

## Введение

В измерительной технике, радиолокации, гидроакустике часто требуется решить адаптивную задачу идентификации [1], т. е. построить модель по имеющимся данным об объекте. Примерами могут служить задачи статистической оценки параметров, регрессии, робастное оценивание, рекуррентное оценивание, анализ данных и т. д.

Решение этих задач можно свести к алгоритму поиска экстремума функции регрессии следующим образом [1]: для задач оценки параметров $x^*$ плотности распределения вероятностей $w(z, x)$ по реализациям случайной величины $z$ вводится функция

$$f(x) = MQ(z,x) = \int Q(z,x)w(z,x^*)dz,$$

где $M$ – математическое ожидание, $Q(z,x) = -\ln[w(z,x)]$.

Очевидно, что в точке $x^*$ минимум функции $f(x)$:

$$f(x) = -\int\left[\ln w(z,x^*)\right]w(z,x^*)dz -$$
$$-\int \ln\{1+[w(z,x)-w(z,x^*)]/\,w(z,x^*)\}w(z,x^*)dz \geq$$
$$\geq f(x^*) - \int\left[w(z,x) - w(z,x^*)\right]dz = f(x^*),$$

поскольку $-\ln(1+a) \geq -a$.

Пусть измерения $z^1, \ldots, z^s$ поступают последовательно, $x^s$ – найденное после обработки $s$ измерений приближение для $x^*$. Тогда можно считать $\nabla Q(z^{s+1}, x^s)$ приближением для $\nabla f(x^s) = M\nabla_x Q(z,x^*)$ и применять метод градиентного типа для минимизации $f(x)$:

$$x^{s+1} = x^s - \rho_s \nabla_x Q(z^{s+1}, x^s) =$$
$$= x^s - \rho_s\left[\nabla_x w(z^{s+1}, x^s)\right]/\,w(z^{s+1}, x^s).$$

Скорость сходимости порядка $O(1/s)$, где $O(b)$ – величина одного порядка малости с $b$.

## Оценка дисперсии сигнала с нулевым средним

Для оценки дисперсии сигнала с нулевым средним можно использовать следующую рекуррентную формулу [2]:

$$\sigma^2_{s+1} = \sigma^2_s + \rho_s(z^2_s - \sigma^2_s),$$

обычно $\rho_s = 1/s$ [3].

Дисперсия с нулевым средним – это математическое ожидание квадрата амплитуды сигнала. Приведенная выше формула требует вычисления $z^2_s$, шаговый множитель $\rho_s = 1/s$ является детерминированным. Для измерения доступна величина $z_s$, не составляет труда определить $|z| = (\mathrm{sign}(z)\sqrt{|z|})^2 = y^2$, где sign(z) – знак $z$.

Полагая функцию плотности распределения $w(z,x)$ четной, получим [4]

$$My = \int(\mathrm{sign}(z)\sqrt{|z|})w(z,x)dz = 0,$$

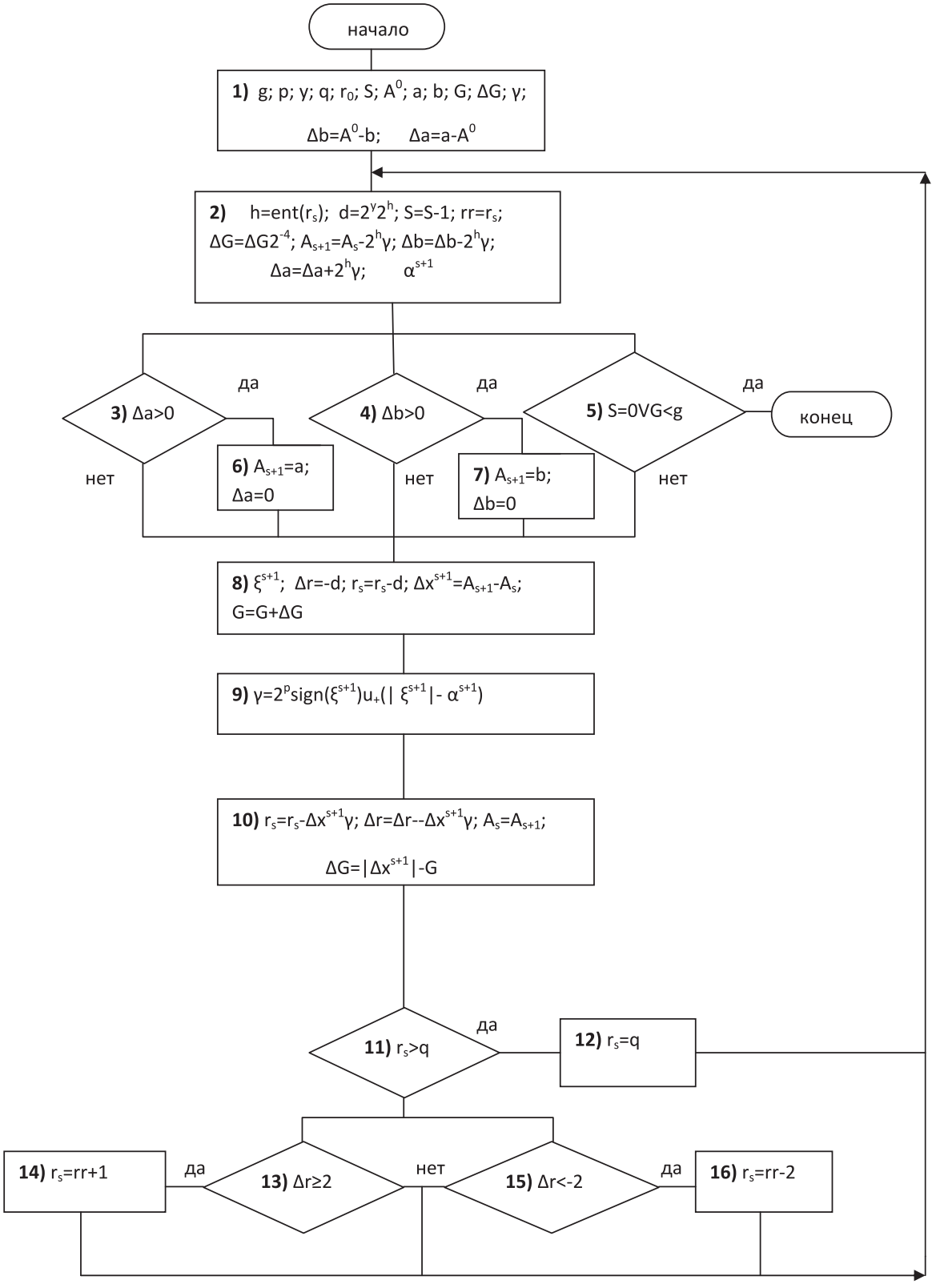
т. к. $\mathrm{sign}(z)\sqrt{|z|}$ – нечетная функция.

Таким образом, измеряя дисперсию величины $y$, т. е. $My^2 = M|z|$, мы измерим $M|z|$, где $|z|$ – амплитуда сигнала. Обозначим $M|z| = A$, тогда $A_{s+1} = A_s + \rho_s(|z_s| - A_s)$, где $A_s$ – оценка $A$ на шаге $s$. Осталось определить величину $\rho_s$. Этот пример показывает, что шаговый множитель $\rho_s = 1/s$ является детерминированным и не учитывает в должной мере расстояния до $x^*$. Следовательно, для ускорения сходимости вычислительного процесса требуется адаптивная подстройка шагового множителя $\rho_s$, т. е. необходимо использовать адаптивные алгоритмы стохастической оптимизации [5].

В общем случае шаговый множитель определяется по формулам:

$$h = -(A_s - |z_s|)(A_s - A_{s-1}) - \delta\rho_{s-1};$$
$$\rho_s = \min\left[\rho_0, \rho_{s-1}a^h\right],$$

где $\rho_0 > 0$, $\delta > 0$, $a > 1$. Вычисление $\rho_s$ требует таких «трудоемких» операций, как умножение и возведение в степень вещественных чисел. Наличие случайных сигналов на входе позволяет отказаться от традиционных вычислительных устройств, где информация представлена многоразрядными двоичными числами, и перейти к использованию стохастических вычислительных устройств (СтВУ), где входная информация преобразуется в импульсы, вероятностные характеристики которых зависят от входного сигнала. В результате преобразования код-вероятность (ПКВ) значительно упрощается аппаратура и уменьшается время выполнения отдельной итерации [6].




Алгоритм адаптивной квазиградиентной оптимизации

Пусть $\xi^s = A_s - |z_s|$ и $|\xi^s| < c = 2^p$ почти наверное (п. н.), тогда на выходе ПКВ имеем $\text{sign}(\xi s)u + (|\xi s| - \alpha S)$, где $u_+$ – функция Хевисайда; α – равномерно распределенная случайная величина в диапазоне [0, c].

Используя общие теоремы о сходимости рекуррентных стохастических алгоритмов [5], можно доказать, что если выполняется $|\xi^s| < c = 2^p$ п. н., $p$ – целое, $\delta > 0$, $q > 0$, $r_0 = 0$, $\gamma^s = c\text{sign}(\xi^s)u_+(|\xi^s| - \alpha^S)$, то последовательность $A_{s+1} = A_s - 2^k\gamma^s$, $s = 0,1, ...$, где $k = \text{ent}(r_s)$ – целая часть числа $r_s$, $r_{s+1} = \min(q, r_s - \gamma^{s+1}(A_{s+1} - A_s) - \delta 2^k)$, сходится к $A$.

Адаптивность параметра $\rho_s$ позволяет быстро достичь значения $A$, а применение СтВУ дает возможность использовать только короткие операции: пересылки, сдвиг, алгебраическое сложение, сравнение чисел, так как

$$\gamma^s \in \{-2p, 0, 2p\},$$

где $p$ – целое.

## Алгоритм

Алгоритм численного метода приведен на рисунке. В блоке 1 задаются начальные значения величин:

- $p$, где $2^p = c > |\xi^s|$ п. н.;
- $y$, где $\delta = 2^y > 0$;
- $[a, b]$ – границы области для $A_s$;
- $q$ – ограничение на степень шагового множителя, обычно $q = 2$;
- $r_0$ – начальное значение степени шагового множителя, обычно $r_0 = 0$;
- $A^0$ – начальная координата;
- $g$ – условие окончания счета;
- $G$ – средняя величина модуля сдвига, начальное значение $G = 10g$;
- $\Delta G$ – приращение модуля сдвига, обычно начальное значение $\Delta G = 0$;
- γ – значение на выходе линейного преобразователя код-вероятность (ЛПКВ), начальное значение $\gamma^0 = 0$;
- $S$ – общее количество итераций;
- $\Delta b = A - b$, $\Delta a = a - A$ – отклонение $A$ от границы области.

В блоке 2 вычисляются значения:

- случайной величины $\alpha^{s+1}$, равномерно распределенной в диапазоне [0, c];
- добавки к степени шага итерации $d$;
- новое значение координаты $A_{s+1}$;
- новое значение модуля сдвига $\Delta G$;
- из содержимого счетчика числа итераций $S$ вычитается 1;
- величина отклонения координат ($\Delta a$ и $\Delta b$) от границы области;
- запоминается текущее значение $r_s$.

В блоках 3, 4, 6, 7 определяется проекция точки на область для $A$.

В блоке 5 проверяется условие окончания вычислений.

В блоке 8 определяются величины:

- $\xi^{s+1} = A_{s+1} - |z_{s+1}|$;
- приращения координат $\Delta x^{s+1}$;
- приращение шага степени $\Delta r$ и новое значение степени шага итерации $r_s$, новое значение среднего модуля сдвига $G$.

В блоке 9 осуществляется линейное преобразование код-вероятность (ЛПКВ) и определяется значение γ.

В блоке 10 вычисляются величины:

- степень шага итерации $r_s$;
- приращение степени шага итерации $\Delta r$;
- новое значение $A_s$;
- новое значение приращения модуля сдвига $\Delta G$.

В блоках 11 и 12 происходит ограничение степени шага итерации и переход на блок 2.

В блоках 13, 14, 15, 16 осуществляется ограничение прироста степени шага итерации с учетом того, что степень $r_s$ не должна меняться за один шаг слишком быстро, введено ограничение $|\Delta r| < 2$. На основании алгоритма строится функциональная схема СтВУ.

## Выводы

Использование стохастического вычислительного устройства позволяет исключить «медленные» операции: возведение в вещественную степень, умножение и деление многоразрядных чисел. Это позволило сократить время выполнения отдельных итераций, что при случайном сигнале на входе устройства уменьшает общее время определения экстремума функции регрессии.

## Литература

1. Поляк Б. Т. Введение в оптимизацию / Б. Т. Поляк. – М. : Наука, 1983. – 384 с.

2. Левин Б. Р. Теоретические основы статистической радиотехники. Кн. 2 / Б. Р. Левин. – М. : Сов. радио, 1968. – 503 с.

3. Невельсон М. Б. Аппроксимация стохастическая и рекуррентное оценивание / М. Б. Невельсон, Р. З. Хасминский. – М. : Наука, 1972. – 304 с.

4. Королюк И. С. Справочник по теории вероятностей и математической статистике / И. С. Королюк, А. В. Скороход. – М. : Наука, 1985. – 640 с.

5. Урясьев С. П. Адаптивные алгоритмы стохастической оптимизации и теории игр / С. П. Урясьев. – М. : Наука, 1990. – 183 с.

6. Федоров Р. Ф. Стохастические преобразователи информации / Р. Ф. Федоров, В. В. Яковлев, Г. В. Добрис. – Л. : Машиностроение, 1978. – 304 с.

# The stochastic processor for definition of an extremum of function of regress

Svistunov S. G.
The Petersburg State University
means of communication of Emperor Alexander I
St. Petersburg, Russia,
ssg47@mail.ru

***Abstract.*** **The summary. In the measuring technique, a radar-location, hydro acoustics often it is required to solve an adaptive problem of identification, i. e. to construct model on available data about object. As examples problems of a statistical estimation of parameters can serve, to regress, the analysis of data etc. In article is offered to be used for the decision of the specified problem by the adaptive quasigradient algorithm realized on a stochastic computer. The offered approach gives the chance to reduce time of the decision and to simplify used hardware. In the article the detailed algorithm of a specialized computer is resulted.**

***Use of a stochastic computer allows to exclude "slow" operations: erection in material degree, multiplication and division of multidigit numbers.*** **It has allowed to reduce time of performance of separate iterations, that at a casual signal on a device input reduces general time of definition of an extremum of function of regress.**

***Keywords:*** **an extremum of function of regress, adaptive algorithms of stochastic optimization, stochastic computers, the converter a code-probability, CCP, StC.**

## References

1. Poliak B. T. *Vvedeniye v optimizatsiyu* [Introduction in optimization]. Moscow, Nauka, 1983, 384 p.

2. Levin B. R. *Teoreticheskiye osnjvy statisticheskoy radiotekhniki* [Theoretical of a basis of a statistical radio engineering], Book 2. Moscow, Sovetskoye Radio, 1968, 503 p.

3. Nevelson M. B., Hasminsky R. Z. *Approksimatsiya stokhasticheskaya i rekurrentnoye otsenivaniye* [Stochastic approximation and recurrent evaluation]. Moscow, Nauka, 1972, 304 p.

4. Koroljuk I. S., Skorokhod A. V. *Spravochnik po teorii veroyatnostey i matematicheskoy statistike* [Fastwalker director on probability theory and the mathematical statistics]. Moscow, Nauka, 1985, 640 p.

5. Uryas'ev S. P. *Adaptivnyye algoritmy stokhasticheskoy optimizatsii i teorii igr* [Adaptive algorithms of stochastic optimization and the theory of games]. Moscow, Nauka, 1990, 183 p.

6. Fedorov R. F., Jakovlev V. V., Dobris G. V. *Stokhasticheskiye preobrazovateli informatsii* [Stochastic information converters]. Leningrad, Mashinostroyeniye, 1978, 304 p.

# Организация автоматизированного контроля качества в жизненном цикле программных средств критически важных систем

Марковский А. С., Самонов А. В., Бурова И. О.
Военно-космическая академия им. А. Ф. Можайского
Санкт-Петербург, Россия
lexx26@list.ru, a.samonov@mail.ru, burova@mamaev.spb.ru

***Аннотация.*** **В статье представлены результаты анализа состояния и перспектив в области разработки программных средств (ПС) критически важных систем. Установлено, что основными направлениями решения накопившихся в данной области проблем являются совершенствование нормативно-методической базы, внедрение современных технологий разработки и реализация сквозного контроля качества ПС на всех этапах их жизненного цикла (ЖЦ). Предложена соответствующая современным технологиям разработки программного обеспечения усовершенствованная модель ЖЦ ПС, в которой точно определены место и роль процессов контроля и обеспечения качества. Описана классификация показателей качества ПС. Представлен метод и технологии реализации автоматизированного сквозного контроля качества на всех этапах жизненного цикла ПС.**

***Ключевые слова:*** **верификация, жизненный цикл, контроль качества, критически важные системы, модель жизненного цикла, показатели качества программных средств, программные средства.**

## Введение

На сегодня активное и повсеместное внедрение автоматизированных систем, важнейшим элементом которых являются программные средства (ПС), в том числе в критически важных системах (КВС), обусловило высокий уровень требований к надежности, производительности, защищенности и другим характеристикам ПС. В то же время, как показывает практика в нашей стране и за рубежом, чем сложнее и объемнее программное обеспечение (ПО), тем больше в нем дефектов.

В качестве подтверждения данного тезиса можно привести результаты исследований исходных кодов открытого и проприетарного ПО, выполненных компанией Coverity [1]. В ходе исследований была проведена проверка на наличие ошибок и уязвимостей более 300 миллионов строк кода 41 проприетарного программного продукта и 37 миллионов строк кода 45 программных продуктов с открытым исходным кодом. Установлено, что в среднем на 1000 строк кода приходится 1 ошибка. С учетом того, что средние размеры проектов с открытым исходным кодом 800 тысяч, а проприетарного ПС – 7,5 миллионов строк кода, получается, что в реальных проектах содержатся сотни ошибок, на выявление и устранение которых расходуются значительные интеллектуальные, временны́е и финансовые ресурсы.

Известные объективные и субъективные трудности создания высококачественных ПС в нашей стране усугублены следующими факторами (обстоятельствами):

• современные ПС разрабатываются на основе устаревших неэффективных технологий и инструментальных средств;

• отечественная нормативно-методическая база в области разработки ПС не соответствует современному уровню информационно-коммуникационных технологий и средств разработки;

• на предприятиях, разрабатывающих ПО для КВС, не обеспечивается требуемый уровень контроля качества ПС.

В связи с изложенным становится очевидной необходимость совершенствовать методологическое, нормативно-организационное и технологическое обеспечение процессов разработки, эксплуатации и сопровождения ПС, используемых в КВС.

## Базовые модели ЖЦ ПС

Вопросы развития и совершенствования технологий и средств проектирования, разработки, эксплуатации и сопровождения ПО исследуются в рамках системной программной инженерии (СиПИ). Одним из основных понятий в СиПИ является понятие жизненного цикла (ЖЦ) ПС. В общем случае ЖЦ определяется моделью и описывается в форме методологии или метода. Модель ЖЦ определяет концептуальный взгляд на организацию ЖЦ, его основные фазы, условия и порядок перехода между ними. Методология ЖЦ задает комплекс работ, их детальное содержание и ролевую ответственность специалистов на всех этапах выбранной модели ЖЦ, обычно определяет и саму модель, а также предлагает практические рекомендации, позволяющие максимально эффективно воспользоваться соответствующей методологией.

Таким образом, ЖЦ ПС – это непрерывный процесс, описывающий все, что происходит с ПС с момента принятия решения о необходимости создания ПС до его изъятия из эксплуатации. Модель ЖЦ ПС – это определенным образом упорядоченная совокупность этапов ЖЦ (видов деятельности и событий), условия и порядок переходов между ними, обеспечивающих достижение цели проекта в установленные сроки в рамках доступного бюджета времени, людских и финансовых средств [2, 3].

Основными понятиями, в терминах которых описывается ЖЦ ПС, являются: виды деятельности, роли (исполнители) и артефакты (результаты).

Виды деятельности – это набор действий, направленных на решение одной задачи или группы тесно связанных задач в рамках разработки и сопровождения ПС. Например, анализ предметной области, описание требований, проектирование, разработка кода, тестирование, управление конфигурациями, развертывание и т. д.

Исполнители – это профессиональная специализация людей, участвующих в создании или сопровождении ПС и имеющих одинаковые интересы или решающих одни и те же задачи по отношению к этому ПС.

Артефакты – различные информационные сущности, документы, модели, программы, создаваемые или используемые в ходе разработки и сопровождения ПС. Например, техническое задание, описание архитектуры, модель предметной области, исходный код, результаты испытаний и т. д.

В настоящее время создание ПС реализуется в рамках одной из трех базовых моделей ЖЦ: каскадной, инкрементной или спиральной, результаты анализа которых приведены в табл. 1. Выбор той или иной стратегии определяется характеристиками проекта, требованиями к конечному продукту, командой разработчиков и конечными пользователями.

Таким образом, использование каскадной стратегии наиболее эффективно в следующих случаях:

1) при разработке проектов с четкими, неизменяемыми в течение ЖЦ требованиями и с понятной реализацией;

2) при разработке проектов невысокой сложности;

3) при выполнении больших проектов в качестве составной части моделей ЖЦ, реализующих другие стратегии разработки.

Использование инкрементной модели наиболее эффективно в следующих случаях:

1) при разработке проектов, в которых большинство требований можно сформулировать заранее, но часть из них можно уточнить через определенный период времени;

2) при разработке сложных проектов с заранее сформулированными требованиями;

3) при необходимости максимально быстро поставить потребителю продукт, имеющий определенные базовые функциональные свойства;

4) при разработке проектов с низкой или средней степенью рисков;

5) при выполнении проекта с применением новых технологий.

Использование эволюционной стратегии наиболее эффективно в следующих случаях:

1) при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или требуют уточнения;

2) при разработке сложных проектов, в том числе тех, которые используют новые технологии.

Анализ нормативно-методических документов, регламентирующих разработку ПС КВС в части состава, структуры и содержания этапов создания и эксплуатации ПС [4–9], позволил определить следующие недостатки и проблемные вопросы:

1) состав, структура и содержание этапов создания ПС ориентированы на реализацию каскадной модели ЖЦ ПС,

Таблица 1

Достоинства и недостатки моделей ЖЦ ПС

| Достоинства | Недостатки |
|---|---|
| Каскадная | |
| 1) Стабильность требований в процессе разработки;<br>2) необходимость пройти только один этап разработки, что обеспечивает простоту применения стратегии;<br>3) простота планирования, контроля и управления проектом;<br>4) доступность для понимания заказчиками | 1) Сложность полного формулирования требований в начале процесса разработки и невозможность их динамического изменения на протяжении ЖЦ;<br>2) линейность процесса разработки, разрабатываемые ПС или система обычно слишком велики и сложны; из-за возврата к предыдущим шагам для решения возникающих проблем увеличиваются финансовые затраты и нарушается график работ;<br>3) непригодность промежуточных продуктов для использования;<br>4) недостаточное участие пользователя в процессе разработки ПС – только при разработке требований и во время приемочных испытаний – не позволяет пользователю предварительно оценить качество ПС или системы |
| Инкрементная | |
| 1) Возможность получить функциональный продукт после реализации каждого инкремента;<br>2) быстрое создание инкремента; это сокращает сроки начальной поставки, позволяет снизить затраты на первоначальную и последующие поставки программного продукта;<br>3) предотвращение реализации громоздких спецификаций требований; стабильность требований во время создания определенного инкремента; возможность учитывать изменившиеся требования;<br>4) снижение рисков по сравнению с каскадной стратегией;<br>5) включение в процесс пользователей, что позволяет оценить функциональные возможности продукта на более ранних этапах разработки и в итоге приводит к повышению качества программного продукта, снижению затрат и времени на его разработку | 1) Необходимость полного функционального определения ПС или системы в начале жизненного цикла для обеспечения планирования инкрементов и управления проектом;<br>2) возможность текущего изменения требований к ПС или к системе, которые уже реализованы в предыдущих инкрементах;<br>3) сложность планирования и распределения работ;<br>4) проявление человеческого фактора, связанного с тенденцией к оттягиванию решения трудных проблем на поздние инкременты, может нарушить график работ или снизить качество программного продукта |
| Спиральная | |
| 1) Возможность уточнения и внесения новых требований в процессе разработки;<br>2) пригодность промежуточного продукта для использования;<br>3) возможность управления рисками;<br>4) обеспечение широкого участия пользователя в проекте начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества;<br>5) реализация преимуществ каскадной и инкрементной стратегий | 1) Неизвестность точного количества необходимых итераций и сложность определения критериев для продолжения разработки на следующей итерации; это может вызвать задержку реализации конечной версии системы или ПС;<br>2) сложность планирования и управления проектом;<br>3) необходимость активного участия пользователей в проекте, что не всегда осуществимо;<br>4) необходимость в мощных инструментальных средствах и методах прототипирования;<br>5) возможность отодвинуть решение трудных задач на последующие циклы может привести к несоответствию полученных продуктов требованиям заказчиков |

в то время как сейчас стандартом де-факто при разработке сложных ПС является спиральная модель ЖЦ;

2) противоречия в используемой терминологии и в видах деятельности, выполняемых на одних и тех же этапах ЖЦ ПС;

3) избыточность и излишняя детализация отдельных видов деятельности и артефактов;

4) не используются преимущества применения концепции процессного подхода к разработке и применению ПС;

5) недостаточно полно и конкретно определены место и роль процессов управления качеством ПС.

В табл. 2 представлена модель ЖЦ ПС КВС, в которой устранены названные недостатки. Структура данной модели описывается с помощью трех основных понятий: вид деятельности, исполнители и артефакты.

Таблица 2

Усовершенствованная модель ЖЦ ПС КВС

| № | Этапы, виды деятельности (процессы) | Исполнители | Артефакты (содержание и форма) |
|---|---|---|---|
| 1 | Анализ потребности, разработка концепции и обоснование требований | | |
| 1.1 | Анализ потребности и обоснование необходимости ПС | Аналитики<br>Заказчики<br>Потребители<br>Организации научного и технического сопровождения (НС, ТС) | Обоснование места и роли ПС в составе КВС. Аналитический отчет |
| 1.2 | Разработка концепции построения, функционирования и применения ПС | Аналитики<br>Заказчики<br>Потребители<br>Организации НС, ТС | Концепция применения ПС в составе КВС |
| 1.3 | Разработка технического задания (ТЗ) на ПС | | |
| 1.3.1 | Разработка проекта ТЗ на ПС | Аналитики<br>Заказчики<br>Потребители | Проект ТЗ на ПС (функциональные и эксплуатационные требования) |
| *1.3.2* | *Верификация требований к ПС* | *Заказчик<br>Организации НС, ТС* | *Заключение о полноте, непротиворечивости и корректности требований к ПС* |
| 1.3.3 | Согласование и утверждение ТЗ | Заказчик<br>Потребители<br>Организации НС, ТС | Утвержденное ТЗ |
| 2 | Разработка ПС | | |
| 2.1 | Планирование разработки ПС | Руководитель проекта<br>Проектировщик | Сквозной график разработки ПС |
| 2.2 | Проектирование ПС | | |
| 2.2.1 | Разработка проекта (архитектуры) ПС | Проектировщик (конструктор) | Проект архитектуры и компонентов ПС |
| *2.2.2* | *Верификация проекта (архитектуры) ПС* | *Организации НС, ТС* | *Заключение о соответствии проекта заданным в ТЗ условиям* |
| 2.3 | Разработка программ и программной документации | | |

Продолжение табл. 2

| № | Этапы, виды деятельности (процессы) | Исполнители | Артефакты (содержание и форма) |
|---|---|---|---|
| 2.3.1 | Разработка программ в соответствии с проектом ПС | Руководитель проекта<br>Разработчики<br>Технические писатели | Программный код и программная документация |
| *2.3.2* | *Тестирование на контрольных примерах (модульное, функциональное комплексное, нагрузочное)* | *Тестировщики* | *Протоколы и акты испытаний. Предложения по доработке* |
| 2.3.3 | Возврат к этапам 2.2.1 или 2.3.1 на доработку или завершение разработки, включая подготовку рабочей конструкторской, программной и эксплуатационной документации (РКД, ПД и ЭД) | Руководитель проекта<br>Проектировщик<br>Разработчики<br>Технические писатели | ПС, РКД, ПД, ЭД, задание на доработку ПС |
| *2.3.4* | *Верификация РКД, ПД и ЭД* | *Отдел Технического Контроля (ОТК) предприятия Организации НС, ТС* | *Акты соответствия РКД, ПД и ЭД заданным в ТЗ условиям* |
| 2.4 | Проведение испытаний ПС | | |
| *2.4.1* | *Предварительные, приемочные, государственные испытания ПС* | *Разработчики<br>Организации НС, ТС<br>Потребители<br>Заказчики* | *Протоколы и акты испытаний* |
| *2.4.2* | *Верификация результатов испытаний* | *Разработчики<br>Организации НС, ТС* | *Экспертное заключение* |
| 3 | Производство | | |
| 3.1 | Постановка ПС на производство | Разработчики предприятия-изготовителя | Копии РКД, ПД и ЭД<br>Технологическая линия производства ПС |
| 3.2 | Изготовление ПС | Разработчики предприятия-изготовителя | ПС с комплектом документации |
| *3.3* | *Контроль и приемка ПС* | *ОТК<br>Потребитель* | *Протоколы и акты приемки ПС* |
| 3.4 | Поставка ПС потребителю | Разработчики предприятия-изготовителя | Акт о передаче ПС потребителю |
| 4 | Применение в составе системы | | |
| 4.1 | Эксплуатация | | |
| 4.1.1 | Опытная (экспериментальная) эксплуатация | Потребитель<br>Разработчики | Отчет с результатами опытной эксплуатации |
| 4.1.2 | Функционирование ПС в составе системы | Потребитель | Результаты применения по назначению (расчеты, информационные документы и т. д.) |
| 4.2 | Сопровождение | | |
| 4.2.1 | Организация сопровождения ПС | Разработчики предприятия-изготовителя | Договоры и акты о сопровождении ПС |

Окончание табл. 2

| № | Этапы, виды деятельности (процессы) | Исполнители | Артефакты (содержание и форма) |
|---|---|---|---|
| *4.2.2* | *Анализ функционирования ПС* | *Потребитель Разработчики предприятия-изготовителя* | *Замечания и рекламации. Предложения о доработке и модернизации* |
| 4.2.3 | Модернизация ПС | Разработчики | Модернизированное ПС |
| *4.2.4* | *Тестирование и верификация модернизированного ПС* | *Разработчики ОТК Потребитель* | *Протоколы и акты испытаний* |
| 4.3 | Прекращение эксплуатации | | |
| 4.3.1 | Подготовка к снятию ПС с эксплуатации | Потребитель Разработчики | Обоснование целесообразности снятия ПС с эксплуатации |
| 4.3.2 | Прекращение эксплуатации ПС | Потребитель Заказчик | Документ о снятии ПС с эксплуатации |

Важной особенностью представленной в табл. 2 модели ЖЦ является наличие точного указания места и роли процессов контроля качества ПС КВС, которые в таблице выделены курсивом. Результатом выполнения данных процессов является оценка степени соответствия артефактов, получаемых на каждом этапе, заданным или ожидаемым требованиям к создаваемому ПС.

## Контроль качества ПС КВС

Качество ПО – это совокупность существенных свойств (характеристик) программного обеспечения, обусловливающих его пригодность для использования по назначению. Как видно из табл. 2, процессы верификации должны осуществляться по отношению ко всем значимым артефактам. Первым из них является ТЗ, в котором определяются требования к функциональным и эксплуатационным характеристикам ПС. Наиболее полная классификация характеристик ПС представлена в стандартах серии ISO/IEC 25000 [10]. В соответствии с этой классификацией все характеристики ПО сведены в 8 групп (рис. 1).

Каждая группа характеристик состоит из подхарактеристик, или атрибутов. Например, такая характеристика ПС, как «надежность», состоит из четырех подхарактеристик: отказоустойчивости, восстанавливаемости, завершенности и доступности [10]. Для оценивания степени, с которой определенная характеристика соответствует установленным требованиям, используются показатели качества. Показатель качества – это переменная или несколько переменных, значение которых характеризует меру качества ПО относительно одного или нескольких существенных свойств. Так, в качестве меры отказоустойчивости системы могут использоваться следующие показатели:

- вероятность безотказной работы $P(t)$;
- средняя наработка на отказ $T$о;
- гамма-процентная наработка до отказа $T\gamma$;
- интенсивность отказов $\lambda(t)$;
- параметр потока отказов $\omega(t)$;
- средняя доля безотказной наработки $I(t)$;
- плотность распределения времени безотказной работы $f(t)$.

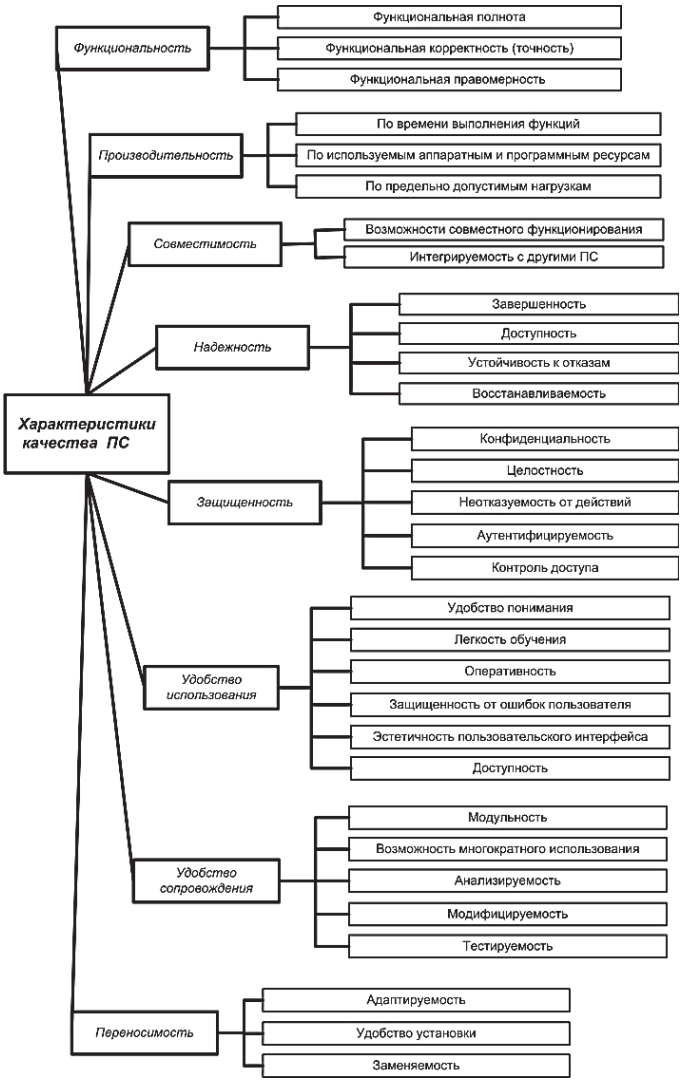



Рис. 1. Классификация характеристик качества ПС

Показатели качества можно спроецировать на основные группы их потребителей: разработчиков, заказчиков и конечных пользователей. В соответствии с этим принципом все показатели распределяются по трем группам [2, 4, 10]:

*Показатель внутреннего качества* – степень, с которой множество статических свойств ПС удовлетворяет заявленным или подразумеваемым требованиям к ПС при использовании в заданных условиях. Статические свойства имеют отношение к архитектуре, внутренней организации и корректности (безошибочности) кода ПС. Примерами показателей внутреннего качества являются количество ошибок спецификации, проектирования и кодирования, избыточность кода, отсутствие контроля вводимых данных и др. Эти показатели используют программисты и тестировщики ПС в ходе кодирования, отладки, тестирования.

*Показатель внешнего качества* – степень, с которой ПС позволяет функционированию системы удовлетворять предъявленным к ней требованиям при использовании в заданных условиях. Примерами таких показателей являются вероятность безотказной работы в течение определенного периода времени, время решения расчетных задач, пропускная способность каналов передачи данных, время восстановления работоспособности после сбоев и другие. Используется во

время различных испытаний ПС представителями заказчика, органов сертификации и конечными пользователями.

*Показатель качества при использовании* – степень, с которой ПС пригодно к использованию по назначению определенными пользователями в заданных условиях применения. Используются, прежде всего, конечными пользователями на этапах эксплуатации ПС.

В табл. 3 представлены методы верификации артефактов, получаемых в ходе разработки ПС КВС.

Таблица 3

Артефакты ЖЦ ПС КВС и методы их верификации

| Артефакт | Метод верификации | Исполнитель |
|---|---|---|
| ТЗ на разработку ПС | Экспертиза (общая и техническая) | Заказчики<br>Потребители<br>Организации НС, ТС |
| Архитектура ПС (проект) | Экспертиза техническая | Аналитики<br>Проектировщики<br>Организации НС, ТС |
| Программный код ПС | 1) Статический анализ исходного кода;<br>2) динамический анализ выполнения программы;<br>3) регрессионное тестирование;<br>4) тестирование на стенде | Программисты<br>Тестировщики |
| РКД, ПД и ЭД ПС | Экспертиза (общая и техническая) | ОТК<br>ПЗ<br>Организации НС, ТС |
| Результаты испытаний | 1) Экспертиза (общая и техническая);<br>2) инспекция | Заказчики<br>ПЗ<br>Организации НС, ТС |
| Программный продукт (на этапе производства и приемки) | 1) Экспертиза (общая и техническая);<br>2) тестирование на стенде | Разработчики<br>ОТК<br>ПЗ<br>Организации НС, ТС |

Как показал анализ, основными методами верификации являются экспертиза требований, проектных решений и программного кода, статические и динамические методы тестирования ПС, аудит, регистрация и анализ результатов эксплуатации.

Наряду с совершенствованием нормативно-методической базы важным направлением совершенствования процессов управления качеством ПС является использование для верификации и валидации артефактов ЖЦ ПС КВС формальных процедур и поддерживающих их инструментальных средств. Для создания таких средств необходимо разработать формальную модель и язык описания всех артефактов. Схема алгоритма верификации представлена на рис. 2.

В настоящее время наиболее подходящими для этого средствами являются унифицированный язык моделирования UML (UnifiedModellingLanguage) [11], язык описания метамоделей MOF (MetaObjectFacility) [12] и протокол обмена метаданными XMI (XML MetadataInterchange) [13].

В результате совместного использования этих средств можно построить абстрактную метамодель ЖЦ ПС КВС, разработать средства управления и обмена данными (моделями) с целью их трансформации в поддерживаемые технологии программирования и реализации автоматизированной

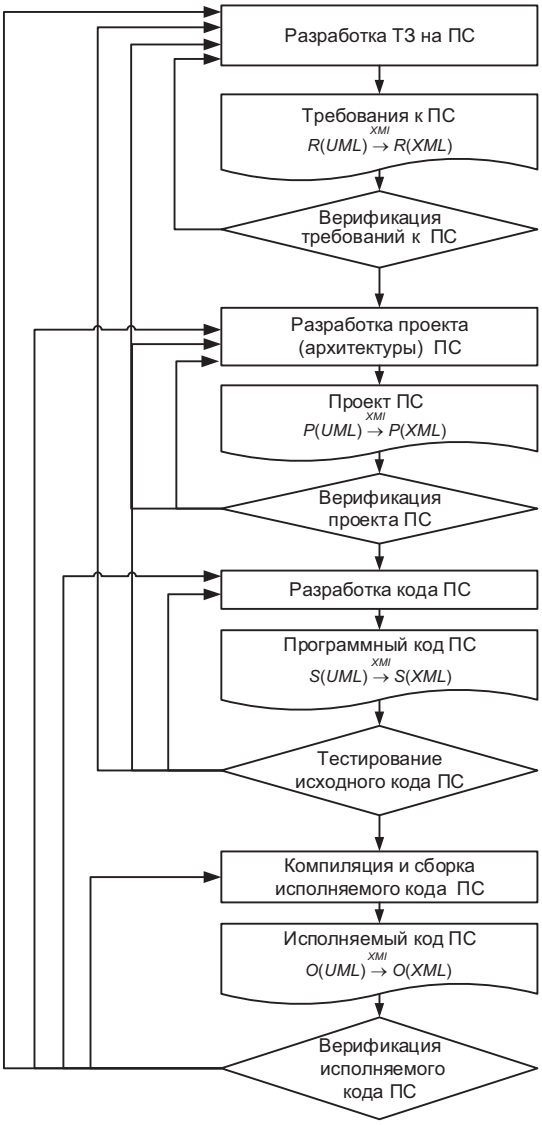



Рис. 2. Алгоритм контроля качества ПС КВС

верификации. Ярким примером таких средств является линейка продуктов MATLAB: Simulink, SystemTest, Simulink Design Verifier, Simulink Verification and Validation [14].

## Заключение

Многочисленные исследования в области оценивания стоимости разработки ПО, в том числе стоимости обнаружения и исправления ошибок, показали, что удельная стоимость исправления дефектов возрастает по экспоненциальному закону распределения по мере продвижения продукта к стадии эксплуатации [15–19]. Представленный на рис. 3 график показывает [20, 21], что затраты на исправление дефекта на этапе тестирования в четыре раза превышают затраты на его устранение на этапе проектирования и в 20 раз, если бы он был обнаружен и устранен на этапе обоснования требований. Стоимость исправления дефектов на этапе эксплуатации превышает затраты на этапах проектирования и тестирования в десятки раз.

В связи с этим становится очевидным, что применение данного подхода на практике позволит не только повысить


Рис. 3. Алгоритм контроля качества ПС КВС

качество разрабатываемого ПО для КВС, но и существенно снизить затраты на поиск и устранение дефектов в нем.

## Литература

1. http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf (дата обращения 05.07.2015).

2. Кулямин В. В. Методы верификации программного обеспечения / В. В. Кулямин. – М. : Ин-т системного программирования РАН, 2008. – 111 с.

3. Генельт А. Е. Управление качеством разработки программного обеспечения : учеб.-методич. пособие / А. Е. Генельт. – СПб. : ИТМО, 2007. – 187 с.

4. ISO/IEC 9126 Software engineering – Product quality. – Part 1–41, 2001.

5. ГОСТ Р 51189-98. Порядок разработки программных средств систем вооружения (введ. 01.07.1999). – М. : Стандартинформ, 2010. – 16 с.

6. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания (введ. 01.01.1992). – М. : Стандартинформ, 2009. – 6 с.

7. ГОСТ Р ИСО/МЭК 12207-2010. Процессы жизненного цикла программных средств (введ. 01.03.2012). – М. : Стандартинформ, 2011. – 105 с.

8. ГОСТ 28195-89. Оценка качества программных средств. Общие положения (введ. 01.07.1990). – М. : Стандартинформ, 2001. – 31 с.

9. ГОСТ Р ИСО/МЭК 15288-2005. Системная и программная инженерия. Процессы жизненного цикла систем (введ. 01.01.2007). – М. : Стандартинформ, 2006. – 57 с.

10. http://iso25000.com (дата обращения 27.05.2015).

11. http://www.uml.org (дата обращения 15.06.2015).

12. http://www.omg.prg/mof (дата обращения 17.11.2015).

13. http://www.omg.prg/spec/XMI (дата обращения 07.10.2015).

14. http://matlab.ru/products/simulink (дата обращения 20.09.2015).

15. Fairley R. E. Managing and Leading Software Projects. – Wiley-IEEE Comput. Soc. Press, 2009. – 512 p.

16. Naveda J. F., Seidman S. B. A Self-Study Guide for Today's Software Professional // IEEE Comput. Soc. Real-World Softw. Eng. Probl. – Wiley-IEEE Comput. Soc. Press, 2006. – 328 p.

17. http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html (дата обращения 20.10.2015).

18. http://www.softserveinc.com/en-us/services/software-testing (дата обращения 20.10.2015).

19. http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools (дата обращения 20.10.2015).

20. Boehm B., Basili V. Software Defect Reduction Top 10 List // IEEE Comput., IEEE Comput. Soc. – 2001. – Vol. 34, No.1. – P. 135–137.

21. Selby R. W. Software Engineering: Barry W. // Boehm's Lifetime Contrib. Software Dev., Manage. Res. – Wiley-IEEE Computer Society Press, 2007. – 832 p.

# Organizatsiya avtomatizirovannogo kontrolya kachestva v zhiznennom tsikle programmnyh sredstv kriticheski vazhnyh sistem

Markovsky A. S., Samonov A. V., Burova I. O.
lexx26@list.ru, a.samonov@mail.ru, burova@mamaev.spb.ru
Military Space Forces Academy
Sankt-Petersburg, Russian Federation

***Abstract.*** **The article presents the results of the analysis of the state and perspectives in the development of software of automated systems of critical infrastructure. It is established that the main directions in solving the field problems are the improvement of normative-methodical base, introduction of modern technologies for the development and implementation of end-to-end control quality software at all stages of their life cycle (LC). It is offered the improved model of LC software complying with the modern technology of software development life cycle, in which precisely defined the place and role processes of quality control. We describe the classification of quality characteristics software. The method and implementation technology for automated end-to-end quality control at all stages of the life cycle software are represented.**

***Keywords:*** **quality control, quality software tools, software tools of weapons system, software development life cycle, model of software development life cycle, verification**

## References

1. http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf (accessed 5 July 2015).

2. Kulyamin V. V. *Metodi verifikatcii programmnogo obespecheniya* [Methods of verification of the software], Moscow, Institut sistemogo programmirovaniya RAN, 2008, 111 p.

3. Genelt A. E. *Upravlenie kachestvom razrabotki programmnogo obespecheniya: uchebno-metodicheskoe posobiye* [Quality management of development of the software], St. Petersburg, ITMO, 2007, 187 p.

4. ISO/IEC 9126 Software engineering – Product quality. Part 1-41, 2001.

5. GOST R 51189-98. *Poryadok razrabotki programmnyh sredstv system vooruzheniya* [Order of development of software of systems of arms], 1999-07-01, Moscow, Standartinform, 2010, 16 p.

6. GOST 34.601-90. *Avtomatizirovannye sistemy. Stadiisozdaniya* [The automated systems. Development stages], 1992-01-01, Moscow, Standartinform, 2009, 6 p.

7. GOST RISO/MEK 12207-2010. *Protsessy zhiznennogo tsikla programmnyh sredstv* [Processes of life cycle of software], 2012–03–01, Moscow, Standartinform, 2011, 105 p.

8. GOST 28195-89. *Otsenka kachestva programmnyh sredstv. Obshchie polozhenya* [Assessment of quality of software. General provisions], 1990-07-01, Moscow, Standartinform, 2001, 31 p.

9. GOST R ISO/MEK 15288-2005. *Systemnaya i programmnaya inzheneriya. Protsessy zhiznennogo tsikla sistem* [Systems and software engineering. System life cycle processes], 2007–01–01, Moscow, Standartinform, 2006, 57 p.

10. http://iso25000.com (accessed 27 May 2015).

11. http://www.uml.org (accessed 15 June 2015).

12. http://www.omg.prg/mof (accessed 17 Nov. 2015).

13. http://www.omg.prg/spec/XMI (accessed 07 Oct. 2015).

14. http://matlab.ru/products/simulink (accessed 20 Sept. 2015).

15. Fairley R. E. Managing and Leading Software Projects. Wiley-IEEE Comput. Soc. Press, 2009, 512 p.

16. Naveda J. F., Seidman S. B. A Self-Study Guide for Today's Software Professional, *IEEE Comput. Soc. Real-World Softw. Eng. Probl.* Wiley-IEEE Comput. Soc. Press, 2006, 328 p

17. .http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html (accessed 20 Oct. 2015).

18. http://www.softserveinc.com/en-us/services/software-testing (accessed 20 Oct. 2015).

19. http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools (accessed 20 Oct. 2015).

20. Boehm B., Basili V. Software Defect Reduction Top 10 List, *IEEE Comput., IEEE Comput. Soc.*, 2001, Vol. 34, no. 1, pp. 135-137.

21. Selby R. W. Software Engineering: Barry W. Boehm's Lifetime Contrib. Software Dev., Manage. Res., Wiley-IEEE Computer Society Press, 2007. 832 p.

# Методика оценивания защищенности информационно-телекоммуникационных узлов

Шинкаренко А. Ф.
Военно-космическая академия имени А. Ф. Можайского
Санкт-Петербург, Российская Федерация
tonio87@rambler.ru

***Аннотация.*** **В работе рассматриваются перспективные направления исследований в области анализа защищенности информационно-телекоммуникационных сетей и вводятся основные показатели для расчета их защищенности. Предлагается многоуровневый подход к оцениванию защищенности, основанный на деревьях атак и зависимостях сервисов.**

***Ключевые слова:*** **показатели защищенности, дерево атак, оценка рисков, уязвимость.**

## Введение

В условиях ограниченных ресурсно-временных возможностей по обеспечению информационной безопасности проблема повышения уровня защищенности информационно-телекоммуникационных сетей (ИТКС) [1] и узлов в отдельности, с одной стороны, является нетривиальной, а с другой – значительно влияет на эффективность мероприятий, направленных на поддержание информационной безопасности. Для каждого информационно-технического объекта на основе анализа его свойств требуется принять решение о применении тех или иных способов и средств защиты от информационно-технических воздействий.

Сети строятся с использованием коммутаторов, маршрутизаторов и других устройств, которые стали чрезвычайно сложными, поскольку они реализуют все большее число сложных распределенных протоколов, стандартизированных международной организацией технических стандартов (на сегодня число активно используемых протоколов и их версий превысило 600), вместе с этим возрастает и число уязвимостей ИТКС, которыми могут воспользоваться злоумышленники при планировании и проведении составного деструктивного программно-аппаратного воздействия. Управление и обеспечение безопасности сложной сетевой инфраструктуры – сегодня в большей степени искусство, чем инженерия. Рост сетевых атак, вирусов и других сетевых угроз свидетельствует о том, что вопросы безопасности до сих пор не имеют надежных решений. Современное международное сообщество осознало, что компьютерные и телекоммуникационные сети являются объектом национальной безопасности.

Применение традиционных средств повышения защищенности ИТКС, основанных на анализе журналов событий безопасности, в силу описанных выше обстоятельств и повышенных требований к безопасности информации является недостаточным [2, 3]. Одно из актуальных направлений решения этой проблемы в настоящее время – совершенствование сервисов защиты информации, в первую очередь, тех служб, которые оценивают состояние ИТКС, управляют защитой и адаптируют политику безопасностии компонентов системы защиты информации.

Имеющиеся программные средства анализа защищенности ИТКС можно условно разделить на следующие классы:

- сбора сведений об ИТКС и узлах: Nmap, Wireshark и др.;
- обнаружения уязвимостей ПО: Nessus (OpenVas), Acunetix, AppScanи др.;
- эксплуатации уязвимостей: ImmunityCanvas, MetasploitFramework, VulnDisco, SAINTexploitи др.

## Релевантные исследования

Известно много работ в предметной области моделирования компьютерных вторжений и обоснования показателей защищенности.

S. Kumar, E. H. Spafford предложили модель компьютерных атак на основе раскрашенных сетей Петри [4]. Каждая сигнатура атаки выражается как шаблон, который показывает взаимосвязь между событиями и их содержанием. Обозначения начального и конечного состояний и связь между ними определяют шаблон событий.

K. Iglun, R. A. Kemmerer, P. A. Porras описали подход к анализу перехода состояний при вторжении для моделирования компьютерных атак [5]. В их статье компьютерная атака представляется как последовательность действий, выполняемых атакующим для компрометации безопасности компьютерной системы. Атаки описываются с помощью диаграмм перехода состояний.

В работе F. Cohen «Моделирование компьютерных атак, защита и последовательность» рассмотрен подход к оцениванию сетевой безопасности как «причинно-следственная модель атаки и защиты информационной системы» [6]. Она состоит из сети, которая отображается узлами и их связями, причинно-следственной связной описательной модели и псевдослучайного генератора чисел. Стоит указать на значительное упрощение подобного представления при моделировании компьютерных вторжений, основанном на причинно-следственной связи.

В [7] представлены наглядные модели сети, возможностей, целей и способов действий атакующего. Эти модели используются для определения устройств, которые будут скомпрометированы с наибольшей вероятностью. Для предсказания поведения атакующего используются экономические принципы, использующие компромисс «выгода – затраты».

В [8–11] атаки описываются и исследуются в структурированной, основанной на графовых деревьях форме.

В [12] представлена высокоуровневая концептуальная модель атак, основанная на намерениях нарушителя (стратегии атаки). Широкомасштабное распределенное обнаружение вторжения сравнивается с задачей военного управления. Статья определяет намерения вторжения как дерево целей. Конечная цель вторжения соответствует корневому узлу. Узлы нижних уровней отображают альтернативные или упорядоченные подцели в достижении верхнего узла/цели. Конечные узлы (листья) являются подцелями. Они могут подкрепляться событиями, сгенерированными в различных средах.

На основе исследований в области показателей защищенности [13–15] можно выделить основные группы показателей: топологические характеристики, показатели нарушителя, характеристики атаки и реакции на атаку, интегральные показатели, стоимостные характеристики и показатели, применяемые при анализе уязвимостей «нулевого дня» [16].

## Методика оценивания защищенности ИТКС на основе деревьев атак

В общем виде методику оценивания защищенности ИТКС можно представить в виде трех этапов: подготовительного, эксплуатации и заключительного. На подготовительном этапе для каждого узла ИТКС формируется список возможных атакующих действий, разбитых на группы по следующим параметрам: класс атаки, необходимый тип доступа и необходимый уровень знаний нарушителя, а для каждой группы, в свою очередь, формируется список конкретных атак и уязвимостей, которые эти атаки используют. На этапе эксплуатации определяется качественный уровень риска для всех угроз, также строятся деревья атак, на основе которых происходит дальнейшее оценивание защищенности ИТКС. Уровень защищенности анализируемой ИТКС на основе деревьев атак определяется на заключительном этапе.

Данная методика объединяет качественный и количественный подходы к оцениванию защищенности и позволяет определить общий уровень защищенности ИТКС. В данном подходе предлагается использовать общую систему оценивания уязвимостей CVSS (*Common Vulnerability Scoring System*) для определения критичности атакующих действий и методику FRAP (*Facilitated Risk Analysis Process*) [17].

Сбор исходной информации о компьютерной сети и формирование модели ИТКС, а также составление общего списка уязвимостей реализуются на основе описания программно-аппаратного обеспечения хоста на языке CPE и таких открытых баз уязвимостей, как NVD (*National Vulnerability Database*). Источниками данных об открытых уязвимостях также могут служить отчеты сканеров безопасности, таких как Nessus, MaxPatrol, Nmap и других. Уязвимости в системе хранятся в формате CVE [18]. По полученной информации формируются множества уязвимостей и выбираются модели нарушителей на основе знаний эксперта по безопасности. Также источниками данных об угрозах информационной безопасности и уязвимостях программного обеспечения могут служить соответствующие банк угроз и банк уязвимостей, разработанный ФСТЭК Российской Федерации [19].

Следующим этапом методики является подготовка данных для формирования деревьев атак и выделения возможных атакующих действий, доступных нарушителю для каждого узла ИТКС. Кроме отдельных уязвимостей при построении дерева атак используются шаблоны атак в формате CAPEC [20], которые могут выступать не только в качестве входной информации для построения графов атак, но и как результат анализа безопасности: они могут описывать наиболее часто встречающиеся последовательности эксплуатации уязвимостей и других действий атакующего.

Также шаблоны содержат описания атак, которые не используют уязвимости: например, первая стадия проведения анализа – это сбор информации о доступных узлах ИТКС. Для этого используется шаблон CAPEC-292 (*Host Discovery*), описывающий группу различных способов проведения сканирования хостов и портов. В эту группу, например, входят CAPEC-294 (*ICMP Address Mask Request*), CAPEC-299 (*TCP SYN Ping*), CAPEC-295 (*ICMP Timestamp Request*) и др. Следующая стадия анализа – поиск уязвимого программного обеспечения. Для этого используются следующие шаблоны: CAPEC-310 (*Scanning for Vulnerable Software*), CAPEC-312 (*Active OS Fingerprinting*), CAPEC-541 (*Application Fingerprinting*) и т. д. На третьей стадии проведения анализа используются как отдельные уязвимости из словаря CVE, так и шаблоны, например CAPEC-233 (*Privilege Escalation*) и т. д.

На этапе эксплуатации происходит первичное построение деревьев атак. Элемент модели атак, описывающий дерево атак, является вектором

$$M = \langle S, S_0, G, \pi \rangle, \quad (1)$$

где $S$ – множество состояний сети, $S_0$ – начальное состояние сети, $G$ – множество показателей, позволяющих определить процент достижения нарушителем своих целей при использовании построенного дерева атак, $\pi = S \cdot S$ – множество переходов между состояниями, которое можно определить имеющимися у злоумышленника атакующими действиями.

Узлы дерева атак задают возможные атакующие действия, связанные между собой в соответствии с тем, в каком порядке их может выполнять определенный нарушитель. Маршрут атаки является частью дерева атак и представляет собой последовательность состояний ИТКС $(S_0, S_1, ..., S_n)$, причем $(S_i, S_{i+1}) \in \pi \forall i \in [0, n]$.

В результате полученных деревьев атак и маршрутов воздействий оцениваются показатели защищенности. К ним относят:

– уязвимость ИТКС;
– слабость ИТКС;
– уязвимость ИТКС к атакам нулевого дня;
– поверхность атаки;
– процент узлов без критичных уязвимостей.

Для определения показателя уязвимости ИТКС необходимо на основе известных уязвимостей узлов и базовой оценки CVSS провести выборку таких узлов, базовый вектор оценки которых превышает 7,0.

Слабость хоста *CWcwss* определяется на основе стандартов «Общее перечисление слабых мест» (*Common Weakness Enumeration, CWE*) и «Общая система оценки слабых мест» (*Common Weaknesses Scoring System, CWSS*). Учитывая, что оценка в системе CWSS может лежать в диапазоне [0,100] и критичность этой оценки начинается от 60 единиц, показатель *CWcwss* вычисляется по формуле

$$CW_{cwss}(s) = \sum_{i=1}^{n} \max(0, CWSS_{cwss}(w_i) - 60) / n, \qquad (2)$$

где $s$ – узел ИТКС; $CWSS_{CWSS}(w_i)$ – оценка CWSS для слабого места $w_i$ узла $s$; $n$ – количество узлов ИТКС с оценкой CWSS выше 60.

Действия злоумышленника во время проведения воздействия характеризует такой показатель, как поверхность атаки – все возможные маршруты атаки, исходя из текущего положения нарушителя на дереве атак и его навыков.

Процент узлов без критичных уязвимостей (под критичными понимаются уязвимости, для которых базовая оценка CVSS – «высокая») определяется отношением количества узлов ИТКС без известных критичных уязвимостей к общему количеству узлов.

Общий уровень защищенности ИТКС также можно вычислить на основе риска для всех угроз ИТКС. Риск определяется как результат возможности (вероятности) угрозы и последствий ее реализации для всей ИТКС.

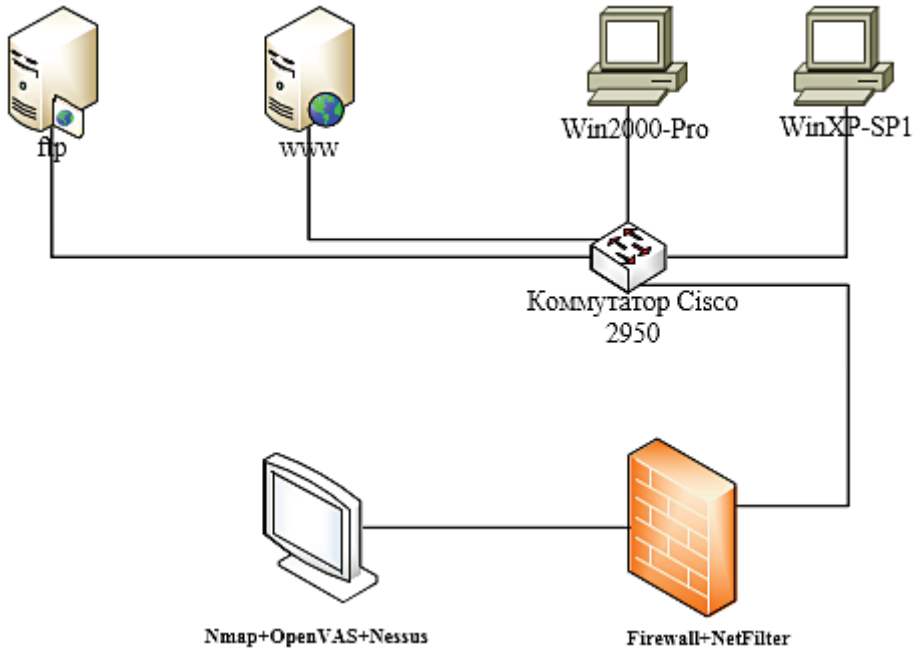
В самом общем виде методика расчета показателя «Уровень риска» выглядит следующим образом. Уровень риска атаки определяется как произведение вероятности успешной реализации атаки на ущерб, причиняемый в случае успешной реализации атаки. Вероятность успешной реализации атаки определяется исходя из навыков злоумышленника, надежности информации о событиях безопасности (показателей систем обнаружения вторжений), критичности атаки (определяется на основе базовой оценки CVSS) и потенциала атаки (определяется как отношение уже получившихся шагов воздействия к общему количеству шагов в атаке). Ущерб в случае успешной реализации атаки включает собственный ущерб (определяется на основе CVSS) и распространенный ущерб (определяется с использованием зависимостей сервисов). Полученный в результате уровень риска используется для принятия решения о необходимости применения дополнительных средств защиты ко всей ИТКС в целом или к ее отдельным узлам.

Приведем пример применения предлагаемого подхода к оцениванию уровня защищенности ИТКС, представленной на рисунке. В качестве исходного месторасположения узла, который будет оценивать защищенности ИТКС, выбрано место, находящееся за пределами этой сети. В процессе построения дерева атак моделировались три основные стадии:

- сбор информации (предварительная разведка);
- выявление топологической структуры и связей сервисов;
- выявление уязвимостей и дальнейшее оценивание защищенности.

На первом этапе отыскиваются взаимосвязи узлов, связанных с начальным месторасположением аналитика, и узлов, связанных с конечными хостами (на рисунке это ftp-сервер, www-сервер, WinXP, Win200-Pro).

Следующим этапом при оценивании защищенности является последовательное выявление уязвимостей на обнаруженных узлах. Примером выполнения данного этапа служит найденная уязвимость CVE-2012-0383, которая позволяет использовать службу Network Address Translation (NAT) программного обеспечения Cisco IOS для удаленного проведения воздействия злоумышленником, отправляя специально сформированные SIP-пакеты, тем самым вызывая «отказ в обслуживании». На примере выявленной уязвимости оценим показатели защищенности. Напомним, что показатель «уязвимость ИТКС» формируется выборкой из всех имеющихся уязвимостей сети (всего в представленной ИТКС было выявлено 48 уязвимостей). Базовый вектор уязвимости CVE-2012-0383 (AV: N/AC: L/Au: N/C: N/I: N/A: C) составляет 7,8 единиц, что означает присоединение этой оценки к общему показателю уязвимости ИТКС [21]. Из 48 уязвимостей, базовый вектор оценки которых превышает 7,0 еди-




Экспериментальная ИТКС

ниц, было выявлено 32, средняя оценка которых составляет 8,1 единиц, что означает высокую степень уязвимости исследуемой ИТКС.

Показатель слабости ИТКС формируется также на основе всех выявленных угроз и уязвимостей согласно стандарту CWE. Оценка CWSS описанной уязвимости составляет 84 единицы, что также говорит о высокой критичности выявленной угрозы. По формуле (2) можно рассчитать показатель слабости ИТКС (оценка CWSS выше 60 единиц оказалась в 36 уязвимостях):

$$CW_{cwss} = 2844 / 36 = 79.$$

Показатель слабости ИТКС равен 79 единицам, что тоже свидетельствует о высокой критичности исследуемой ИТКС.

Процент узлов без критичных уязвимостей составляет $(16/48) \cdot 100 = 33{,}3\,\%$.

После анализа исследуемой ИТКС было построено около 80 различных маршрутов атак (маршруты атак отличаются друг от друга набором использованных уязвимостей). В результате выбран маршрут, имеющий минимальные значения сложности и максимальные значения уровня доступа для каждого узла. На основе данных об этом маршруте рассчитан уровень защищенности ИТКС в целом. Оценка уровня защищенности для анализируемой ИТКС составило 3 из 4, где уровень 1 – максимальная степень защищенности.

Таким образом, общая рекомендация к исследуемой ИТКС – повысить уровень ее защищенности, используя основные меры повышения информационной безопасности. Наиболее слабым местом в ИТКС оказался www-сервер, так как через него прошли практически все маршруты атак.

## Заключение

В настоящей работе рассмотрены основные исследования в области оценивания защищенности сетей и выделены их основные показатели. На основе предложенных показателей, а также особенностей архитектуры системы анализа защищенности сформирована система методики расчета этих показателей, которые предполагаются для реализации в рамках системы оценки защищенности.

## Литература

1. ГОСТ Р 52653-2006. Информационно-коммуникационные технологии в образовании. Термины и определения. – М. : Стандартинформ, 2006. – 11 с.

2. Котенко И. В. Перспективные направления исследований в области компьютерной безопасности / И. В. Котенко, Р. М. Юсупов // Защита информации. Инсайд. – 2006. – № 2. – С. 46–57.

3. Blakely B. A. Cyberprints Identifying cyber attackers by feature analysis : Doctoral Diss. – Iowa State Univ. 2012.

4. Kumar S. An Application of Pattern Matching in Intrusion Detection / S. Kumar, E. H. Spafford // Tech. Rep. CSDTR 94013. The COAST Project. Department of Comput. Sci. Purdue Univ. – West Lafayette, 1994.

5. Iglun K. State Transition Analysis: A Rule-Based Intrusion Detection System / K. Iglun, R. A. Kemmerer, P. A. Porras // IEEE Trans. Software Eng. – 1995. – No. 21 (3).

6. Cohen F. Simulating Cyber Attacks, Defenses, and Consequences / F. Cohen // IEEE Symp. Security and Privacy. – Berkeley, CA. 1999.

7. Yuill J. Intrusion-detection for incident-response, using a military battlefield-intelligence process / J. Yuill, F. Wu, J. Settle, F. Gong, R. Forno, M. Huang, J. Asbery // Comput. Networks. – 2000. – No. 34.

8. Huang M.-Y. A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis / M.-Y. Huang, T. M. Wicks // First Int. Workshop on the Recent Advances in Intrusion Detection, Raid'98. – Louvain-la-Neuve, Belgium, 1998.

9. Schneier B. Attack Trees / B. Schneier // Dr. Dobb's J. – 1999. –Vol. 12.

10. Котенко И. В. Применение графов атак для оценки защищенности компьютерных сетей и анализа событий безопасности / И. В. Котенко, А. А. Чечулин // Системы высокой доступности. – 2013. – Т. 9, № 3. – С. 103–110.

11. Абрамов Е. С. Применение графов атак для моделирования вредоносных сетевых воздействий / Е. С. Абрамов, А. В. Андреев, Д. В. Мордвин // Изв. ЮФУ. Технические науки. – 2012. – № 1. – С. 165–174.

12. Moitra S. D. A Simulation Model for Managing Survivability of Networked Information Systems / S. D. Moitra, S. L. Konda // Tech. Rep. CMU/SEI-2000-TR-020 ESC-TR-2000-020. – 2000. – 47 p.

13. Chi S.-D. Network Security Modeling and Cyber Attack Simulation Methodology / S.-D. Chi, J. S. Park, K.-C. Jung, J.-S. Lee // Lecture Notes in Computer Sci. – Carnegie Mellon Univ., 2001. – Vol. 2119.

14. Templeton S. J. A Requires/Provides Model for Computer Attack / S. J. Templeton, K. Levitt // NSPW 2000 : Proc. of the 2000 Workshop on New Security Paradigms. – NY : ACM, 2000. – P. 31–38.

15. Morin B. M2d2 : A formal data model for ids alert correlation / B. Morin, L. Me, H. Debar, M. Ducasse // Lecture Notes in Comput. Sci. – Berlin : Springer-Verlag, 2002. – Vol. 1516. – P. 115–137.

16. Меры защиты информации в государственных информационных системах : методический документ (утв. ФСТЭК РФ 11.02.2014).

17. Peltier T. R. How to complete a risk assessment in 5 days or less / T. R. Peltier // Auerbach publ. – 2008. – P. 1–55.

18. http://cve.mitre.org (дата обращения 25.11.2015).

19. Банк данных угроз безопасности информации ФСТЭК РФ – URL : http://bdu.fstec.ru/threat (дата обращения 25.11.2015).

20. http://capec.mitre.org (дата обращения 25.11.2015).

21. http://bdu.fstec.ru/calc (дата обращения 25.11.2015).

# The method of estimation of the security of information and telecommunication

Shinkarenko A. F.
Military Space academy named after A. F. Mozhaisky
S. Petersburg, Russia
tonio87@rambler.ru

***Abstract.*** **The work considers the perspective directions of research in the field of security metrics and establishes key indicators, based on which the calculation of the security of the system. Proposed tiered approach to the assessment of security and their calculation methodology based on attack trees and dependency services.**

***Keywords:*** **security metrics, attack tree, risk assessment, vulnerability.**

## References

1. GOST R 52653-2006 *Informatsionno-kommunikatsionnye tekhnologii v obrazovanii. Terminy i opredeleniya* [Information and communication technologies in education. Terms and definitions], Moscow, Standartinform, 2006, 11 p.

2. Kotenko I. V., Yusupov R. M. Promising areas of research in the field computer security [Perspektivnie napravleniya issledovaniy v oblasti komp'yuternoy bezopasnosti], *Zashchita informacii. Insayd [Data protection. Inside]*, 2006, no. 2, pp. 46-57.

3. Blakely B. A. Cyberprints Identifying cyber attackers by feature analysis. Doctoral Dissertation: Iowa State Univ. 2012.

4. Kumar S., Spafford E. H. An Application of Pattern Matching in Intrusion Detection, *Tech. Rep. CSDTR 94 013. The COAST Project. Department of Comput. Sci. Purdue Univ.*, West Lafayette, 1994.

5. Iglun K., Kemmerer R. A., Porras P. A. State Transition Analysis: A Rule-Based Intrusion Detection System *IEEE Trans. Software Eng.*, 1995, no. 21 (3).

6. Cohen F. Simulating Cyber Attacks, Defenses, and Consequences, *IEEE Symp. Security and Privacy*, Berkeley, CA. 1999.

7. Yuill J., Wu F., Settle J., Gong F., Forno R., Huang M., Asbery J. Intrusion-detection for incident-response, using a military battlefield-intelligence process, *Comput. Networks*, 2000, no. 34.

8. Huang M.-Y., Wicks T. M. A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis, *First Int. Workshop on the Recent Advances in Intrusion Detection, Raid'98*, Louvain-la-Neuve, Belgium, 1998.

9. Schneier B. Attack Trees, *Dr. Dobb's J.*, 1999, Vol. 12.

10. Kotenko I. V., Chechulin A. A. The use of attack to evaluate the security of computer networks and analysis of security events [Primenenie grafov atak dlya otsenki zashchishchennosti komp'yuternykh setey i analiza sobytiy bezopasnosti], *Sistemy vysokoy dostupnosti [High Availability Systems]*, 2013, Vol. 9, no. 3, pp. 103-110.

11. Abramov E. S., Andreev A. V., Mordvin D. V. The use of attack graphs for modelling malicious network effects [Primenenie grafov atak dlya modelirovaniya vredonosnykh setevykh vozdeystviy], *Izvestiya YuFU [News SFU]*, 2012, no. 1, pp. 165-174.

12. Moitra S. D., Konda S. L. A Simulation Model for Managing Survivability of Networked Information Systems. Tech. Rep. CMU/SEI-2000-TR-020 ESC-TR-2000-020, 2000, 47 p.

13. Chi S.-D., Park J. S., Jung K.-C., Lee J.-S. Network Security Modeling and Cyber Attack Simulation Methodology, *Lecture Notes in Comput. Sci.*, 2001, Vol. 2119.

14. Templeton S. J., Levitt K. A Requires/Provides Model for Computer Attack, *NSPW 2000: Proc. of the 2000 Workshop on New Security Paradigms*, NY, ACM, 2000, pp. 31-38.

15. Morin B., Me L., Debar H., Ducasse M. M2d2: A formal data model for ids alert correlation, *Lecture Notes in Comput. Sci.*, Berlin, Springer-Verlag, 2002, Vol. 1516, pp. 115-137.

16. *Mery zashchity informatsii v gosudarstvennykh informatsionnykh sistemakh* [Rrotection of information in state information systems], methodical document FSTEK RF, from 11.02.2014.

17. Peltier T. R. How to complete a risk assessment in 5 days or less, Auerbach publ., 2008, pp. 1-55.

18. http://cve.mitre.org (accessed 25 Nov. 2015).

19. http://bdu.fstec.ru/threat (accessed 25 Nov. 2015).

20. http://capec.mitre.org (accessed 25 Nov. 2015).

21. http://bdu.fstec.ru/calc (accessed 25 Nov. 2015).

# Implementation of the Role Based Access Control in Application for Mobile Device on the Android OS Platform*

Diasamidze S. V., Kuzmenkova E. Yu., Kuznetsov D. A., Sarkisyan A. R.
Petersburg State Transport University
St. Petersburg, Russia
sv.diass99@ya.ru

***Abstract.*** **The article is dedicated to solving the problem of privacy of personal data stored on the mobile device. We consider the method of the personal information protection based on role-based access control, it's advantages and software implementation on the Android OS platform.**

***Keywords:*** **Information security, Data security, Role based access control, Confidentiality.**

## Introduction

Modern development of information technologies and the spread of mobile devices has led to the fact that modern mobile device – smartphone or tablet – is used as a universal personal device, that includes mobile office, entertainment center and Internet work tool. There is a huge amount of personal information stored in memory of the smartphone: contacts of your colleagues, friends and relatives with their personal data; call history; corporate correspondence; settings of the Wi-Fi access points, which are located within the habitat of the owner; social network applications (often with saved passwords); Bank details or mobile/SMS banking, photos, videos, notes, etc. This concentration of business and personal data leads to the fact that the abstract value of information outweighs the price of the device itself. That is why the task of information protection mobile device is extremely important [1, 2].

This article describes one way of solving the problem of confidentiality of personal data stored on personal mobile device.

## The problems of security

One of the security problems appear in situation when it is necessary to provide the stranger with a temporary access to the device. For example, after purchasing a new smartphone acquaintances or friends ask to see and appreciate it. In this situation, on the one hand, it is uncomfortable to refuse them, and it would be wrong to give outsiders access to confidential information stored on your phone, such as private messages, photos and videos, accounts, contacts, etc. The solution to this problem would be to create a guest account, which would have limited access to the above resources.

Also it often happens that a child (son, daughter, younger brother or sister) asks to play games on your phone or tablet. However there is a possibility that while operating the phone he will exit the game, and view different application, accidentally reset settings or delete any important information. You can provide a limited account, which will have available only game apps.

Currently on the market there are apps with similar functionality, such as AppLock, Leo Privacy Guard, etc. But they contain other principles of limiting access to data. You have the option to set separate passwords for each application that from our point of view is not convenient and appropriate. Also, as a result of our testing it became clear that if you turn off your phone, the settings of these applications are discarded, that violates the data protection principles. In our application, Allock we tried to eliminate such errors.

Certainly, as with any role-based access control the main will be the account of the device owner, who is granted with full access rights to all resources and applications.

## Role based access control

As a base mechanism we selected the role-based access control. It's basic idea is to maximize the approximation of the system logic to the actual separation of staff's functions in the organization, which means that the method of role-based access control monitors user access to information based on the types of their activities in the system. The subjects' rights of access to the objects are grouped considering the specifics of their application, forming a role.

To analyze and study properties of systems role-based access right the mathematical models are used [3].

The base model of role-based access control defines the main principles and elements of such models.

Main elements:

$U$ – set of users;

$R$ – set of roles;

$P$ – set of access rights on the objects of the computer system;

$S$ – set of user's sessions;

$PA$ – function, that defines a set of access rights for each role: $PA$: $R \to 2^P$, wherein for each $p \in P$ exists $r \in R$ such that $p \in PA(r)$.

$UA$ – function, that defines for each user a set of roles that he can be authorized: $UA$: $U \to 2^R$.

For completeness of the mathematical model the basic functionsare introduced:

*user*: $S \to U$ – function that determines for each user session on behalf of which it's activated.

*roles*: $S \to 2^R$ – function that defines for the user a set of roles to which he can be authorized in this session.

In the basic model of role-based access control the following rules are defined:

* This article is published with the support of Federal state budget educational institution of higher professional education "Petersburg State Transport University of Emperor Alexander I" of initiative research works of student research teams.

1. One subject can have multiple roles.
2. One role can have multiple subjects.
3. One role can have multiple permissions.
4. One permission can belong to multiple roles [4].

One of the important mechanisms of the basic model of access rights differentiation are restrictions on many roles that can be authorized by the user, or to which he authorizes in the same session. This mechanism is also necessary for widespread use of the basic model of role differentiation, because it provides better conformity toused in computer systems technologies of data processing [5].

The relationships between the structural elements of the basic model role-based access rights, as shown in fig. 1.

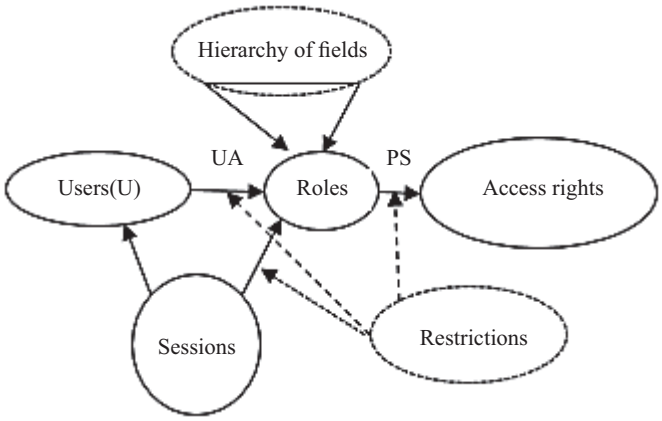



Fig. 1. The structure of the role-based access control

The simplified model of role-based access control can be represented as follows (fig. 2).

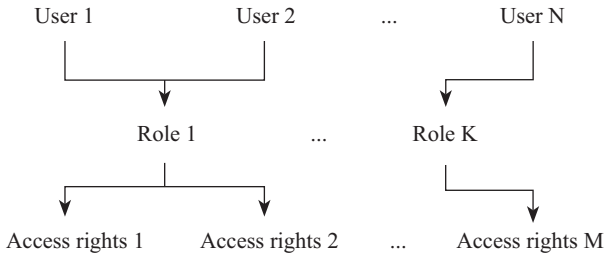



Fig. 2. The simplified model of role-based access control

## The advantages of the role-based access control

Role-based access control is neutral to specific types of rights and methods of their verification; it can be considered as an object oriented framework that facilitates administration, because it enables you to make subsystem access controlled by arbitrarily large number of users, primarily through the establishment of relationships between roles, similar to inheritance in object oriented systems. In addition, the number of roles should be much less than users. As a result, the number of administered connections becomes proportional to the sum (not the multiplication) of the number of users and objects that is impossible to decreasein the order of magnitude [6].

While developing applications for mobile devices model of the role-based access rights is the most convenient, reliable to protect user's personal data. It has a number of advantages over the other security policies.

First, the simplicity of administration. In the classical models of access control the rights to perform certain operations on the object are registered for each user or group of users. In role model the separation of the concepts "role" and "user" allows you to break the task into two parts: the definition of user roles and defining the rights of access to the object for the role. This approach greatly simplifies the process of administration, because when you change the scope of responsibility of the user it is enough to remove his old role and assign the other corresponding to his new rights. For example, in Allock for the role "Guest" at first was installed access to five gaming applications, but the owner at any time may extend or narrow the range of access of this user [7].

Secondly, the principle of least privilege. A role model allows the user to register in the system with minimal role that allows him to perform the required tasks. Users with multiple roles, do not always require all possible privileges to perform specific tasks. According to the principle of least privilege, the user receives only those access rights which he needs to perform a specific task. This requires to clarify the objectives of the task, the set of privileges required to execute and restrict user privileges by this set. Prohibition of user privileges that are not required to perform the task, avoids opportunities to circumvent the system security policy. That is, the user in role "Guest" can not access your correspondence or your schedule for the day/week, phone settings, he has only the access to the function "make a call", "send SMS", as well as access to some gaming applications.

## The implementation of the application

While adapting mechanism of differentiation of access rights based on roles we were asked to identify some typical roles with pre-defined powers, such as "Owner", "Guest", "Children". The user is also given the opportunity to introduce new roles and define their access rights to resources of the mobile device

As an environment for developing application was used Android Studio with minimum support operating system version 4.0, which based on the published statistics, allows you to use the app on 99% of active devices under Android OS (fig. 3) [8].

| Version | Codename | API | Distribution |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.3% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 5.6% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 5.1% |
| 4.1.x | Jelly Bean | 16 | 14.7% |
| 4.2.x | | 17 | 17.5% |
| 4.3 | | 18 | 5.2% |
| 4.4 | KitKat | 19 | 39.2% |
| 5.0 | Lollipop | 21 | 11.6% |
| 5.1 | | 22 | 0.8% |

Fig. 3.Statistics of Android version using. March 2015

The design meets the guidelines of (the manufacturer's instructions on formalization) Google material design, supported from version 5.0.

General algorithm of the application looks as follows:

1. Installation of the application Allock on a mobile device, then the main menu is automatically loaded (fig. 4).

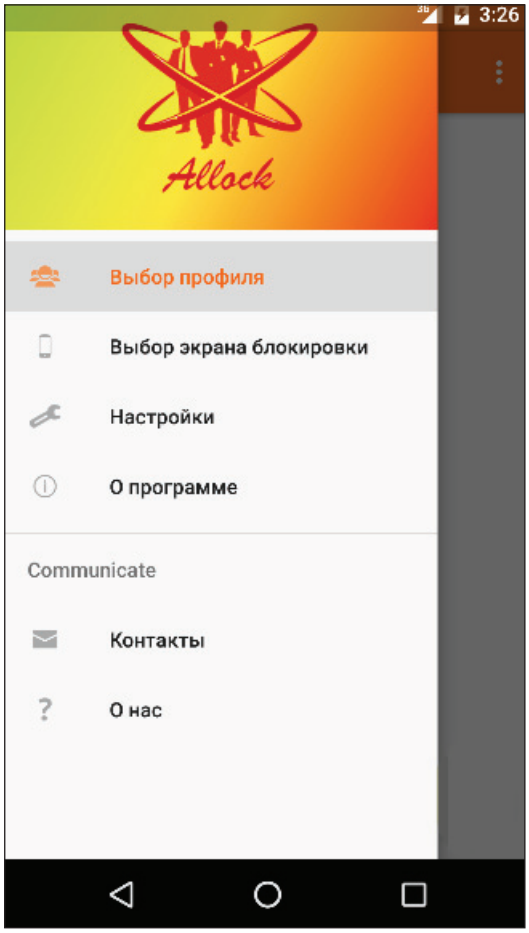



Fig. 4. Main menu of the Allock application

2. Selecting and customizing access for one of the model groups, by defaultgroups such as «Owner», «Guest», «Children» are installed in application with appropriate access rights (fig. 5). You can create a new group at the discretion of the owner of the device.

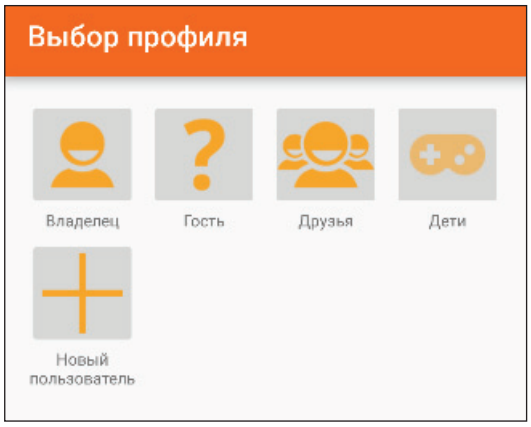



Fig. 5. Menu of group choice

3. Working with the list of installed applications. After selecting a specific group there will appear a list of all installed applications. The applicationsthat would be available for the user authorized under account of this groupshould be selected in the appropriate field [6].

4. Setting a unique password for the group. To increase the level of security in the Allock application there is a uniqueness rule of used authenticators -passwords for each group should be different. The measure was introduced in the absence of the name of the group on the lock screen for more simple use the application (the user needs only to enter a password, which is an identifier and authenticator of the group at the same time).

For receiving a list of all installed applications we use classes *UserApps.java* (fig. 6) and *AppAdapter.java* (fig. 7). *UserApps.java*is responsible for the build of screen form (fig. 8), and as we get the list of applications in this class, then this class is inherited from *ListActivity* class. The important methods are *checkForLaunchIntent* and class *LoadApplications* (fig. 9), which loadtheappsinstalled on device. *AppAdapter.java* is the class adapter for *UserApps.java.* The adapter isused to build some dynamic data (in our case it is the listwith unknown length) of objects with information about applications. It takes the provided data and places them in order, immediately setting the described components (we have this *ImageView*, a *TextView* for the icons and app name).

The key method is *getView* in which the process described abovetakes place.

## Conclusion

The important point in the development of the app was to create our own lock screen, attached to the all the installed profiles using the password as identifier-authenticator. The screen extends the capabilities of standard one, allows you to recognize a few passwords and activate the appropriate account (group) with a specific set of authorized applications.

The advantages of our developed program are simplicity of use, automation of complicated processes of access rightsdifferentiation that is absolutely imperceptible for users, flexibility and easiness of settings.

## References

1. Artamonov V.A. Security problems of mobile devies, systems and applications [Problemy bezopasnosti mobilnykh ustroystv, system i prilozheniy], *IT Bezopasnost [IT Security]*. Available at: http://itzashita.ru/mobilnyie-ustroystva/bezopasnost-mobilnyih-ustroystv-sistem-i-prilozheniy.

2. Yakushin P. Secutity of the mobile enterprise [Bezopasnost mobilnogo predpriyatiya], *Otkryt yesistemy. SUBD [Open systems. SUBD]*, 2013, no. 1, pp. 22-26.

3. Devyanin P.N. *Modeli bezopasnosti computernykh sistem. Upravlenie dostupom i informatsyonnymi potokami* [Security models of computer systems. Control of access and informational threads], 2nd ed., Moscow, Goryachaya liniya – Telekom, 2013, 338 p.

4. *Upravlenie dostupom na osnove roley* [Role based access control]. Available at: https://ru.wikipedia.org/wiki/Управление_доступом_на_основе_ролей (accessed 11 Feb. 2016).

```
public class UserApps extends ListActivity {

    private PackageManager packageManager = null;
    private List applist = null;
    private AppAdapter listadapter = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user_apps);
        packageManager = getPackageManager();
        new LoadApplications().execute();
    }

    public void onClickStart(View view) {
        Intent intent = new Intent(this, LoginActivity.class);
        startActivity(intent);
    }

    private List<ApplicationInfo> checkForLaunchIntent(List<ApplicationInfo> list) {

        ArrayList appList = new ArrayList();
        for(ApplicationInfo info : list) {
            try{
                if(packageManager.getLaunchIntentForPackage(info.packageName) != null) {
                    appList.add(info);
                }
            } catch(Exception e) {
                e.printStackTrace();
            }
        }

        return appList;
    }
```

Fig. 6. Class UserApps

```
public class AppAdapter extends ArrayAdapter{

    private List appList = null;
    private Context context;
    private PackageManager packageManager;

    public AppAdapter(Context context, int resource,
                      List objects) {
        super(context, resource, objects);
        this.context = context;
        this.appList = objects;
        packageManager = context.getPackageManager();
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        View view = convertView;

        if(null == view) {
            LayoutInflater layoutInflater = (LayoutInflater) context
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = layoutInflater.inflate(R.layout.list_item, null);
        }
        ApplicationInfo data = (ApplicationInfo) appList.get(position);

        if(null != data) {
            TextView appName = (TextView) view.findViewById(R.id.app_name);
            ImageView iconView = (ImageView) view.findViewById(R.id.app_icon);

            appName.setText(data.loadLabel(packageManager));
            iconView.setImageDrawable(data.loadIcon(packageManager));
        }
        return view;
    }
}
```

Fig. 7. Class AppAdapter

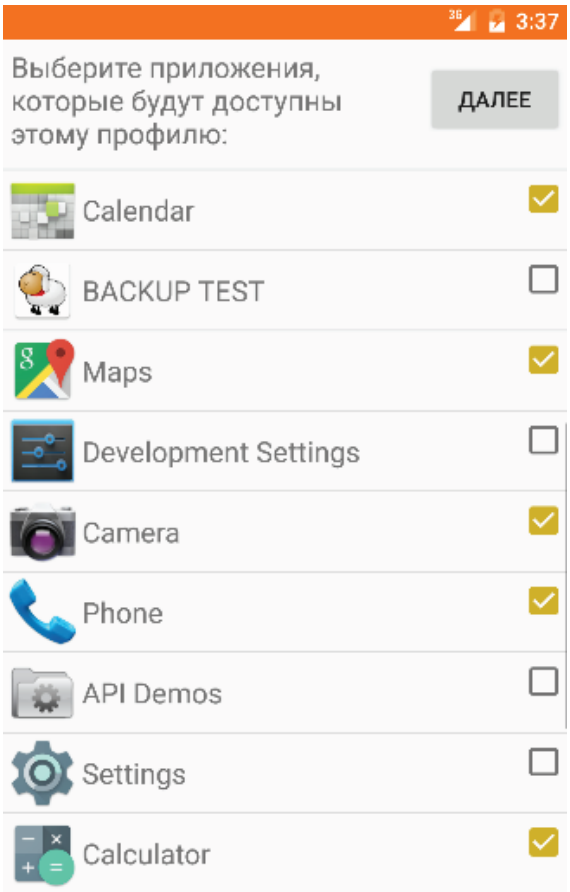



Fig. 8. Working with the list of installed applications

```
private class LoadApplications extends AsyncTask<Void, Void, Void> {

    private ProgressDialog progress = null;

    @Override
    protected Void doInBackground(Void... params) {

        applist = checkForLaunchIntent
                (packageManager.getInstalledApplications(PackageManager.GET_META_DATA));
        listadapter = new AppAdapter(UserApps.this, R.layout.list_item, applist);
        return null;
    }
}
```

Fig. 9. Class LoadApplications

5. Stepanenko I. D. *Modeli upravleniya dostupom: diskretnaya, mandatnaya, rolevaya* [Models of access control: discrete, mandatory, role-based]. Available at: http://gman1990.ru/articles.php?article_id=83 2015.

6. *Ponyatie I naznachenie modeli bezopasnosti* [Concept and purpose of security model]. Available at: http://www.studfiles.ru/preview/2880350/page:4 (accessed 11 Feb. 2016).

7. Gaydamakin N.A. *Razgranichenie dostupa k informatsii v komputernykh sistemakh* [Access distinction to information in computer systems], Ekaterinburg, Ural Univ. Publ., 2003, 328 p.

8. *Statistika versiy Andriod: mart 2015* [Android version statistics: march 2015]. Available at: http://puregoogle.ru/2015/03/03/http://puregoogle.ru/2015/03/03/statistika-versij-android-mart-2015 (accessed 20 Feb. 2016).

# Реализация ролевой политики разграничения прав доступа в приложении для мобильного устройства под управлением ОС Android

Диасамидзе С. В., Кузьменкова Е. Ю., Кузнецов Д. А., Саркисян А. Р.
ФГБОУ ВО ПГУПС
Санкт-Петербург, Россия
sv.diass99@ya.ru

***Аннотация.*** **Статья посвящена решению проблемы обеспечения конфиденциальности личных данных, хранящихся на мобильном устройстве. Рассматривается вариант метода защиты персональной информации, основанный на ролевой политике разграничения прав доступа, его преимущества и программная реализация на платформе ОС Android.**

***Ключевые слова:*** **защита информации, безопасность данных, ролевая политика разграничения прав доступа, конфиденциальность.**

Литература

1. Артамонов В. А. Проблемы безопасности мобильных устройств, систем и приложений / В. А. Артамонов // ИТ Безопасность. – URL : http://itzashita.ru/mobilnyie-ustroystva/bezopasnost-mobilnyih-ustroystv-sistem-i-prilozheniy.

2. Якушин П. Безопасность мобильного предприятия / П. Якушин // Открытые системы. СУБД. – 2013. – № 1. – С. 22–26.

3. Девянин П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками : учеб. для вузов / П. Н. Девянин. – 2-е изд., перераб. и доп. – М. : Горячая линия – Телеком, 2013. – 338 с.

4. Управление доступом на основе ролей. – URL : https://ru.wikipedia.org/wiki/Управление_доступом_на_основе_ролей (дата обращения 11.02.2016).

5. Степаненко И. Д. Модели управления доступом : дискретная, мандатная, ролевая. – URL : http://gman1990.ru/articles.php?article_id=83.

6. Понятие и назначение модели безопасности. – URL : http://www.studfiles.ru/preview/2880350/page:4 (дата обращения 11.02.2016).

7. Гайдамакин Н. А. Разграничение доступа к информации в компьютерных системах / Н. А. Гайдамакин. – Екатеринбург : Изд-во Урал. ун-та, 2003. – 328 с.

8. Статистика версий Android : март 2015. – URL : http://puregoogle.ru/2015/03/03/statistika-versij-android-mart-2015 (дата обращения 20.02.2016).

# Расчет резерва времени для выполнения защитных мероприятий информационного объекта

Живов А. Д., Семенов А. Н., Гвоздева Г. А.
ФГБУ «4ЦНИИ» Минобороны России
Королев, Россия
4cnii@mil.ru

***Аннотация.*** **На основе игровой модели информационной безопасности рассматривается система «нападение – защита». «Нападение» представлено последовательностью независимых случайных величин – интервалов времени выполнения целевых операций. Рассчитывается суммарная продолжительность этих интервалов – резерв времени для выполнения защитных мероприятий объекта. Начало отсчёта – обнаружение события разведывания объекта, конец – момент завершения возможной компьютерной атаки. Полученные результаты могут быть полезны при определении условий (требований) встраивания новых программно-аппаратных средств в общую систему обеспечения безопасности объектов.**

***Ключевые слова:*** **нападение, защита, операция, интервал, разведывание, распределение, композиция, расчет.**

## Введение

При построении систем защиты информации используются разнообразные методы и модели, среди них важное место занимают игровые модели. В частности, игровая модель системы защиты информации, которая используется для выбора решения, обеспечивающего оптимальное соотношение между затратами на средства защиты и снижением риска эксплуатации системы, рассматривается в [1].

В статье [2] изложены принципы количественного оценивания информационной безопасности с позиций теории игр. Введены определения игровой матрицы и ее количественной интерпретации. Представлены методики анализа, планирования и проектирования информационной безопасности на основе предложенных моделей.

В диссертационной работе [3] разработана игровая модель для обеспечения оптимальности системы защиты информации. Модель поиска оптимального проекта представляет собой игру статистика с природой, где под статистиком понимается владелец информационной системы, а под природой – злоумышленники.

В статье [4] рассматривается теоретико-игровая модель, с помощью которой решается задача оптимального выделения ресурсов кибер-безопасности, таких как время администратора для разных задач и др.

В работе [5] исследуются модели безопасности взаимодействий для различных вариантов игры, таких как слабое звено или лучший выстрел, чтобы представлять практические сценарии безопасности.

В [6] авторы решают проблему нахождения оптимального оборонительного покрытия. Из-за неопределенности действий атакующего они пытаются определить такое покрытие, максимизируя наихудший выигрыш над целями в наборе атаки. В игре для кибер-безопасности можно считать, что защитник не знает выигрыша для злоумышленника и может только оценить затраты.

В настоящей статье рассматривается система «нападение – защита», где сторона нападения представляет собой сложный пространственно распределенный комплекс средств, реализующий процесс разведывания, передачи, приема, обработки и интерпретации разведданных с возможным принятием и выполнением решения о компьютерной атаке объекта разведывания.

Для обеспечения безопасности последнего жизненно важно оперативно рассчитать вероятностно-временные характеристики случайной величины – продолжительности этого процесса (т. е. искомого резерва времени). В данной статье приведена схема расчета суммарной плотности распределения упомянутой величины и соответственной функции распределения. Полученные результаты содержат полную информацию для принятия решения о выполнении защитных мероприятий.

## Модель «нападения»

Модель «нападения» представлена последовательностью независимых случайных величин $t_1, t_2, t_3, ..., t_i, ..., t_n$ – значений интервалов времени выполнения целевых операций компонентами указанного комплекса.

К значениям $t_i$ интервалов времени «прикреплены» соответственные распределения плотности вероятностей $\varphi(t_i)$ (рис. 1). Распределение $\varphi(T_\Sigma)$ описывает плотность распределения суммы $T_\Sigma$ – искомого резерва времени (также случайной величины). Интеграл распределения $\varphi(T_\Sigma)$ является функцией распределения $F(T_\Sigma)$.

В зависимости от меняющихся условий обстановки сторона «нападения» меняет состав и конфигурацию компонентов разведывательно-атакующего комплекса, порождая тем самым различные (ситуационные) последовательности, отличающиеся составом интервалов $t_i$, их распределениями $\varphi(t_i)$ и в конечном счёте – распределениями $\varphi(T_\Sigma)$ и $F(T_\Sigma)$.

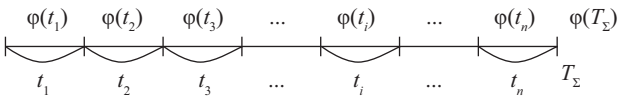



Рис. 1. Схема описания плотности $\varphi(T_\Sigma)$

## Модель «защиты»

Сторона «защиты» обладает:

• достаточно достоверным знанием структуры существующих комплексов стороны «нападения»;

• свойством в реальном времени обнаруживать события разведывания объекта информатизации;

• приблизительным знанием временны́х характеристик операций, выполняемых потенциальным нарушителем;

• возможностью выполнять защитные мероприятия за допустимое время $T^*$, располагая резервом времени $T_\Sigma$.

## Постановка задачи

Ставится задача: рассчитать распределения $\varphi(T_\Sigma)$ и $F(T_\Sigma)$ для обоснования решения по выполнению защитных мероприятий согласно критерию

$$F(T_\Sigma) = P\left[T_\Sigma >> T^*\right], \qquad (1)$$

где $P\left[T_\Sigma >> T^*\right]$ – вероятность того, что объект разведывания (возможная цель) располагает достаточным резервом времени для выполнения защитных мероприятий. Символ >> означает, что «лицо, принимающее решение», неформально оценивая риск возможной атаки, назначает для суммы $T_\Sigma$ некоторый «запас», значение которого зависит от степени доверия к результатам расчёта рассматриваемых распределений.

## Решение задачи

Применительно к заданной ситуационной последовательности схема решения задачи представляется следующим образом.

Распределение суммы независимых случайных величин $\varphi(T_\Sigma)$, будучи композицией распределений $\varphi(t_i)$, рассчитывается по правилам вычисления их свёрток. Сначала вычисляется распределение $\varphi(t_{1,2})$ суммы первых двух независимых случайных величин (значений $t_1$ и $t_2$) путём расчёта свёртки согласно формуле [7, 8]

$$\varphi(t_{1,2}) = \varphi(t_1) \cdot \varphi(t_2) = \int_0^\infty \varphi(t_1)\varphi(t_2 - t_1)dt, \qquad (2)$$

где * – символ свертки.

Далее рассчитывается последовательность свёрток:

$$\begin{gathered}\varphi(t_{1,2}) \cdot \varphi(t_3) = \varphi(t_{1,2,3});\\ \varphi(t_{1,2,3}) \cdot \varphi(t_4) = \varphi(t_{1,2,3,4});\\ \varphi(t_{1,2,3,\ldots,t_{n-1}}) \cdot \varphi(t_n) = \varphi(T_\Sigma).\end{gathered} \qquad (3)$$

Функция распределения $F(T_\Sigma)$ рассчитывается интегрированием полученного распределения $\varphi(T_\Sigma)$. Результаты расчёта, описывающие распределения $\varphi(T_\Sigma)$ и $F(T_\Sigma)$, представляются в символьном (формула) и в графическом виде. Задача решена.

Аналогичным образом рассчитываются другие ситуационные последовательности.

Полученные результаты содержат полную информацию, необходимую для анализа и принятия решения согласно критерию (1).

Достоверность результатов расчёта зависит от точности приведения модели «нападения» в соответствие с составом и вероятностно-временными характеристиками компонентов действующего разведывательно-атакующего комплекса (задача структурной и параметрической идентификации).

В идеале «защита» должна располагать структурой (формулой) и параметрами каждого распределения $\varphi(t_i)$.

Однако получить от действующих компонентов рассматриваемого комплекса опытные данные относительно продолжительности выполнения каждой операции и в достаточном объёме проблематично. Это обстоятельство актуализирует задачу добывания необходимых сведений из всех доступных источников.

На практике обычно исходят из допущения, что случайные величины $t_i$ распределены по экспоненциальному закону:

$$\varphi(t_i) = \lambda_i \exp(e^{-\lambda_i t_i}), \qquad (4)$$

где $\lambda_i = 1/\tau_i$ – параметр экспоненциального распределения; $\tau_i$ – среднестатистическое значение времени выполнения $i$-й операции.

В этом случае распределения $\varphi(T_\Sigma)$ и $F(T_\Sigma)$ описываются обобщённым законом Эрланга $n$-го порядка [7], согласно которому математическое ожидание (МО) и дисперсия (Д) указанных распределений рассчитываются по простым формулам:

$$MO = \sum_1^n 1 \Big/ \lambda_i = \sum_1^n \tau_i;\ \text{Д} = \sum_1^n 1 \Big/ \lambda_i^2 = \sum_1^n \tau_i^2. \qquad (5)$$

Экспоненциальное распределение случайной величины $t_i$ предполагает, что в большинстве случаев $i$-я операция выполняется относительно быстро, что не всегда соответствует практике. Рассмотрим возможность применения другого вида распределения.

В нашем случае наибольший вклад в сумму $T_\Sigma$ вносят операции, продолжительность которых варьируется в известных пределах от минимального ($t_{\min}$) до максимального ($t_{\max}$) значения и может быть очень велика. На практике значения этих пределов с приемлемой точностью известны по данным от отечественных аналогов.

Для формального описания подобного вида распределений мировое сообщество испытателей разработало двухпараметрическое семейство бета-распределений [9]. Каждый экземпляр такого семейства представлен аналитическим выражением, значения параметров которого определяют форму графика и подобраны сообразно физической сущности рассматриваемого процесса.

Плотность бета-распределения описывается формулой

$$\varphi(t) = \frac{1}{B(\alpha,\beta)} t^{\alpha-1}(1-t)^{\beta-1}, \qquad (6)$$

где $B(\alpha,\beta)$ – бета-функция. В семействе бета-распределений в нашем случае наиболее приемлем экземпляр с параметрами формы $\alpha = 2$ и $\beta = 5$ (рис. 2).

Бета-функция рассчитывается через Гамма-функции:

$$B(\alpha,\beta) = B(2, 5) = B(\alpha)B(\beta)/B(\alpha+\beta) = B(2)\cdot B(5)/B(7).$$

Математическое ожидание и дисперсия бета-функции равны, соответственно, $\alpha/\alpha + \beta$ и $\alpha\beta/(\alpha+\beta)^2(\alpha+\beta+1)$.

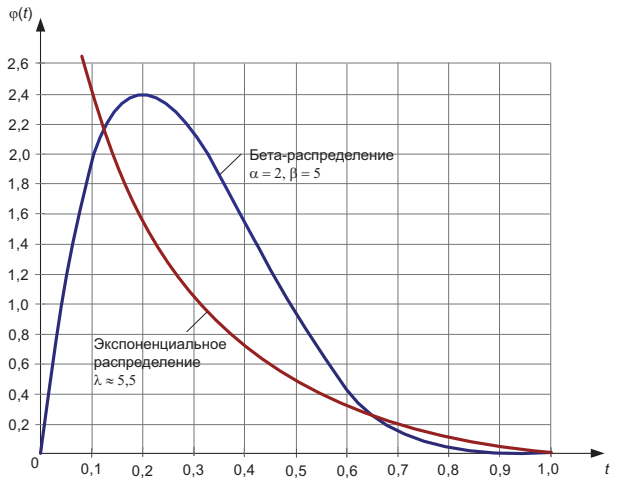



Рис. 2. Плотность экспоненциального и бета-распределений

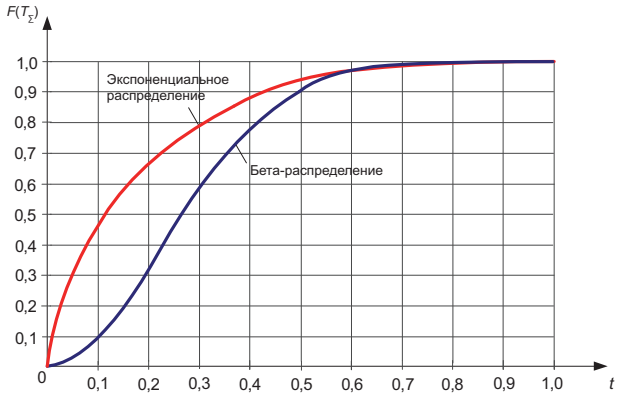



Рис. 3. Функции экспоненциального и бета-распределений

На рис. 3 совмещены два графика плотности вероятности случайной величины $t_i$: для стандартного бета-распределения, сосредоточенного на условной временно́й базе – отрезке от нуля до единицы, и экспоненциального распределения, параметр которого ($\lambda \approx 5{,}5$) подобран так, чтобы график этого распределения уместился на общей временно́й базе.

Гамма-функции рассчитываются по правилу [10]: $B(n) = (1-n)!$. Для значений $n = 2, 5, 7$ имеем: $B(2) = 1$; $B(5) = 24$; $B(7) = 720$.

Отсюда $B(\alpha,\beta) = B(2,5) = 24/720 = 1/30$, а выражение (6) приобретает вид $\varphi(t) = 0{,}33t(1-t)^4$.

На рис. 3 приведены два графика соответственных функций распределения.

Формулы и значения числовых характеристик для вышеприведённых распределений сведены в таблицу.

При практических расчётах отрезку от нуля до единицы (условной временно́й базе) следует придавать смысл от $t_{\min}$ до $t_{\max}$ временных единиц (минут, часов и др.).

Сравнительные характеристики экспоненциального и бета-распределений

| Распределение | Математическое ожидание | Дисперсия |
|---|---|---|
| Бета-распределение | $\alpha/\alpha+\beta \approx 0{,}28$ | $\alpha\beta/(\alpha+\beta)^2(\alpha+\beta+1) \approx 0{,}025$ |
| Экспоненциальное | $1/\lambda \approx 0{,}18$ | $1/\lambda^2 \approx 0{,}033$ |

Нетрудно оценить, насколько велико различие суммарных значений $T_\Sigma$ для рассмотренных распределений. Целесообразность описания бета-распределением случайных величин $t_i$, для которых с приемлемой точностью известны значения их границ, подтверждается, если измеренные опытным путём значения $t_i$) будут группироваться вблизи теоретического значения математического ожидания, равного 0,28 ($t_{\max} - t_{\min}$).

## Заключение

Естественным продолжением данной статьи должна стать серия модельных экспериментов, реализующих расчёты согласно последовательности (3) при различных комбинациях исходных данных.

Результаты расчёта необходимо проанализировать и тематически обработать, чтобы:

• определить основные закономерности изменения формы графиков распределений $\varphi(T_\Sigma)$ и $F(T_\Sigma)$;

• подтвердить или отвергнуть утверждение о применимости обобщённого закона Эрланга $n$-го порядка к любым распределениям;

• определить правила практического применения «защитой» результатов расчёта.

Приведенные в статье результаты могут быть полезны при определении условий (требований) встраивания новых программно-аппаратных средств в общую систему обеспечения безопасности информационных объектов.

## Литература

1. Герасименко В. А. Основы защиты информации / В. А. Герасименко, А. А. Малюк. – М. : МИФИ, 1997. – 537 с.

2. Шахов В. Г. Игровые и топологические модели информационной безопасности / В. Г. Шахов, А. В. Морозов, А. П. Тиунов, А. Н. Громов // Изв. Транссиба. – 2015. – № 1 (21). – С. 89–94.

3. Арьков П. А. Разработка комплекса моделей для выбора оптимальной системы защиты информации в информационной системе организации : дис. … канд. техн. наук. – Волгоград, 2009. – 185 с.

4. Fielder A. Game Theory Meets Information Security Management / A. Fielder, E. Panaousis, P. Malacaria, C. Hankin1, F. Smeraldi; eds. N. Cuppens-Boulahia et al. // IFIP Int. Federation for Information Proc. SEC 2014. – IFIP AICT 428 б 2014. – P. 15–29.

5. Grossklags J. Secure or insure? A game-theoretic analysis of information security games / J. Grossklags, N. Christin, J. Chuang // Proc. of the 17th Int. Conf. on World Wide Web «WWW 2008». – ACM, 2008. – P. 209–218.

6. Kiekintveld C. Security games with interval uncertainty / C. Kiekintveld, T. Islam, V. Kreinovich // Proc. 12th Int. Conf. on Autonomous Agents and Multiagent Systems «AAMAS 2013». – Int. Foundation for Autonomous Agents and Multiagent Systems, Richland, 2013. – P. 231–238.

7. Вентцель Е. С. Теория вероятностей и её инженерные приложения / Е. С. Вентцель, Л. А. Овчаров. – М. : Наука, 1988. – 480 с.

8. Шор Я. Б. Статистические методы анализа и контроля качества и надёжности / Я. Б. Шор. – М. : Сов. Радио, 1962. – 552 с.

9. Бета-распределение. http://ru.wikipedia.org

10. Бронштейн И. Н. Справочник по математике / И. Н. Бронштейн, К. А. Семендяев. – М. : Наука, 1962. – 608 с.

# The Calculation of Time Reserve for the Execution of Protective Actions of Information Object

Zhivov, A. D., Semenov A. N., Gvozdeva G. A.
FSBI "4 CRI defense of Russia"
Korolyov, Russia
4cnii@mil.ru

***Abstract.*** **Based on the game model of information security system is considered "attack-defense". «Attack» is a sequence of independent random variables – time intervals for performing target operations. The total length of these time intervals – time reserving for the implementation of protective actions of an object – is calculated. The beginning of the countdown – the event consisting in the fact that the object is detected, the end of the countdown – the moment of completion of a possible computer attack. The obtained results can be useful in determining the conditions (requirements) of embedding new software and hardware into the overall system of safety of objects.**

***Keywords:*** **attack, defense, interval, detection, distribution, composition, calculation.**

## References

1. Gerasimenko V. A., Maluk A. A. *Osnovy zashchity informatsii* [Information Protection Basics], Moscow, MIFI, 1977, 537 p.

2. Shakhov V. G., Morosov A. P., Tiunov A. N. Gambling and topological information security model [Igrovye i topologicheskie modeli informatsionnoi bezopasnosti], *Izvestiia Transsiba [Proc. Trans-Sib]*, 2015, no. 1 (21), pp. 89-94.

3. Arkov P. A. *Razrabotka kompleksa modelei dlya vybora optimal'noi sistemy zashchity informatsii v informatsionnoy sisteme organizatsii* [Development of complex models to select optimal system of information protection in information systems of the organization. Diss. on competition of a sci. degree Cand. tech. Sci.], Volgograd, 2009, 185 p.

4. Fielder A., Panaousis E., Malacaria P., Hankin C., Smeraldi F. Game Theory Meets Information Security Man-agement; eds. N. Cuppens-Boulahia et al. *IFIP Int. Federation for Information Proc.*, SEC 2014, IFIP AICT 428, pp. 15-29.

5. Grossklags J., Christin N., Chuang J. Secure or insure? A game-theoretic analysis of information security games, *Proc. 17th Int. Conf. on World Wide Web "WWW 2008"*, ACM, 2008, pp. 209-218.

6. Kiekintveld C., Islam T., Kreinovich V. Security games with interval uncertainty, *Proc. 12th Int. Conf. on Autonomous Agents and Multiagent Systems "AAMAS 2013"*, Int. Foundation for Autonomous Agents and Multiagent Systems, Richland, 2013, pp. 231-238.

7. Venttsel' E.S., Ovcharov L.A. *Teoriya veroiatnostey i eyo inzhenernye prilozheniya*. [Probability theory and its engineering applications], Moscow, Nauka, 1988, 480 p.

8. Shor Ia.B. Statisticheskie metody analiza i kontrolia kachestva i nadezhnosti [Statistical methods of analysis and control of quality and reliability], Moscow, Sovietskoye radio, 1962, 552 p.

9. http://ru.wikipedia.org > Beta-distribution.

10. Bronshtein I. N., Semendyaev K. A. *Spravochnik po matematike* [Handbook of mathematics], Moscow, Nauka, 1962, 608 p.

# Выбор метрики размера проекта в модели оценки трудоемкости разработки программ

Титов А. И.
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
titovvvv@rambler.ru

***Аннотация.*** **В статье рассматриваются различные подходы к оценке трудоемкости разработки программного обеспечения (ПО). Анализируется зависимость оценки трудоемкости разработки ПО от размера проекта. Описываются основные виды существующих метрик и возможность их применения. Предлагаются показатели для сравнения метрик.**

***Ключевые слова:*** **управление проектами, оценка трудоемкости, разработка ПО, метод функциональных точек, метод ucp, uml.**

## Введение

В сфере разработки программного обеспечения управление расписанием проекта приобретает особую важность, поскольку большая часть стоимости разработки складывается, как правило, из непосредственных трудозатрат исполнителей [1]. Также создаваемые системы могут иметь высокую сложность, а исходные требования могут изменяться на разных этапах жизненного цикла [2, 3]. Эти факторы осложняют планирование и повышают риск неуспешного завершения проекта. Чтобы определить, реалистичны ли цели проекта, а также повысить точность планирования, используется оценка трудоемкости разработки программного обеспечения.

Выбор метода оценки должен быть обоснован, в первую очередь, решаемой задачей. Для одних проектов сначала необходимо оценить объем функциональности, а затем, исходя из полученной оценки, определить сроки реализации и объемы работ. В других возможна противоположная ситуация: сначала определяются бюджеты и временны́е рамки, после чего оценивается объем реализуемых функций. Также на выбор метода оценки влияют следующие факторы [4]:

- размер проекта;
- стиль разработки;
- стадия разработки;
- возможная точность.

Сегодня разработано множество методов оценки трудоемкости, некоторые из них универсальны, другие придуманы специально для управления определенным портфелем проектов. Из наиболее распространенных методов оценки можно выделить несколько категорий:

- экспертная оценка;
- регрессионные модели;
- функциональные точки;
- нейронные сети;
- комбинированные методы.

Наиболее перспективны комбинированные методы, поскольку при совмещении подходов может возникнуть принципиально новое решение, позволяющее получить более высокую точность или расширить область применения. Так, при объединении нейросетевого подхода к оценке трудозатрат с регрессионными моделями возможно создание универсальной модели с большим числом показателей, которые могут являться входными параметрами. Поскольку для разных категорий методов оценки существуют принципиально разные способы учета фактора размера проекта, возникает потребность в выборе метрики размера проекта.

Среди прочих факторов, влияющих на оценку, размер проекта является наиболее важным показателем [4]. Хотя оценки размера недостаточно для понимания общей сложности разрабатываемого продукта, существует явная зависимость между размером проекта и его трудоемкостью. На рис. 1 показана зависимость роста объема работ от увеличения размера проекта, рассчитанная по модели COCOMO. В качестве исходных данных для построения зависимости взяты проекты, использованные при разработке модели [5, 6]. На полученном графике можно увидеть зависимость объема работ от размера проекта, причем с ростом размера проекта общая динамика становится более ярко выраженной.


Рис. 1. Зависимость роста объема работ от увеличения размера проекта

Чтобы определить, по каким показателям можно сравнивать метрики размера ПО, необходимо рассмотреть основные группы метрик подробнее.

## Число строк кода

Количество строк кода (Sourcelinesofcode) – это размерно-ориентированная метрика программного обеспечения, в которой объем ПО рассчитывается исходя из количества строк в тексте исходного кода.

Эта методика возникла в 1950-е годы. Основным носителем информации в те времена была перфокарта, причем на одной перфокарте кодировалась одна строка исходного кода. Поскольку каждая строка кода являлась отдельным физическим носителем, можно было подсчитать число этих объектов и определить трудоемкость и производительность труда программистов.

Среди методик учета числа строк есть две основные:

• по числу физических строк (LOC) – определяется как общее число строк исходного кода, включая комментарии и пустые строки;

• по числу логических строк кода (LLOC) – определяется как общее количество команд и зависит от используемого языка программирования. Если язык поддерживает размещение нескольких команд в одной строке, то одна физическая строка должна быть учтена как несколько логических, если она содержит более одной команды языка.

Также имеются производные от основных методик, которые в зависимости от задачи могут содержать дополнительную информацию по следующим показателям:

• число пустых строк;

• число строк, содержащих комментарии;

• процент комментариев (отношение строк кода к строкам комментария, производная метрика стилистики);

• среднее число строк для функций (классов, файлов);

• среднее число строк, содержащих исходный код для функций (классов, файлов);

• среднее число строк для модулей.

Наибольшее распространение эта метрика получила в языках программирования, в которых одна строка кода реализует строго одну команду. В современных высокоуровневых языках одну и ту же функциональность можно описать различным числом строк кода, поэтому для них данная метрика может слабо коррелировать с реальными трудозатратами. Кроме того, при использовании современных средств разработки ПО часто используется генерация кода для определенных действий, что может еще сильнее усложнить итоговую взаимосвязь между числом строк и трудоемкостью.

При этом у измерений в строках кода есть ряд преимуществ. Например, данные по количеству строк в завершенных проектах или модулях программ могут быть легко собраны при помощи служебных средств интегрированных сред разработки (IDE) или специальных программ.

## Метрики Холстеда

Метрики, основанные на анализе числа строк и синтаксических элементов исходного кода программы, были предложены М. Холстедом (MauriceHalstead) в 1977 г. [7]. Метрики Холстеда (Halstead complexity measures) частично позволяют учесть возможность реализации одной и той же функциональности разным числом строк и операторов. Наиболее частым сценарием использования этих метрик является оценка сложности промежуточных продуктов разработки, однако набор метрик также содержит оценки размера.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

• NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

• NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);

• Noprtr (Number of Operators) – общее число операторов в программе;

• Noprnd (Number of Operands) – общее число операндов в программе.

На основе этих характеристик вычисляют различные метрики размера, сложности и качества программ, такие как:

• словарь программы (Halstead Program Vocabulary):

$$HPVoc = NUOprtr + NUOprnd;$$

• длина программы (Halstead Program Length):

$$HPLen = Noprtr + Noprnd;$$

• объем программы (Halstead Program Volume):

$$HPVol = HPLen \cdot \log_2 HPVoc;$$

• сложность программы (Halstead Difficulty):

$$HDiff = \frac{NUOprtr}{2} \cdot \frac{NOprnd}{NUOprnd};$$

• усилия на разработку программы (Halstead Effort):

$$HEff = HDiff \cdot HPVol.$$

Хотя метрики Холстеда позволяют использовать дополнительные возможности по анализу исходного кода, которые отсутствуют в метрике числа строк кода, с точки зрения оценки размера проекта большая часть задач остается нерешенной. Оценка общего числа операторов и их операндов до завершения этапа разработки программы может оказаться еще более сложной задачей, чем оценка числа строк кода.

## Функциональные точки

Анализ функциональных точек (Function points) – это метод измерения размера программного обеспечения с точки зрения пользователей системы [8]. Метод был разработан Аланом Альбрехтом (Alan Albrecht) в середине 1970-х годов, впервые опубликован в 1979 г. Широкое распространение эта методика получила в середине 1980-х годов, после того как была сформирована организация IFPUG, занимающаяся развитием метода, а также публикующая новые версии [9]. На текущий момент актуальна версия метода 4.3.

Данный метод предназначен для оценки объема программного продукта по функциональной модели, т. е. оценивается объем функций разрабатываемой системы. Основная цель этого метода – в декомпозиции системы таким образом, чтобы обеспечить приемлемую сложность анализа. Базовой единицей измерения, на которой основывается данный метод, является функциональная точка. Оценка размера в функциональных точках базируется на количестве и сложности следующих элементов:

• внешних входных элементов – всех элементов, предназначенных для ввода или управления данными, которые поступают в систему;

• внешних выходных элементов – всех элементов для ввода и управления данными, которые выходят за внешние границы системы;

• внешних запросов – комбинации входных и выходных элементов, в которых входной элемент сопоставляется с простой выходной формой;

• внутренних логических файлов – каждого логического файла (группы данных), который создается или используется в системе;

• внешних интерфейсных файлов – каждого файла под управлением другой системы, с которым взаимодействует измеряемая программа.

Общая схема взаимодействия функциональных типов представлена на рис. 2.

Анализ функциональных точек включает в себя следующие этапы [10]:

1) определения типа оценки;

2) определения области оценки и границ продукта;

3) подсчета функциональных точек, связанных с данными;

4) подсчета функциональных точек, связанных с транзакциями;

5) определения суммарного количества не выровненных функциональных точек (UFP);

6) определения значения коэффициента выравнивания (FAV);

7) расчета количества выровненных функциональных точек (AFP).

На двух первых этапах анализируется предмет оценки, определяются границы продукта, выявляется разрабатываемая функциональность. На этапах 3–5 происходят общий подсчет и суммирование функциональных точек без учета коэффициента выравнивания. При выполнении 6-го пункта в методе расчета вводятся общесистемные требования, которые накладывают различные ограничения и увеличивают сложность разработки. Сложность этих требований оцениваются коэффициентом выравнивания (FAV), который зависит от 14 общих системных характеристик (total degree of influence, TDI) и вычисляется по формуле

$$FAV = (0{,}01 \cdot \sum TDI) + 0{,}65.$$

В 7-м пункте оценка подсчитывается в выровненных функциональных точках, причем вариант подсчета зависит от изначального выбора типа оценки. Например, базовая оценка, учитывающая только непосредственно разработку продукта, вычисляется как произведение количества точек и коэффициента выравнивания:

$$AFP = UFP \cdot FAV.$$

При необходимости может быть выполнен еще один этап вычисления – преобразование величины AFP или UFP в число строк кода. Коэффициенты преобразования выбираются в зависимости от языка программирования. Пример коэффициентов преобразования, по данным ряда исследований [4, 11] приведен в табл. 1.

Таблица 1

Коэффициенты преобразования функциональных точек в число тысяч строк кода (SLOC)

| Язык | Число команд в функциональной точке | | |
|---|---|---|---|
| | Минимум | Среднее значение | Максимум |
| C | 60 | 128 | 170 |
| C++ | 40 | 55 | 140 |
| C# | 40 | 55 | 80 |
| Java | 40 | 55 | 80 |
| SQL | 7 | 13 | 15 |
| Макроассемблер | 130 | 213 | 300 |

В целом измерение размера проекта функциональными точками является более актуальной метрикой, чем измерение числом строк кода. Ключевое преимущество этого подхода – оценка основана на требованиях к продукту, что позволяет оценить трудоемкость на самых ранних этапах работы над проектом, сразу после выявления необходимых требований. На последующих этапах работы оценку можно уточнить, поэтому этот метод можно применять при использовании гибких методологий разработки ПО. Также надо отметить, что декомпозиция системы, выполняемая на начальных этапах расчета, может в дальнейшем стать основой для документирования функциональности системы, а значит, снизить трудозатраты проектной команды.

Однако использование функциональных точек в качестве единиц измерения имеет ряд существенных недостатков. Для вычисления функциональных точек необходимо детально изучить спецификацию требований и подсчитать все входные и выходные элементы, файлы, транзакции, что может быть весьма трудоемко. При этом полученная оценка весьма субъективна, поскольку некоторые бизнес-процессы могут иметь высокую алгоритмическую сложность, но обладать достаточно простыми внешними вводами и выводами. По мере развития методологий разработки программного

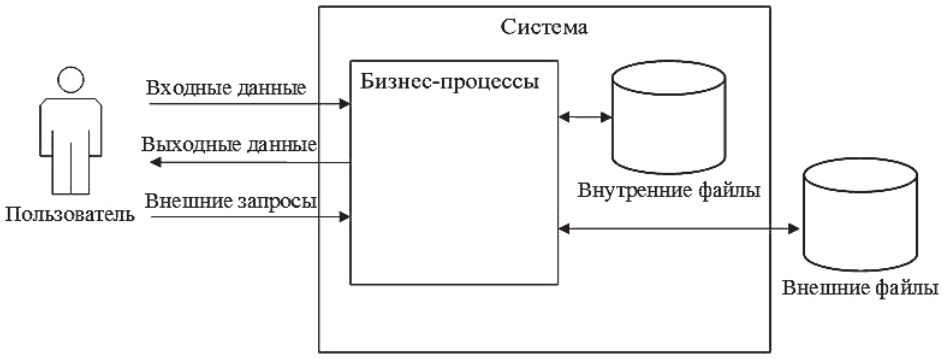



Рис. 2. Общая схема взаимодействия функциональных типов

обеспечения эксперты предложили новые методы оценки, основанные на методе функциональных точек.

## Точки свойств

Метод точек свойств (Feature points) представляет собой расширение метода функциональных точек, которое учитывает не только требования к системе, но и особенности ее реализации. Главное отличие от метода функциональных точек заключается в том, что получаемая оценка корректируется с учетом алгоритмической сложности реализации. К исходным типам элементов добавляется еще один – алгоритмы. В методе точек свойств алгоритмы определяются как набор правил, определяющих какую-либо существенную вычислительную задачу.

Благодаря учету алгоритмической сложности точки свойств лучше адаптированы к оценке систем с высокой сложностью алгоритмов, например [12]:

- системного ПО, операционных систем или компиляторов;
- систем, функционирующих в реальном времени;
- инженерных приложений, таких как системы автоматизированного проектирования (САПР) или математическое ПО;
- систем искусственного интеллекта;
- систем поддержки телекоммуникаций.

## Объектные точки

В современных методологиях разработки объектно-ориентированное программирование (ООП) занимает особое место, фактически является наиболее широко распространенной методологией. Поскольку в изначальном варианте метода функциональных точек не было предусмотрено применения объектно-ориентированного подхода, был разработан адаптированный вариант, оперирующий терминами ООП. Его принципиальным отличием от других вариаций метода функциональных точек является то, что он не расширяет стандартный набор типов элементов, а использует совершенно иные [13]:

- формы (Screens definitions);
- отчеты (User reports);
- модули (3GL Modules).

Фактически в данной метрике каждому уникальному классу или объекту назначается одна объектная точка (Objectpoints). В целом оценка производится примерно по тем же этапам, что и функциональные точки:

1) подсчет количества форм, отчетов и компонентов;
2) классификация каждого экземпляра объекта по уровню сложности;
3) определение веса для объектов;
4) суммирование взвешенных объектов;
5) определение процента повторного использования кода;
6) определение уровня продуктивности;
7) подсчет значения длительности работы в человеко-месяцах.

По сравнению с методом функциональных точек ощутимые различия есть при использовании факторов преобразования на этапах 5–7. Во-первых, подсчитывается процент повторного использования, исходя из него количество объектных точек пересчитывается:

$$NOP = \frac{OP \cdot (100 - r)}{100},$$

где OP – количество объектных точек; $r$ – процент повторного использования кода.

В конце рассчитывается значение длительности работы (PM) в человеко-месяцах:

$$PM = \frac{NOP}{PROD},$$

где PROD – оценка уровня продуктивности, определяемая исходя из опыта и возможностей разработчиков.

Данная метрика удобна для оценки проектов, которые ведутся по объектно-ориентированной методологии и имеют большое число компонентов с визуальной составляющей. Поскольку определение числа точек больше ориентировано на визуальные компоненты, в проектах с высокой алгоритмической сложностью подсчет объектов может затрудниться.

## Метод UCP

Метод UCP (Usecasepoints) представляет собой оценку размера проектов на основе диаграмм UML (Unified Modeling Language) и методологии RUP (Rational Unified Process). Как и многие другие современные методы оценки, UCP базируется примерно на тех же принципах, что и метод функциональных точек. Главное различие заключается в замене единиц измерения с функциональных точек на варианты использования (Use Cases).

Оценка по методу UCP складывается из следующих этапов [14]:

1) *оценка акторов*. На этом шаге определяются все акторы системы (сущности, взаимодействующие с системой извне). После определения для каждого актора в соответствии с его оценкой устанавливается вес (табл. 2);

Таблица 2

Оценки акторов в модели UCP

| Оценка актора | Описание | Вес |
|---|---|---|
| Простой | Внешняя система, взаимодействующая с помощью заранее описанного API (REST, SOAP, DLL) | 1 |
| Средний | Внешняя система, взаимодействующая с помощью более сложного или гибкого API либо с помощью сетевых протоколов (TCP/IP, FTP, HTTP) или СУБД | 2 |
| Сложный | Взаимодействие с пользователем через графический интерфейс | 3 |

2) *нескорректированная оценка вариантов использования*. Рассчитывается исходя из количества транзакций (табл. 3).

Таблица 3

Оценка вариантов использования в модели UCP

| Оценка варианта использования | Количество транзакций | Вес |
|---|---|---|
| Простой | До 3 | 5 |
| Средний | От 4 до 7 | 10 |
| Сложный | Более 7 | 15 |

Существуют также альтернативные критерии для оценки сложности, например, количество классов, реализуемых в рамках одного варианта использования, либо количество объектов в базе данных, изменяемых в рамках одного варианта использования;

3) *оценка технических факторов*. Используется для определения сложности архитектуры приложения и степени влияния нефункциональных требований. Каждый фактор оценивается по шкале от 0 (фактор не значим) до 5 (фактор оказывает существенное влияние), после чего умножается на вес фактора. Описания всех 13 факторов, используемых в модели, приведены в табл. 4;

Таблица 4

Оценка технических факторов

| Фактор | Описание | Вес |
|---|---|---|
| T1 | Распределенность системы | 2,0 |
| T2 | Время отклика/производительность | 1,0 |
| T3 | Увеличение продуктивности пользователя | 1,0 |
| T4 | Сложность внутренней обработки | 1,0 |
| T5 | Повторная используемость кода | 1,0 |
| T6 | Удобство установки | 0,5 |
| T7 | Удобство использования | 0,5 |
| T8 | Переносимость на другие платформы | 2,0 |
| T9 | Поддержка системы | 1,0 |
| T10 | Параллельные вычисления | 1,0 |
| T11 | Функции безопасности | 1,0 |
| T12 | Доступ для третьих лиц | 1,0 |
| T13 | Необходимость обучения пользователя | 1,0 |

4) *оценка внешних факторов*. Используется для определения коэффициента влияния организационных рисков на разработку. Вычисления производятся по аналогии с техническими факторами. Описания внешних факторов приведены в табл. 5;

Таблица 5

Оценка внешних факторов

| Фактор | Описание | Вес |
|---|---|---|
| E1 | Знание предметной области | 1,5 |
| E2 | Опыт разработки приложений | 0,5 |
| E3 | Навык использования ООП | 1,0 |
| E4 | Уровень ведущего аналитика | 0,5 |
| E5 | Мотивация проектной команды | 1,0 |
| E6 | Неизменяемость требований | 2,0 |
| E7 | Частичная занятость сотрудников | –1,0 |
| E8 | Сложность языка разработки | –1,0 |

5) *окончательный подсчет*. Оценивается общее число вариантов с учетом прочих факторов по формуле

$$UCP = (UUCW + UAW) \cdot TCF \cdot ECF,$$

где UUCW – нескорректированная оценка вариантов использования; UAW – оценка акторов; TCF – оценка технических факторов; ECF – оценка внешних факторов.

При использовании метода UCP как метрики размера проекта вычисления на этом заканчиваются, при использовании в качестве модели оценки трудоемкости требуется перевести полученный результат из количества вариантов использования в трудозатраты в человеко-месяцах [15]. В отличие от вычисления количества элементов, перевод оценки в человеко-месяцы может быть довольно трудоемкой и вариативной задачей, поскольку рекомендуемые коэффициенты перевода (20–28 часов на один элемент) не всегда могут соответствовать реальным трудозатратам. Выбор количества часов на один элемент должен зависеть от степени абстракции при создании диаграмм и опыта разработки схожих модулей.

## Выбор показателей для сравнения

После проведения анализа основных метрик размера проекта были определены основные показатели, по которым можно провести сравнение. Поскольку основной целью сравнения является выбор метрики размера проекта, наиболее подходящей для комбинированных методов оценки трудоемкости разработки, предпочтение было отдано более универсальным показателям, которые характеризуют метрики с точки зрения управления жизненным циклом проекта.

1. Возможность оценки на ранних этапах разработки (П1). Оценка размера проекта имеет наибольшую важность именно на ранних этапах разработки, поскольку является базовой информацией для построения календарного графика работ. Оценивание на поздних этапах проекта, например, после завершения этапа разработки и начала этапа внедрения, не дает информации, которая может существенно повлиять на ход завершения проекта. При этом оценка размера на ранних этапах должна иметь возможность постепенного дополнения в процессе сбора и анализа требований к продукту, а также непосредственно при разработке.

2. Возможность оценки бизнес-аналитиком (П2). Приблизительная оценка размера проекта может потребоваться на самых ранних этапах проекта, например, при заключении контракта. Если оценка требуется до детализации требований к продукту и формирования проектной команды разработчиков, то в этой ситуации оценку может выполнить бизнес-аналитик. В этом случае на самом базовом уровне оценивается объем необходимой функциональности и количество связей с внешними системами. Полученная оценка должна быть выполнена таким образом, чтобы ее можно было детализировать на этапах анализа требований и разработки.

3. Наличие инструментов для автоматизированной оценки (П3). В определенных ситуациях оценить размер необходимо не только для планирования, но и для характеристики уже выполненных работ. Поскольку подсчет оценки требует определенных трудозатрат, для его выполнения на поздних этапах проекта следует использовать метрики с возможностью автоматизации анализа и вычислений.

4. Учет алгоритмической сложности (П4). Оценка, основанная исключительно на объеме функциональности, может дать искаженные данные с точки зрения трудоемкости разработки. При реализации нескольких похожих по функ-

циональности компонентов трудозатраты могут значительно различаться в зависимости от алгоритмической сложности этих компонентов.

### Сравнение метрик

Рассмотренные метрики были проанализированы и оценены по выбранным показателям (табл. 6).

Таблица 6

Сравнение метрик размера ПО

| Метрика | П1 | П2 | П3 | П4 |
|---|---|---|---|---|
| SLOC | | | ✓ | |
| Halstead | | | ✓ | ✓ |
| Function Points | ✓ | ✓ | | |
| Feature Points | ✓ | | | ✓ |
| Object Points | ✓ | ✓ | | |
| UseCasePoints | ✓ | ✓ | | ✓ |

Как можно увидеть из результатов сравнения, среди выбранных метрик отсутствуют варианты, которые соответствовали бы всем показателям. Среди метрик, основанных на подсчете строк кода, наиболее удачна реализация метрик Холстеда, поскольку они позволяют оценить не только размер, но и алгоритмическую сложность. Для оценки на ранних этапах разработки следует использовать одну из адаптаций метода функциональных точек. Среди группы этих метрик метод UCP соответствует наибольшему числу показателей. Кроме соответствия выбранным показателем этот метод обладает и другими преимуществами: возможностью оценки в рамках объектно-ориентированного подхода и использованием нотации UML, которая достаточно широко распространена в IT-отрасли.

Чтобы подтвердить возможность использования метода UCP в оценке на ранних этапах разработки ПО, были спроектированы диаграммы UML для системы АСМАДС [16]. На начальных этапах проекта разработки отсутствовала


Рис. 3. Пример диаграммы вариантов использования для одного из компонентов системы АСМАДС

большая часть нефункциональных требований, а также требований к способу реализации, таких как платформа и язык программирования. Исходя из этого, приближенная оценка была получена при помощи диаграмм вариантов использования по первоначальным функциональным требованиям. Пример диаграммы, построенной для одного из компонентов системы, представлен на рис. 3.

При проектировании определены основные классы системы и разработаны диаграммы последовательностей, на основе которых уточнена оценка по диаграммам вариантов использования (рис. 4).

Таким образом, при использовании метода UCP предварительная оценка легла в основу дальнейшего проектирования системы и была уточнена на более поздних этапах разработки.

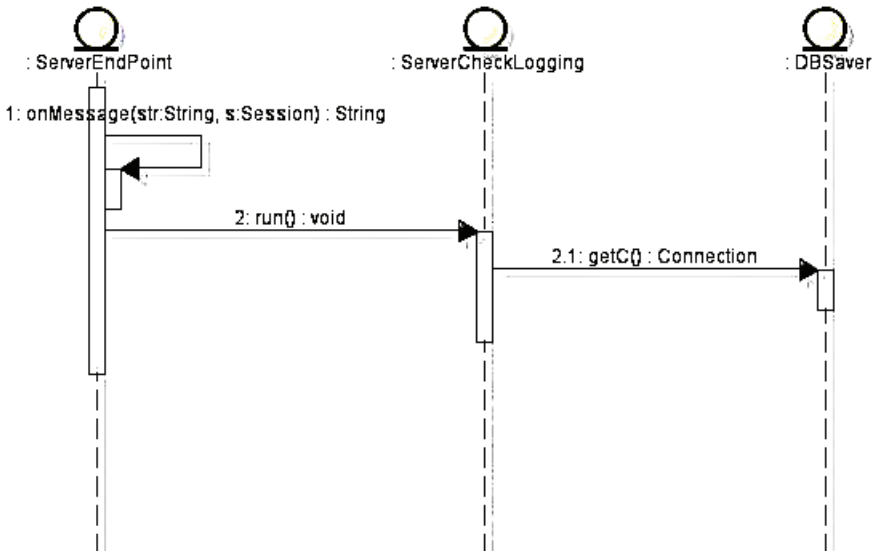



Рис. 4. Пример диаграммы последовательностей для варианта использования «Проверка авторизации»

## Заключение

В условиях непрерывного развития методологий разработки ПО необходимо построить модели оценки проекта, соответствующие современным тенденциям разработки. Поскольку метрика размера проекта является важной частью многих моделей оценки трудоемкости, разработка новых моделей должна начинаться именно с выбора метрики.

По результатам сравнения ключевых метрик предложены показатели, на основе которых можно выбрать метрики в зависимости от требований к разрабатываемой модели проекта. Также по выбранным показателям проведено сравнение для выбора метрики, наиболее подходящей для комбинированных методов оценки трудоемкости. Метод UCP, наиболее соответствующий выбранным показателям, применен для оценки трудоемкости системы АСМАДС, чтобы подтвердить возможность его использования для оценки размера проекта.

Дальнейшие исследования целесообразно продолжить в направлении:

• расширения числа показателей для сравнения метрик;

• разработки новых метрик размера проекта, позволяющих производить оценку на разных итерациях проекта в гибких методологиях разработки;

• улучшения метода UCP для повышения точности оценки при использовании разных типов диаграмм UML.

## Литература

1. Кульдин С. П. Генетический подход к проблеме оценке сроков и трудоемкости разработки программного обеспечения с заданными требованиями к качеству / С. П. Кульдин // Прикладная информатика. – 2010. – № 5 (29). – С. 30–42.

2. A Guide to the Project Management Body of Knowledge (PMBOK Guide). – 5th ed. (Руководство к своду знаний по управлению проектами (Руководство PMBOK. Russian)). 2014. – 589 с.

3. Липаев В. В. Проектирование и производство сложных заказных программных продуктов / В. В. Липаев. – М.: СИНГТЕГ, 2011. – 399 с.

4. Макконелл С. Сколько стоит программный проект / С. Макконелл. – М.: Русская редакция; СПб.: Питер, 2007. – 304 с.

5. Boehm B. Software engineering economics / B. Boehm. – New Jersey: Prentice-Hall, 1981. – 42 p.

6. Boehm B. Software Cost Estimation with Cocomo II. New Jersey, Prentice-Hall. 2000. 544 p.

7. Halstead M. H. Elements of Software Science / M. H. Halstead. – Amsterdam: Elsevier North-Holland, Inc. 1977. – 127 p.

8. Albrecht J. Software function, source lines of codes, and development effort prediction: a software science validation / J. Albrecht, J. E. Gaffney // IEEE Trans Software Eng. SE-9. – 1983. – P. 639–648.

9. IFPUG. Function Point Counting Practices Manual Release 4.0. Int. Function Point Users Group. – Westerville, Ohio, 1994. – 370 p.

10. Архипенков С. Я. Лекции по управлению программными проектами / С. Я. Архипенков. – М., 2009. – 127 с.

11. Тютюнников Н. Н. Оценка размера создаваемого программного средства с использованием функциональных точек / Н. Н. Тютюнников // Перспективы развития информационных технологий. – 2014. – № 18. – URL : http://cyberleninka.ru/article/n/otsenka-razmera-sozdavaemogo-programmnogo-sredstva-s-ispolzovaniem-funktsionalnyh-tochek.

12. Фатрелл Р. Т. Управление программными проектами. Достижение оптимального качества при минимуме затрат / Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И.; пер. с англ. – М.: СПб.: Киев: Вильямс, 2004. – 1136 с.

13. Minkiewicz A. Measuring Object-Oriented Software With Predictive Object Point / A. Minkiewicz // ASM'97 – Appl. Software Meas. – Atlanta, 1997. – P. 225–254.

14. Banerjee G. Use case points – an estimation approach / G. Banerjee. – URL : http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/reporty/use_case_points.pdf.

15. Cohn M. Estimating with Use Case Points / M. Cohn // Methods Tools. – 2005. – Vol. 13, № 3. – P. 3–13.

16. Брынь М. Я. Программный комплекс для мониторинга деформаций особо опасных объектов / М. Я. Брынь, А. Д. Хомоненко, В. П. Бубнов, А. А. Никитчин, С. А. Сергеев, П. А. Новиков, А. И. Титов // Проблемы информационной безопасности. Компьютерные системы. – 2014. – № 1. – С. 36–41.

# Selecting Size of Project Metrics in Software Development Estimation Model

Titov A. I.
Petersburg State Transport University
Saint-Petersburg, Russia
titovvvv@rambler.ru

***Abstract.*** **The article describes various approaches to the effort estimation of software development. It is also considered the dependence of bounds for the complexity of software development on the project size. It describes the main types of existing metrics and their application.The size metrics for comparing are described.**

***Keywords:*** **projectmanagement, software development effort estimation, function points, use case points, uml.**

## References

1. Kul'din S. P. Genetic approach to the problem of estimating the timing and complexity of software development with the specified quality requirements [Geneticheskijpodhod k problem ocenki srokov I trudoemkosti razrabotki programmnogo obespechenija s zadannymi trebovanijami k kachestvu], *Prikladnaja informatika [Appl. Inf.]*, 2010, no. 5 (29), pp. 30-42.

2. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 5th ed. 2014, 589 p.

3. Lipayev V. V. *Proyektirovaniye i proizvodstvo slozhnykh zakaznykh programmnykh produktov* [Design and production of complex custom software], Moscow, SINGTEG, 2011, 399 p.

4. Makkonell S. *Skol'ko stoit programmnyj proekt* [Software Estimation: Demystifying the Black Art], Moscow, Russkaja redakcija; St. Petersburg, Piter, 2007, 304 p.

5. BoehmB. Softwareengineeringeconomics. New Jersey, Prentice-Hall. 1981. 42 p.

6. Boehm B. Software Cost Estimation with Cocomo II, New Jersey, Prentice-Hall, 2000, 544 p.

7. Halstead M. H. Elements of Software Science, Amsterdam: Elsevier North-Holland, Inc., 1977, 127 p.

8. Albrecht J., Gaffney J. E. Software function, source lines of codes, and development effort prediction: a software science validation, *IEEE Trans Software Eng. SE-9*, 1983, pp. 639-648.

9. IFPUG. Function Point Counting Practices Manual Release 4.0. Int.l Function Point Users Group, Westerville, Ohio, 1994, 370 p.

10. Arhipenkov S.Ja. *Lekcii po upravleniju programmnymi proektami* [Software project management lectures], Moscow, 2009, 127 p.

11. Tyutyunnikov N. N. Otsenka razmera sozdavayemogo programmnogo sredstva s ispolzovaniyem funktsionalnykh tochek [Estimate the size to create software tools using functional points], *Perspektivy razvitiya informatsionnykh tekhnologiy [Prospects of development of information technologies]*, 2014, no. 18 (available at: http://cyberleninka.ru/article/n/otsenka-razmera-sozdavaemogo-programmnogo-sredstva-s-ispolzovaniem-funktsionalnyh-tochek).

12. Fatrell R. T., Shafer D. F., Shafer L. I. *Upravlenie programmnymi proektami. Dostizhenie optimal'nogo kachestva pri minimum zatrat* [Quality software project management first edition], Moscow, St. Petersburg, Kiev, Vil'jams, 2004, 1136 p.

13. Minkiewicz A. Measuring Object-Oriented Software With Predictive Object Point, *ASM'97 – Appl. in Software Meas.*, Atlanta, 1997, pp. 225-254.

14. Banerjee G. Use case points – an estimation approach. (available at: http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/reporty/use_case_points.pdf).

15. Cohn M. Estimating with Use Case Points, *Methods Tools*, 2005, Vol. 13, no. 3, pp. 3-13.

16. Bryn M.Ya., Khomonenko A. D., Bubnov V. P., Nikitchin A. A., Sergeyev S. A., Novikov P. A., Titov A. I. Programmnyy kompleks dlya monitoring deformatsiy osoboopasnykh obyektov [Software system for monitoring deformations especially dangerous objects], *Problemy informatsionnoy bezopasnosti. Kompyuternyye sistemy [Problems of information security. Computer systems]*, 2014, no. 1, pp. 36-41.

# Решение интегрального уравнения Винера – Хопфа методом гипердельтной аппроксиимации

Смагин В. А.
СПИИРАН
Россия, Санкт-Петербург
va_smagin@mail.ru

***Аннотация.*** **Представлен метод приближённого решения интегрального уравнения Винера – Хопфа при гладких распределениях вероятностей, составляющих его компонент. Метод основывается на гипердельтной аппроксимации исходных распределений. Использование в ней преобразования Фурье и характеристической функции позволяет работать в методе со случайными величинами, сосредоточенными на всей вещественной оси абсцисс.**

***Ключевые слова:*** **интегральное уравнение Винера – Хопфа, гипердельтная аппроксимация распределений, коррекция, преобразование Фурье, характеристическая функция, начальные моменты, скачок распределения функции ожидания.**

## Введение

К рассматриваемым в настоящей статье задачам по духу и направленности можно отнести небольшое число публикаций. Для примера отметим следующие из них.

В [1] рассматриваются вопросы решения интегрального уравнения Винера – Хопфа на основе использования двух вариантов аппроксимации, одна из которых проводится на основе теоремы дискретизации Шеннона. Указываются границы применимости и приложения к задачам статистики.

В [2] рассмотрен новый класс обобщенных вариационных неравенств и новый класс обобщенных уравнений Винера – Хопфа, включающих многозначные и не расширительные отображения в вещественном Гильбертовом пространстве.

В статье [3] рассматривается применение методов аппроксимации для решения операторных уравнений, содержащих дискретный оператор Винера – Хопфа. Приводятся оценки погрешности и затухающие свойства решений, получаемых с точки зрения некоторых гладких пространств.

В работе [4] рассматриваются некоторые приложения Дельта-функции Дирака к решению задач статистики. При этом некоторые известные результаты обобщаются на случай двух переменных.

В докладе [5] рассматривается применение смесей Дельта-функций Дирака (гипердельтного распределения) для аппроксимации плотностей функций распределений в аналитической форме. Предлагаются субоптимальные решения и процедура вывода, основанная на последовательном разбиении пространства состояний и размещении компонентов с помощью локальной оптимизации.

## Гипердельтная аппроксимация нормального распределения

Известно, что нормальное распределение вероятностей может быть представлено дискретным гипердельтным распределением в аналитическом виде [6]. С учётом трёх начальных моментов это представление имеет вид

$$f(t) \approx C_1\Delta(t-T_1) + C_2\Delta(t-T_2), \qquad (1)$$

где $f(t) = (1/\sqrt{2\pi}\sigma)\exp(-(t-m)^2/2\sigma^2)$ – плотность вероятности аппроксимируемого распределения; $C_1$, $C_2$ – вероятности, $C_1 > 0$, $C_2 > 0$, $C_1 + C_2 = 1$; $\Delta(t)$ – дельта-функция Дирака; $T_1 > 0$, $T_2 > 0$ – положительные постоянные величины. Неизвестные значения $C_i$, $T_i$, $i = 1,2$ вычисляются из четырёх нелинейных уравнений, составленных методом моментов. В результате решения получают

$$f_a(t) = \frac{1}{2}(\Delta(t-m+\sigma) + \Delta(t-m-\sigma)), \qquad (2)$$

где $m$, $\sigma$ – параметры аппроксимируемого распределения.

В статье [7] показано, что если хотя бы один из параметров $T_i$, $i = 1,2$ принимает отрицательное значение, т. е. располагается на отрицательном луче вещественной оси абсцисс, то можно составить аналогичную [6] систему нелинейных уравнений. В ней для определения искомых параметров применяются не обычные производные от дельта-функций, а производные от характеристических функций:

$$\begin{aligned} &C_1\varphi_1(0) + C_2\varphi_2 = 1;\\ &C_1\frac{\varphi_1'(0)}{i} + C_2\frac{\varphi_2'(0)}{i} = \nu_1;\\ &C_1\frac{\varphi_1''(0)}{i^2} + C_2\frac{\varphi_2''(0)}{i^2} = \nu_2;\\ &C_1\frac{\varphi_1'''(0)}{i^3} + C_2\frac{\varphi_2'''(0)}{i^3} = \nu_3. \end{aligned} \qquad (3)$$

В системе уравнений (3) использованы обозначения характеристической функции и её производных, а также мнимая единица $i = \sqrt{-1}$.

Система уравнений (3), в отличие от предшествующей системы, может иметь как положительные, так и отрицательные значения начальных моментов $\nu_j$, $j$ = 1–3. Выражения

для дельта-функций в отличие от $\Delta(t-a)$ могут принимать вид $\Delta(t+a)$, а функция Хевисайда (для вырожденного распределения вероятностей) – вместо $\varphi(t)=e^{ita}$ – $\varphi(t)=e^{-ita}$, где $a$ – некоторая вещественная постоянная.

Это позволяет распространить метод аппроксимации вероятностей не только на положительные значения, но и на положительные и отрицательные значения случайной величины. Предложенная коррекция метода основывается на применении вместо преобразования Лапласа преобразования Фурье и характеристической функции.

## Пример аппроксимации двух нормальных распределений

Пусть заданы две плотности вероятности нормального распределения. Первая характеризует время между заявками, поступающими в одноканальную СМО:

$$a(x)=(1/\sqrt{2\pi}\delta)\exp(-(x-n)^2/2\delta^2) \quad (4)$$

с параметрами $n=20h$, $\delta=5h$. Вторая характеризует время обслуживания:

$$b(x)=(1/\sqrt{2\pi}\sigma)\exp(-(x-m)^2/2\sigma^2) \quad (5)$$

с параметрами $m=10h$, $\sigma=3h$. Требуется представить выражения (4) и (5) в виде гипердельтных распределений. Для распределения (4) будем иметь:

$$a_a(x)\approx\frac{1}{2}(\Delta(x-15)+\Delta(x-25)). \quad (6)$$

Для распределения (5) получим:

$$b_a(x)\approx\frac{1}{2}(\Delta(x-7)+\Delta(x-13)). \quad (7)$$

Выражения (6) и (7) получены из формулы, приведенной в [6]:

$$f(x)\approx\frac{1}{2}(\Delta(x-m+\sigma)+\Delta(x-m-\sigma)). \quad (8)$$

Изображения Лапласа (6) и (7), соответственно, будут равны:

$$a_a^*(s)\approx\frac{1}{2}(e^{-15s}+e^{-25s}); \quad b_a^*(s)=\frac{1}{2}(e^{-7s}+e^{-13s}), \quad (9)$$

где *, $s$ – символ преобразования и переменная Лапласа.

## Характеристическая функция для формул плотностей вероятностей

Характеристическая функция плотности вероятности $f(x)$ определяется выражением

$$\varphi(t)=\int_{-\infty}^{\infty}e^{itx}f(x)dx, \quad (10)$$

в котором $i=\sqrt{-1}$. Если случайная величина сосредоточена на положительной полуоси абсцисс, тогда характеристические функции для (4) и (5) принимают вид

$$\varphi_a(t)=\frac{1}{2}(e^{it(n-\delta)}+e^{it(n+\delta)}); \quad (11)$$

$$\varphi_b=\frac{1}{2}(e^{it(m-\sigma)}+e^{it(m+\sigma)}).$$

Если случайная величина сосредоточена на отрицательной полуоси абсцисс, то характеристические функции для (4) и (5) принимают вид

$$\varphi_a(t)=\frac{1}{2}(e^{-it(n-\delta)}+e^{-it(n+\delta)});$$
$$\varphi_b(t)=\frac{1}{2}(e^{-it(m-\sigma)}+e^{-it(m+\sigma)}). \quad (12)$$

Характеристическая функция для суммы случайных величин, имеющих плотности вероятностей (4) и (5), определяемая плотностью вероятностей

$$c_+(x)=\int_0^t b(x-z)a(z)dz, \quad (13)$$

принимает вид

$$\varphi_+(t)=e^{it(n=m)}\cos(\delta t)\cos(\sigma t), \quad (14)$$

а характеристическая функция для разности случайных величин, определяемая плотностью вероятностей

$$c_-(x)=\int_0^\infty b(x+z)a(z)dz, \quad (15)$$

– вид

$$\varphi_-(t)=e^{-(n-m)it}\cos(\delta t)\cos(\sigma t). \quad (16)$$

## Распределение максимума в теории случайных процессов

Рассмотрим решение задачи о распределении максимума на примере системы массового обслуживания для одноканальной системы $G/G/1$, приведённое в [8]. Основное рекуррентное соотношение для данной СМО представляется в виде

$$w_{n+1}=\max[0,w_n+u_n], \quad (17)$$

где $w_{n+1}$ – время ожидания обслуживания поступающего требования в систему с номером $n+1$; $u_n=x_n-t_n$ – разность между временем обслуживания $n$-го требования и промежутком времени между моментами поступления $n$-го и $n+1$-го требований. Обе случайные величины $s_n$ и $t_n$ независимы от номера $n$ и друг от друга и распределены с плотностями вероятностей $b(t)$ и $a(t)$, а их разность распределена с плотностью вероятности

$$c(t)=\int_0^\infty b(t+u)a(u)du. \quad (18)$$

Определяемая функция распределения времени ожидания требования в системе при условии, что существует её стационарный предел, представляется уравнением

$$W(t)=\begin{cases}\int_{-\infty}^{t}W(t-u)c(u)du, & t\ge 0,\\ 0, & t<0.\end{cases} \quad (19)$$

В книге [8] обсуждались способы преодоления трудностей при изучении системы $G/G/1$ путём построения приближений и границ для точных решений. Среди них – довольно грубое дискретное приближение. Идея этого подхода состояла в изменении входных распределений $A(t)$ и $B(t)$ таким образом, чтобы основное рекуррентное соотношение (17) позволяло получать прямое аналитическое решение для распределения времени ожидания.

Итеративное решение этого уравнения достаточно просто, когда обе входные случайные величины представляются дискретными случайными величинами. В моменты времени $k\tau$ ($k$ = 0, 1, 2, …, с единицей отсчёта времени $\tau$) они принимают только ненулевые значения. Тогда можно записать предел рекуррентной процедуры, получить систему линейных разностных уравнений, которая может быть решена методом $Z$-преобразований, и узнать точное распределение времени ожидания.

Но если случайные величины непрерывны, то возникает задача: как непрерывные случайные величины аппроксимировать дискретными таким образом, чтобы сохранялось существо искомого решения. Вопрос, как выбрать такое приближение, ещё не исследован. Естественная рекомендация состоит в том, чтобы согласовать как можно больше моментов исходных распределений, начиная с первого момента. Исследование точности такого приближения только начинается [3].

В данной статье попытаемся проиллюстрировать этот метод на примере аппроксимации нормальных распределений с точностью до трёх начальных моментов. Используя гипердельтное распределение, можно определить моменты скачков входных распределений. Распределения вероятностей в соответствии с (6) и (7) будут равны:

$$A(t)=\begin{cases}0, & t<n-\delta,\\ \dfrac{1}{2}, & n-\delta\le t<n+\delta,\\ 1, & t\ge n+\delta,\end{cases} \qquad (20)$$

$$B(t)=\begin{cases}0, & t<m-\sigma,\\ \dfrac{1}{2}, & m-\sigma\le t<m+\sigma,\\ 1, & t\ge m+\sigma.\end{cases} \qquad (21)$$

Так как берутся только дискретные случайные величины, можно записать $a(k)=P[t_n=k\tau]$ и $b(k)=P[x_n=k\tau]$. Дискретные функции могут быть представлены и как плотности вероятностей с импульсами в соответствующих точках, как мы показали в первом разделе статьи.

Если пользоваться уравнением (17), то можно найти распределение вероятностей для $u_n$. Определив $c(k)=P[u_n=k\tau]$, так как $u_n=x_n-t_n$, очевидно, что $c(k)$ в общем виде представляется свёрткой [8]:

$$c(k)=a(-k)*b(i)=\sum_{i=-\infty}^{\infty}a(-k+i)\,b(i)\,. \qquad (22)$$

Если $a(k)$и $b(k)$ содержат небольшое число членов, то свёртка легко вычисляется. Но остаётся нерешённым вопрос, как выбрать единицу отсчёта времени $\tau$.

Пример 1. Представим плотности исходных распределений (4) и (5), используя формулу (8) в виде дискретных распределений. Эти распределения применительно к свёртке (22) показаны на рис. 1. Распределение $a(-k)$ показано на отрицательной полуоси. Для уменьшения размера рисунка масштаб по оси абсцисс изменён, численные значения интервалов между дельта-функциями, указанные в (6) и (7), не соответствуют реальным значениям.

Для выполнения рекуррентной процедуры в уравнении (17) предположим, что начальная величина $w_0=0$. Кроме того, определим вероятность $p_n=P\,[w_n=k\tau]$. Теперь можно применить рекурсию, состоящую в выполнении операций, описанных уравнением

$$w_{n+1}(y)=\pi(w_n(y)\cdot c(y)), \qquad (23)$$

в котором $\pi$-оператор заменяет плотность вероятности своего аргумента путём замены всей вероятности, связанной с отрицательными значениями, на импульс в точке $y=0$, площадь которого равна этой вероятности. Предельное решение следует из уравнения

$$w(y)=\pi(w(y)\cdot c(y))\,. \qquad (24)$$

Оно даёт стационарную плотность вероятности времени ожидания в системе. Эта плотность должна быть такой, что когда она образует свёртку с $c(y)$ и результирующая плотность переносит свою вероятность с отрицательной оси в импульс, расположенный в нуле, результирующая плотность имеет такой же вид, как и $w(y)$, с которой начиналось рассмотрение.

Для численного расчёта примем следующие значения параметров плотностей: для $a(x)$ $n=11h$, $\delta=3h$, а для $b(x)$

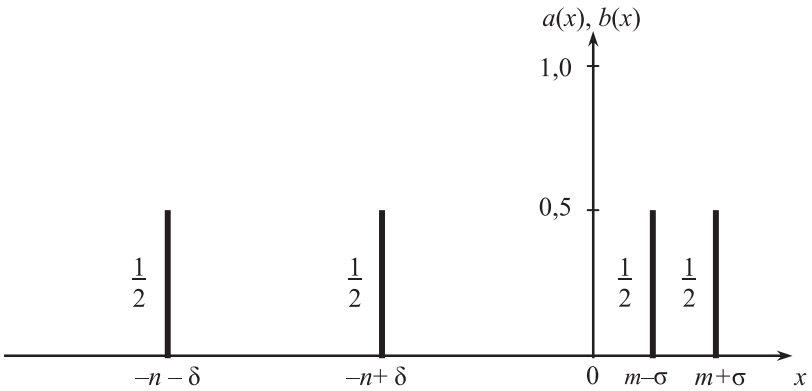



Рис. 1. Дискретное представление исходных распределений

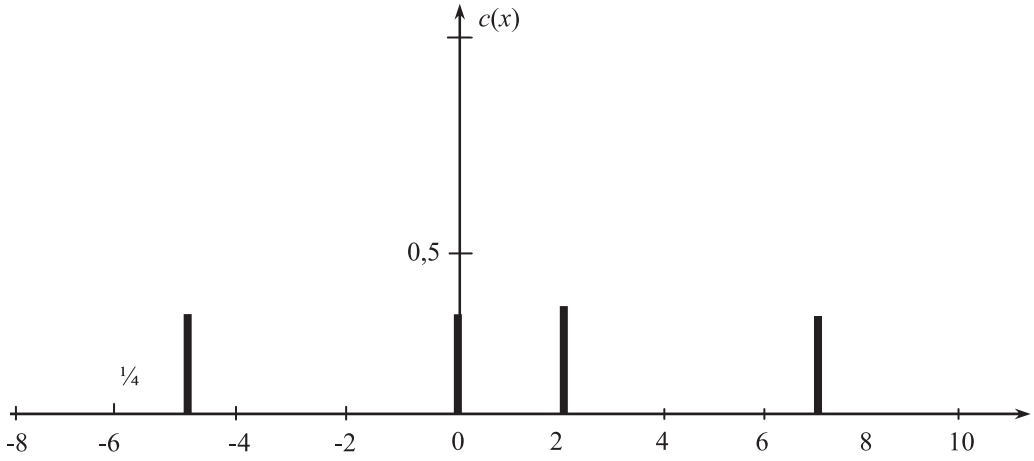



Рис. 2. Графическое изображение дискретной свёртки

$m = 10$, $\sigma = 3h$. В этом случае выражение для свёртки (22) принимает вид

$$c(x) = \frac{1}{4}(\Delta(x) + \Delta(x-2) + \Delta(x+5) + \Delta(x-7)). \quad (25)$$

График этого выражения показан на рис. 2.

Далее поступая, как описано в начале примера, и полагая $k = x$, можно записать выражение для определения величины вероятности на $k$-м шаге:

$$p(k) = \frac{1}{4}p(k+5) + \frac{1}{4}p(k) + \frac{1}{4}p(k-2) + \frac{1}{2}p(k-7), \quad k = 1,2,3,... \quad (26)$$

Граничное уравнение для $p(0)$ равно:

$$p(0) = \frac{1}{4}p(0) + \frac{1}{4}p(5) + \frac{1}{4}p(-2) + \frac{1}{4}p(-7), \quad k = 0. \quad (27)$$

Теперь имеем знакомую задачу решения системы линейных разностных уравнений. К такому же результату приводит метод факторизации Винера. Достоинство описанного здесь метода состоит в том, что он использует в явном виде дискретную природу случайных величин. В обоих случаях трудная часть решения состоит в нахождении корней многочлена (корни знаменателя $P(z) = \sum_k p(k) z^k$, тогда как в методе факторизации спектра корни находят из выражения $A^*(-s)B^*(s) - 1$ [8].

В данном разделе статьи показано, как превратить непрерывные задачи в дискретные, для которых применимы довольно простые методы. Это проиллюстрировано решением задачи при обоих нормальных распределениях случайных величин между требованиями их обслуживания. Однако вопросы выбора шага дискретизации и адекватности приближения требуют дальнейшего исследования. Здесь мы показали, как переходить от непрерывных распределений к дискретным на основе гипердельтного распределения. Для выполнения численного расчёта распределения времени ожидания в системе при любых непрерывных распределениях мы рекомендуем процедуру со свёрткой, детально изложенную в [8].

## Об определении распределения максимума в непрерывном виде

Решение уравнения (17) на основе гипердельтной аппроксимации возможно не только в дискретном, но и в непрерывном виде. Принцип такого подхода заключается в том, чтобы использовать для решения уравнения (17) не плотности вероятностей исходных распределений $a(x)$ и $b(x)$, а их характеристические функции, а более конкретно – характеристическую функцию их разностной свёртки (15).

Представляя начальную и все последовательные вероятности состояний $P_n$ СМО также характеристическими функциями, производя их последовательное свёртывание с характеристической функцией разности исходных случайных величин и сдвигая на каждом шаге отрицательную вероятность в начало координат, получим на положительном луче оси абсцисс характеристическую функцию случайной величины времени ожидания.

Иначе говоря, операция решения (17) на основе гипердельтной дискретной аппроксимации заменяется операцией решения этого уравнения на основе действий с характеристическими функциями тех же исходных случайных величин, указанных в разделе «Пример аппроксимации двух нормальных распределений». Для рассмотренных в нём двух нормальных распределений выражение для характеристической функции свёртки плотности разности случайных величин будет равно

$$\varphi(t) = e^{-(n-m)t}\cos(\sigma t)\cos(\delta t). \quad (28)$$

Характеристическая функция начальной единичной вероятности состояния СМО будет принимать вид

$$P(0) = 1. \quad (29)$$

Далее следует применять алгоритм вычисления, аналогичный дискретному алгоритму, рассмотренному в [8].

Шаг 1. Произвести свёртку характеристических функций (28) и (29).

Шаг 2. По полученной свёртке вычислить вероятность попадания случайной величины на отрицательную часть оси абсцисс. Полученную вероятность сосредоточить в нуле оси абсцисс.

Шаг 3. Получить составную плотность вероятности на положительной полуоси абсцисс и найти для неё характеристическую функцию.

Шаг 4. Умножить полученную характеристическую функцию на характеристическую функцию (28).

Далее шаги 2–4 необходимо последовательно повторять до тех пор, пока плотность вероятности на положительной полуоси абсцисс не будет изменяться, т. е. совпадать с предшествующей на ней плотностью вероятности.

После этого следует использовать полученную плотность вероятности для нахождения стационарной функции распределения времени ожидания обслуживания поступающего требования в СМО.

Вместо исходных нормальных распределений могут использоваться любые распределения, плотности вероятностей которых позволяют на основе гиперэкспоненциальной аппроксимации получать необходимые характеристические функции. Свёртки сводятся к перемножению характеристических функций.

На элементарном примере покажем, как вычислить значения распределения времени ожидания в начале первого и второго циклов – характерные разрывные значения, присущие началу распределения времени ожидания.

Положим, что заданы значения параметров указанных нормальных распределений: $n = 11h$, $\delta = 2h$, $m = 10h$, $\sigma = 3h$. Тогда скачок распределения времени ожидания в начале первого цикла определится как $1-\rho_0 = \int_{-\infty}^{0} c(z)dz$ и численно будет равен $1 - \rho_0 = 0{,}609$. Соответствующая характеристическая функция равна $\varphi(t) = e^{-(n-m)it}\cos(\sigma t)(\cos\delta t)$. Вычислим начальные моменты $c(x)$: $\nu_1 = -1h$, $\nu_2 = 14h$, $\nu_3 = -40h$ и, решив систему уравнений гипердельтной аппроксимации, найдём значения искомых параметров $C_1 = 0{,}5$, $C_2 = 0{,}5$, $T_1 = -4{,}606h$, $T_2 = 2{,}606h$. Составим по ней аппроксимационную функцию $F(x) = 0{,}5\,\Phi(x-2{,}606) + 0{,}5\,\Phi(x+4{,}606)$, а затем найдём условную функцию распределения для положительной полуоси $F_1(x,\tau) = \dfrac{F(x+\tau)-F(\tau)}{1-F(\tau)}$. Обе функции показаны на рис. 3.

Величина $\tau = -5$. Легко записать выражение для условной плотности вероятности, расположенной на положительной полуоси:

$$c_1(x) = 0{,}5\,\Delta(x) + 0{,}5\,\Delta(x-7{,}606). \qquad (30)$$

Соответствующая (30) характеристическая функция имеет вид

$$\varphi_1(t) = 0{,}5(1+e^{7{,}606it}). \qquad (31)$$

Умножая характеристические функции $\varphi(t)$ и $\varphi_1(t)$, получим общую характеристическую функцию

$$\varphi\varphi(t) = 0{,}5\cos(2t)\cos(3t)e^{-it}(1+e^{7{,}606t}). \qquad (32)$$

Используя (32), найдём значения начальных моментов и параметров новой гипердельтной аппроксимации: $\nu_1 = 2{,}802h$; $\nu_2 = 35{,}320h^2$; $\nu_3 = 252{,}96h^3$; $C_1 = 0{,}5$; $C_2 = 0{,}5$; $T_1 = -2{,}438h$; $T_2 = 8{,}044h$. Запишем выражения для плотности вероятности и функции распределения:

$$c(x) = \frac{1}{2}(\Delta(x+2{,}438) + \Delta(x-8{,}044));$$

$$C(x) = \frac{1}{2}(\Phi(x+2{,}438) + \Phi(x-8{,}044)). \qquad (33)$$

Далее составим условную функцию распределения для положительной полуоси абсцисс и вычислим её значения при $\theta = -2{,}438h$:

$$C_1(x,\theta) = \frac{C(x+\theta)-C(\theta)}{1-C(\theta)}. \qquad (34)$$

Графики обеих функций представлены на рис. 4.

Начальный скачок функции распределения времени ожидания составит $1 - \rho_1 = C_1(0{,}1;0) = 0{,}333$.

Подобным образом согласно указанному пошаговому алгоритму могут быть определены следующие итеративные значения скачков распределения времени ожидания. Более того, при проведении дополнительного исследования можно составить алгоритм вычисления стационарного значения величины скачка функции распределения времени ожидания СМО.

Поведение функции за скачком будет определяться дискретной аппроксимацией функции распределения. В данной статье исследование выполнено только с точностью до трёх начальных моментов. Для получения непрерывной функции распределения времени ожидания следует слева направо соединить середины скачков непрерывной плавной линией.

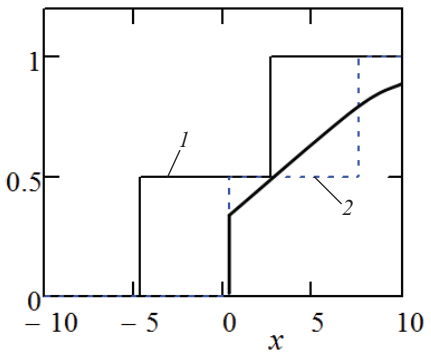



Рис. 3. Аппроксимационная (1) и условная (2) функции распределения

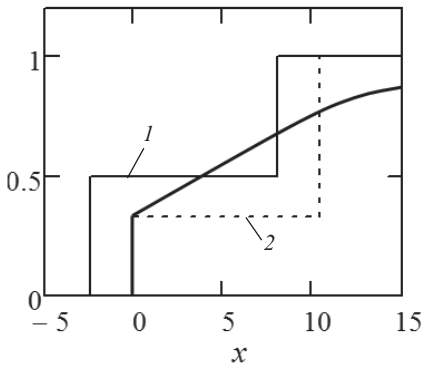



Рис. 4. Функция распределения (1) и условная функция распределения (2)

## Заключение

Решению интегрального уравнения Винера – Хопфа распределения максимума случайного блуждания, когда координата процесса может принимать как положительное, так и отрицательное значение, посвящено большое число исследований. Однако его решение представляет определенные трудности, когда оба исходных распределения непрерывны и нетривиальны.

Основываясь на идее дискретного решения уравнения, предложенного в [8], в настоящей статье развивается дискретный метод решения уравнения именно для непрерывных исходных распределений. В его основе лежит предложенный автором метод гипердельтной аппроксимации произвольных распределений [6]. Но для того, чтобы этот метод не только был пригоден для положительных значений случайной величины, но и был распространён на всю вещественную ось, его скорректировали, применив в нём преобразования Фурье и характеристическую функцию. Это позволило получить итеративное численное решение уравнения Винера – Хопфа при трёх начальных моментах обеих нормальных исходных распределений. Для иллюстрации решения приведены частные примеры, подтверждающие работоспособность метода.

Метод может применяться для произвольных гладких распределений вероятностей. Количество начальных моментов, привлекаемых в методе, можно увеличить. Это приведёт к повышению точности решения, однако численный процесс решения усложнится. Хотя с практической точки зрения иногда ограничение может быть оправдано и малым числом моментов.

Применённый аппарат использования преобразования Фурье и характеристической функции, на наш взгляд, в дальнейшем можно усовершенствовать с целью получения решения уравнения в замкнутом аналитическом виде.

Источник [9] следует рассматривать как дополнительный для ознакомления с предшествующим исследованием авторов в данной области.

## Литература

1. Payandeh Najafabadi A. T. An Approximate to Solution of a Subclass of Wiener-Hopf Integral Equation / A. T. Payandeh Najafabadi, D. Z. Kucerovsky // Proc. World Congr. Eng. “WCE 2009”, 1–3 July, 2009, London, U. K. – Vol. II. – URL : https://www.researchgate.net/publication/44260185_An_Approximate_To_Solution_Of_A_Subclass_Of_Wiener-Hopf_Integral_Equation.

2. Mansotral P. Wiener-Hopf Equation Technique for Generalized Variational Inequalities and Nonexpansive Mappings / P. Mansotral, B. S. Komal // Appl. Math. Sci. – 2012. – Vol. 6, № 18. – P. 869–878.

3. Nowak M. A. Approximation methods for a class of discrete Wiener-Hopf equations / M. A. Nowak // Opuscula Math. – 2009. – Vol. 29, № 3. – P. 271–288.

4. Chakraborty S. Some Applications of Dirac’s Delta Function in Statistics for More Than One Random Variable / S. Chakraborty // Appl. Appl. Math. – 2008. – Vol. 3, Is. 1. – P. 42–54.

5. Klumpp V. Dirac Mixture Trees for Fast Suboptimal Multi-Dimensional Density Approximation / V. Klumpp, U. D. Hanebeck // Proc. 2008 IEEE Int. Conf. Multisensor Fusion and Integration for Intelligent Systems “MFI 2008”, Seoul, Republic of Korea, Aug. 2008.

6. Смагин В. А. О моделировании случайных процессов на основе гипердельтного распределения / В. А. Смагин, Г. В. Филимонихин // Автоматика и вычислительная техника. – 1990. – № 1. – С. 25–31.

7. Смагин В. А. Коррекция гипердельтного распределения в теории случайных процессов / В. А. Смагин // Информация и космос. – 2015. – № 4. – С. 60–64.

8. Клейнрок Л. Вычислительные системы с очередями : пер. с англ. / Л. Клейнрок. – М. : Мир, 1979. – 600 с.

9. Смагин В. А. Аппроксимационный метод расчёта разомкнутых сетей массового обслуживания / В. А. Смагин, Г. В. Филимонихин // Автоматика и вычислительная техника. – 1986. – № 4. – С. 28–33.

# The Decision of the Integrated Equation of Wiener – Hopf by Method of Hyper-Delta Approximation

Smagin V.A.
SPIIRAS
Russia, St. Petersburg
va_smagin@mail.ru

***Abstract.*** **The method of the approached decision of the integrated equation of Wiener – Hopf is presented at smooth distributions of probabilities making its component. The method is based on giperdelta approximations of initial distributions. Use in it of transformation of Fourier and characteristic function allows to work in a method with the random variables concentrated to all material axis of abscises.**

***Keywords:*** **Wiener – Hopf integrated equation, giperdelta approximation of distributions, correction, transformation of Fourier, characteristic function, the initial moments, jump of distribution of function of expectation.**

## References

1. Payandeh Najafabadi A.T., Kucerovsky D.Z. An Approximate to Solution of a Subclass of Wiener-Hopf Integral Equation. *Proc. World Congr. Eng. "WCE 2009"*, 1–3 July, 2009, London, U.K., Vol. II. Available at: https://www.researchgate.net/publication/44260185_An_Approximate_To_Solution_Of_A_Subclass_Of_Wiener-Hopf_Integral_Equation.

2. Mansotra1 P., KomalB.S. Wiener-Hopf Equation Technique for Generalized Variational Inequalities and Nonexpansive Mappings, *Appl. Math. Sci.*, 2012, Vol. 6, no. 18, pp. 869-878.

3. Nowak M.A. Approximation methods for a class of discrete Wiener-Hopf equations, *Opuscula Math.*, 2009, Vol. 29, no. 3, pp. 271-288.

4. Chakraborty S. Some Applications of Dirac's Delta Function in Statistics for More Than One Random Variable, *Appl. Appl. Math.*, 2008, Vol. 3, Is. 1, pp. 42-54.

5. Klumpp V., Hanebeck U.D. Dirac Mixture Trees for Fast Suboptimal Multi-Dimensional Density Approximation // Proc. 2008 IEEE Int. Conf. Multisensor Fusion and Integration for Intelligent Systems "MFI 2008", Seoul, Republic of Korea, Aug. 2008.

6. Smagin V.A., Filimonihin G.V. About modeling random processes on the basis of the limit distribution [O modelirovanii sluchajnyh processov na osnove giperdeltnogo raspredeleniya], *Avtomatika i vychislitelnaya tekhnika [Automation and Computer Engineering]*, 1990, no. 1, pp. 25-31.

7. Smagin V.A. Correction giperdeltnogo distribution in the theory of random processes [Korrekciya giperdeltnogo raspredeleniya v teorii sluchajnyh processov], *Informaciya i kosmos [Information and Space]*, 2015, no. 4, pp. 60-64.

8. Klejnrok L. *Vychislitelnye sistemy s ocheredyami* [Computer systems with queues], Moscow, Mir, 1979, 600 p.

9. Smagin V.A., Filimonihin G.V. The approximation method for calculation of open queuing networks [Approksimacionnyj metod raschyota razomknutyh setej massovogo obsluzhivaniya], *Avtomatika i vychislitelnaya tekhnika [Automation and Computer Engineering]*, 1986, no. 4, pp. 28-33.