# 7 МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ

УДК 004.005

## МИНИМИЗАЦИЯ РИСКОВ ПОТЕРИ ДОСТУПНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОСЛЕ УСТАНОВКИ ОБНОВЛЕНИЙ ИЛИ ИЗМЕНЕНИЯ ФУНКЦИОНАЛЬНОСТИ

**С.А. Арустамов, М.Г. Генин**

Дается описание стратегии обновления информационных систем на основе понятия простого обновления. Проводится сравнительный анализ процесса обновления при использовании стратегии простых обновлений и без использования данной стратегии. На основе этого доказывается обоснованность такого способа обновления для снижения риска сбоя системы после проведенного обновления. Приводится пример, подтверждающий эффективность данного подхода.

**Ключевые слова:** информационные системы, обновление, управление обновлениями, стратегия обновления, простое обновление, вероятность успеха, риск сбоя.

### Введение

Вопросами управления обновлениями (или изменениями) информационных систем занимаются достаточно давно, им посвящено много публикаций. Одна из первых работ датируется 1980 г. и принадлежит фирме IBM [1]. В ней впервые управление изменениями рассматривается не как отдельная дисциплина, а как составная часть управления информационными системами. С тех пор эта область знаний сильно расширилась, были предложены различные подходы к управлению изменениями, однако этот основополагающий принцип в силе и по сегодняшний день. В настоящее время процесс управления изменениями обычно связывают с методологией ITIL [2]. Наряду с ITIL, существуют и другие методологии, посвященные управлению изменениями в информационных системах, например, COBIT [3], Microsoft Operational Framework [4]. В 2005 г. был разработан международный стандарт ISO/EIC 20000 [5], который объединил все эти методологии в одну общую концепцию управления процессами в информационные технологии (ИТ), в том числе и управления изменениями.

Тем не менее, несмотря на достаточно большое количество разработанных подходов и методик, все они являются в значительной степени организационными и носят формальный характер. В них предлагаются процедуры планирования, утверждения, реализации и контроля за выполнением изменений в системе, обсуждаются вопросы организации взаимодействия руководства, бизнес-подразделений и подразделений ИТ при выполнении тех или иных действий, распределяются роли и зоны ответственности сотрудников. Но все эти процедуры являются внешними по отношению собственно к процессу обновления, они не дают ответа на вопрос о том, как нужно проводить обновление систем наиболее безопасным образом, чтобы риск возникновения внештатных ситуаций при этом был минимальным.

В настоящей статье обсуждаются риски, связанные с обновлением систем, а также предлагаются некоторые способы снижения этих рисков.

Рассматриваются ситуации, при которых установка обновления носит императивный (обязательный) характер. При этом риски отказа от установки как вероятностная категория отсутствуют, а последствиями отказа являются детерминированные ущербы (отзыв лицензии на ведение бизнеса, прекращение поддержки предыдущей версии, нарушение вновь вводимой корпоративной политики, штрафы, накладываемые надзорными органами, или другие негативные последствия).

### Снижение риска сбоя системы при использовании стратегии простых обновлений

Под большой системой будем понимать систему, состоящую из большого числа модулей и взаимосвязей между ними. Обновление таких систем представляет собой нетривиальную задачу и связано со следующими проблемами.

- Высокий риск возникновения неполадок после установки обновления в продукцию даже при условии выполнения всех требований по тестированию и процедуре миграции. Это связано с тем, что во многих случаях бывает слишком сложно или практически невозможно полностью смоделировать работу продукционной системы вместе с ее окружением во время тестирования. Например, крайне сложно смоделировать подключение тысяч пользователей или взаимодействие с внешними поставщиками услуг при параллельной работе продукционной системы.
- Невозможность возврата к предыдущей версии без потерь времени и (или) данных, т.е. возврат либо требует очень много времени, либо выполняется сравнительно быстро, но с потерей данных. Проблемы могут выявляться не только в момент запуска системы после обновления, но и в течение некоторого времени после начала работы, когда данные уже изменились по сравнению с теми, которые были в момент запуска. При принятии решения о возврате к предыдущей версии системы сохранение новых данных может оказаться невозможным.

Предлагаемый подход к обновлению систем основан на том предположении, что риск возникновения проблем есть всегда и он в принципе не может быть устранен. Представляется возможным лишь снизить последствия возникновения рисковых ситуаций. Для этого, в свою очередь, нужно определиться с тем, какие рисковые ситуации для нас приемлемы, а какие – нет. Применительно к обновлениям программного обеспечения это можно переформулировать более строго следующим образом: нужно определиться с максимально допустимым временем либо для исправления ошибки, которая может возникнуть после установки того или иного обновления, либо для возврата к предыдущему состоянию системы, т.е. до обновления. Затем при установке обновлений нужно придерживаться этих допустимых параметров, другими словами, единовременно устанавливать только такие обновления, которые возможно откатить или исправить за допустимое время без потерь данных.

Для этого нужно определиться с максимально допустимым временем возврата, а затем попытаться разбить планируемое обновление на несколько таких простых обновлений, откат каждого из которых в отдельности возможен за допустимый интервал времени. Устанавливать каждое такое простое обновление нужно отдельно, в определенной последовательности.

После всего сказанного, можно сформулировать следующие условия, которым должно удовлетворять каждое простое обновление в составе большого обновления:

- в определенный момент времени устанавливается только одно простое обновление;
- после успешной установки каждого «простого» обновления система работоспособна;
- в случае возникновения проблем после установки простого обновления возврат к предыдущей конфигурации происходит за допустимый интервал времени и либо не приводит к потере данных вообще, либо приводит лишь к минимальной потере данных, связанных с новым функционалом.

Покажем, что последовательная установка полного обновления, используя принцип его разбиения на простые обновления, удовлетворяющие вышеприведенным условиям, более безопасна, чем установка всего обновления сразу.

**Определение.** Будем называть обновление $u_i$ *простым*, если возможен процесс возврата за допустимое время к предыдущему состоянию системы по прошествии некоторого времени после установки обновления $u_i$. Если $u_i$ прошло успешно, то система работоспособна и готова к установке обновления $u_{i+1}$.

Будем исходить из того, что любое полное обновление $U$ может быть представлено в виде суммы некоторого количества простых обновлений $u_i$, $i$=1,..., $N$:

$U = u_1 + u_2 + ... + u_N$.

**Утверждение.** Последовательная установка простых обновлений $u_i$, $i$=1,..., $N$ в составе обновления $U$ безопаснее, чем установка всего обновления $U$ сразу.

**Доказательство.** Исходим из того, что после установки любого $u_i$ система работоспособна, поэтому есть выбор: либо выделить время и установить все обновление $U$ целиком, либо поэтапно устанавливать каждое $u_i$, после чего делать паузу и давать возможность системе поработать некоторое время, затем устанавливать $u_{i+1}$ и т.д.

Обозначим $p(u_i)$ – вероятность успешной установки обновления $u_i$. Рассчитаем теперь вероятность успешной установки $P(U)$ всего полного обновления в том и в другом случае.

Рассмотрим случай установки всего обновления сразу. Если все обновление $U$ состоит из одного простого обновления $u_1$**,** можем записать:

$P(U)=p(u_1)$.

Для обновления, состоящего уже из двух простых обновлений $u_1$ и $u_2$, формула будет выглядеть как произведение вероятностей

$P(U)=p(u_1)\cdot p(u_2)$,

так как успех всего обновления возможен только при успехе обновлений $u_1$ и $u_2$ .

Соответственно, для $N$ обновлений формула будет выглядеть следующим образом:

$$P(U) = p(u_1) \cdot p(u_2) \cdot ... \cdot p(u_N) = \prod p(u_i),\ i=1,\ldots,N. \qquad (1)$$

Поскольку любая $p(u_i)$ меньше единицы, то можно утверждать, что $P(U) < p(u_i)$ для любого $i$. Отсюда вытекает, что при одновременной установке всех простых обновлений $u_i$ в составе обновления $U$ вероятность успешного исхода $P(U)$ всегда ниже, чем при единовременной установке только одного обновления $u_i$.

Рассмотрим теперь ситуацию, когда обновления $u_i$ в составе обновления $U$ устанавливаются не одновременно, а через определенные промежутки времени, между которыми система работает (можно так сделать, так как по требованиям после установки каждого $u_i$ система работоспособна). Для этого заметим, что установка обновления $u_i$ производится только после того, как обновление $u_{i-1}$ уже было успешно установлено, и после этого система некоторое время проработала.

Посмотрим, как ведет себя вероятность $p(u_i)$ при таком способе обновления. Вообще говоря, пока что авторы не касались вопроса о том, как определить вероятность $p(u_i)$. Вопрос этот очень сложный, и пока не хотелось бы подробно его рассматривать. Однако можно сделать некоторые предположения о том, от чего она зависит и как на нее повлиять. Например, можно утверждать, что в первые моменты после запуска обновленной системы вероятность сбоя выше, чем в последующие. Как правило, большинст-

во проблем после обновления возникает в течение первого сеанса работы. Если первый сеанс проблем не выявил, то вероятность того, что они возникнут позднее, существенно ниже. Таким образом, можно утверждать, что после установки обновления $u_i$ с течением времени вероятность его успеха $p(u_i)$ будет возрастать и в предельном случае стремиться к единице. На рис. 1 вероятность $p(u_i)$ представлена графически как функция времени.

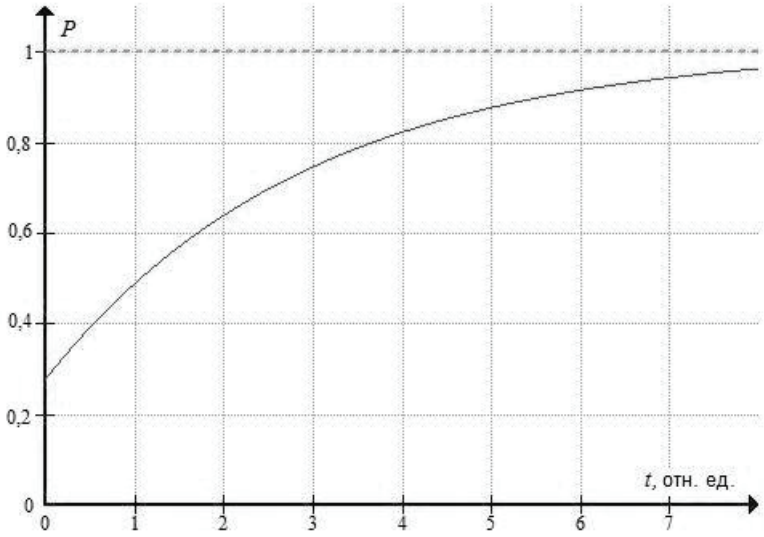


Рис. 1. Зависимость вероятности успеха обновления от времени

Здесь момент времени, равный нулю, соответствует ситуации запуска системы сразу после обновления $u_i$, а момент времени $T$ – состоянию, когда система уже проработала некоторое время после обновления. Чем больше система проработала после установки $i$-го обновления, тем меньше шансов, что из-за него возникнет сбой. Если между каждым обновлением $u_1, u_2, ..., u_{i-1}$ проходит достаточное количество времени, то вероятности успеха этих уже установленных обновлений $p(u_1),\ldots, p(u_{i-1})$ близки к единице (рис. 2).

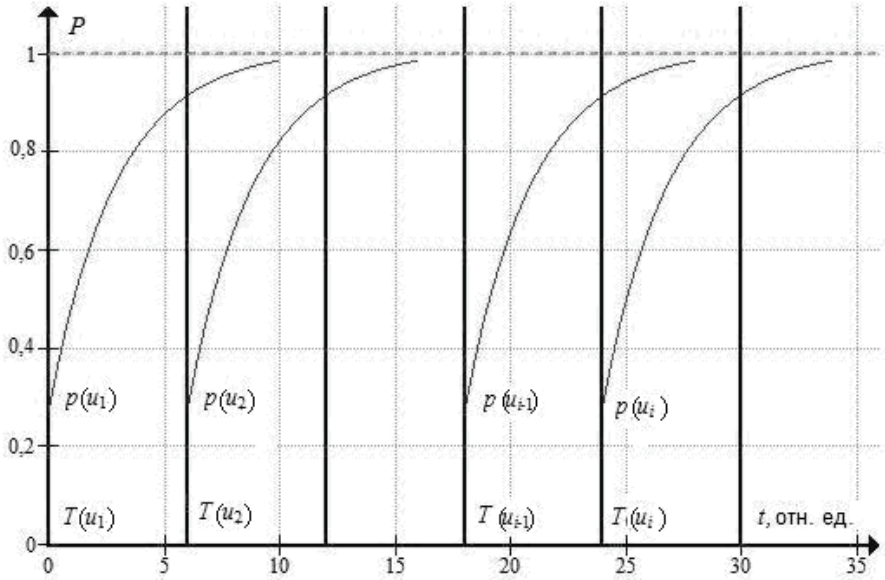


Рис. 2. Вероятности успеха для уже установленных обновлений

Обозначим $P(u_i)$ – вероятность успеха всех обновлений системы с первого по $i$-е при последовательной их установке. По аналогии с (1) можем записать:

$P(u_i) = p(u_1) \cdot p(u_2) \cdot \ldots \cdot p(u_i) = \prod p(u_j),\ j=1, \ldots, i.$

Учитывая, что $p(u_1), \ldots, p(u_{i-1})$ в случае последовательной установки близки к единице при установке $i$-го обновления, можем приближенно записать, что

$P(u_i) \approx p(u_i).$

Таким образом, можем принять, что при последовательной установке обновлений $u_1, u_2, \ldots, u_i$ вероятность успеха всех этих обновлений равна вероятности успеха последнего, $i$-го обновления. Действительно, все предыдущие обновления уже установлены ранее и после каждого обновления система уже проработала некоторое время.

Напомним, что при одновременной установке всех обновлений вероятность $p(u_i)$ успеха каждого из них от первого до последнего не стремится к единице, а имеет начальное значение для момента времени ноль, а, следовательно, вносит свой негативный вклад в общую вероятность $P(U)$. Как следует из (1), вероятность $P(U)$ успеха всех обновлений $u_i$, установленных разом, существенно ниже вероятности $P(u_i)$ успеха при последовательной установке обновлений с $u_1$ по $u_i$, и так вплоть до последнего $u_N$. Таким образом, можно записать, что

$$P(U) < P(u_i)$$

для любого $i$ от единицы до $N$. Следовательно, при выполнении требования о работоспособности системы при установке любого из простых обновлений в составе полного обновления вероятность успешного исхода выше именно при последовательной, растянутой во времени установке обновлений. При одновременной установке сразу всех простых обновлений риск сбоя существенно возрастает.

**Ограничения по времени обновления и их влияние на риск сбоя**

В приведенных выше рассуждениях авторы не касались того, до какой степени нужно разбивать полное обновление на простые обновления, его составляющие. Исходя из сделанных выводов, получается, что чем на более мелкие части разбито обновление, тем более безопасна его последовательная установка. С другой стороны, при последовательной установке между каждыми следующими друг за другом простыми обновлениями система должна поработать, должно пройти время. Следовательно, при более мелком разбиении время установки всего полного обновления также будет расти, так как шагов (т.е. простых обновлений) будет больше. Если предположить, что промежуток времени между соседними обновлениями примерно один и тот же, то время, которое система должна проработать в *продукции*, при установке всего полного обновления $U$ будет таким:

$$T(U) = T_c \cdot N + \sum T(u_i),\ i=1, \ldots, N,$$

где $T_c$ – время работы системы между соседними простыми обновлениями; $N$ – количество простых обновлений $u_i$, составляющим $U$, а $T(u_i)$ – время установки $i$-го обновления.

Если считать, что установка обновлений $u_i$ проводится во внерабочее время и (или) что время установки каждого $u_i$ мало по сравнению со временем $T_c$ работы системы между простыми обновлениями, то время установки $T(u_i)$ каждого $i$-го простого обновления можно не учитывать и второе слагаемое в формуле опустить:

$$T(U) \approx T_c \cdot N. \qquad (2)$$

Таким образом, можно приближенно считать, что время установки всего полного обновления $U$ прямо пропорционально количеству простых обновлений. При отсутствии необходимости уложиться в определенное время при обновлении системы можно на этом остановиться и проводить последовательные обновления в комфортном режиме, т.е. дробить полное обновление на желаемое количество частей и устанавливать эти части одну за другой с подходящим расстоянием во времени. Если обновление в целом достаточно масштабное, то его установка в таком комфортном режиме установки может сильно затянуться.

Рассмотрим теперь ситуацию, когда время ограничено и необходимо уложиться в определенные рамки. В этом случае необходимо повлиять на время $T(U)$, т.е. сократить его. Согласно (2) это можно сделать двумя способами: уменьшая время работы системы $T_c$ между простыми обновлениями и сокращая количество простых обновлений в составе полного.

Посмотрим, что будет происходить, если начнем сокращать время между соседними обновлениями. Для этого обратимся к рис. 1, 2, где изображены зависимости вероятности успеха установки обновления от времени, которое проработала система с момента установки. Из него следует, что вероятность успеха тем выше, чем больше времени прошло. Соответственно, при сокращении времени между обновлениями вероятность успеха каждого простого обновления снижается (рис. 3).

Рассмотрим теперь второй из возможных вариантов: сокращение количества простых обновлений. По сути дела, это означает, что за один раз устанавливается не одно обновление $u_i$, а сразу несколько простых обновлений от $u_{i-k}$ до $u_i$. В этом случае вероятность успеха установки этих нескольких обновлений в одном пакете будет равна произведению вероятностей успеха каждого из обновлений:

$$P(u_{i-k}, \ldots, u_i) = p(u_{i-k}) \cdot p(u_{i-k+1}) \cdot \ldots \cdot p(u_i).$$

Так как любая $p(u_i)$ меньше единицы, то в соответствии с (1) можно сделать вывод, что вероятность успешной одновременной установки нескольких обновлений ниже вероятности установки каждого обновления отдельно.

Рассмотрев оба варианта сокращения времени, приходим к выводу, что сократить время установки обновления можно только за счет повышения риска сбоя системы. Повышенный риск, в свою очередь, может быть либо следствием сокращения времени работы системы между простыми обновлениями, либо следствием установки нескольких простых обновлений сразу (причем второй вариант сводится к первому, так как при этом время между этими обновлениями сокращается до нуля). На рис. 4 изображены оба варианта.

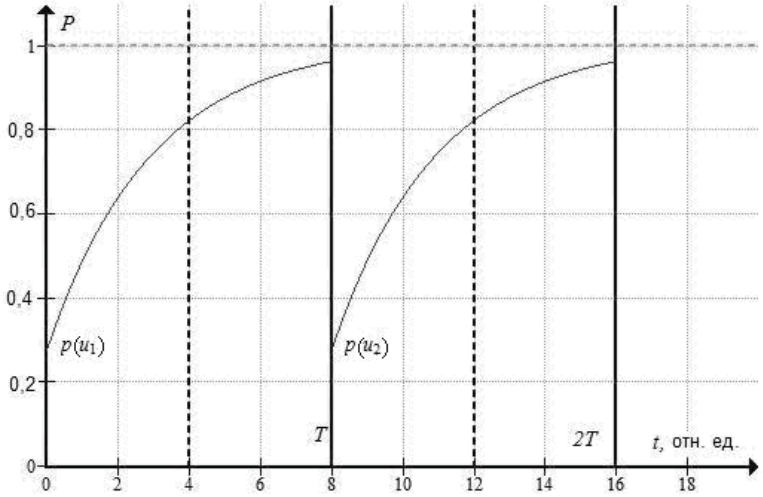


Рис. 3. Влияние сокращения времени между соседними обновлениями на их вероятности успеха

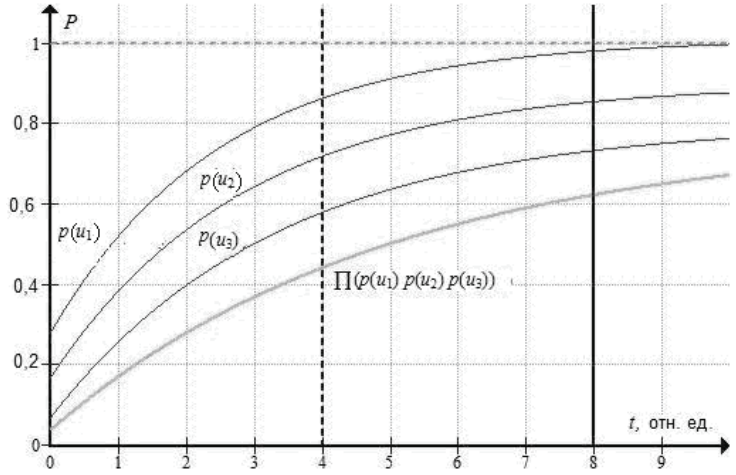


Рис. 4. Вероятность успеха при одновременной установке нескольких обновлений

**Тестирование предложенной методики**

Для апробации предложенной методики было выбрано финансово-кредитное учреждение (ФКУ) с головным отделением в Санкт-Петербурге и филиалом в Москве. Особенностью выбранного ФКУ является большая нагрузка на московский филиал за счет сосредоточения в нем большей части сотрудников и клиентов. В каждом из офисов есть набор серверов, работающих под ОС Microsoft Windows 2003, в том числе сервер сетевой печати, который используется сотрудниками каждого из офисов.

После получения обновлений (приблизительно 30) с сайта компании Microsoft было принято решение об их установке на серверы в обоих филиалах. Проведенное предварительное тестирование на тестовых серверах подтвердило работоспособность серверов и корректное выполнение всех заявленных функций, в том числе и функции сервиса печати. По результатам тестирования было решено проводить обновление поэтапно, сервер за сервером, сначала в офисе в Санкт-Петербурге, а затем в Московском филиале.

На следующий день после окончания установки всех обновлений на рабочие серверы в офисе в Санкт-Петербурге все заявленные функции выполнялись корректно. По истечении недели после успешно проведенного обновления в Санкт-Петербургском офисе обновление провели в Московском филиале.

В середине следующего дня после установления обновлений в Московском филиале на сервере печати возникли неполадки. Они проявлялись в том, что отправленные на печать документы не распечатывались на соответствующих принтерах. При этом задания корректно отображались в очереди на печать каждого из подключенных устройств, но ни удалить то или иное задание, ни полностью очистить очередь печати не представлялось возможным. Выключение и включение принтеров тоже не давало никаких результатов. Помогал только перезапуск сервиса печати (spooler), однако это давало лишь временное облегчение, так как по мере накопления документов в очереди на печать ситуация повторялась. Кроме того, при перезапуске сервиса печати часть заданий, уже отправленных на печать, терялась.

Ввиду критичности данного сервиса было принято решение восстановить из резервной копии образ сервера печати на момент до установки этих обновлений. Так как проблема на тестовом окружении не воспроизводилась, было решено устанавливать по одному обновлению в день на рабочий сервер печати, а на следующий день наблюдать за ситуацией. Действуя таким образом, только на седьмой день удалось определить обновление, которое вызывало описанные проблемы с печатью. Это обновление было удалено, задержка по времени была минимальной, и после этого сервер печати стал корректно выполнять свои функции.

В дальнейшем причина такого поведения сервиса печати при установке данного обновления была установлена. Как было сказано в инструкции, прилагавшейся к данному обновлению, оно устраняло критическую проблему в системе безопасности и очень рекомендовалось к установке. В ходе более подробного изучения материалов из базы знаний Microsoft выяснилось, что данное обновление исправляло ошибку в одной из системных библиотек, которую использует в своей работе сервис печати. При этом изменилось значение по умолчанию одного из параметров, который влияет на обработку заданий на печать. Было установлено, что для корректного функционирования сервиса печати при том количестве принтеров и объеме документов, которые печатались в московском офисе, необходимо предварительно установить большее, чем по умолчанию, значение для данного параметра в конфигурации сервиса печати, а уже затем устанавливать данное обновление. В Санкт-Петербургском офисе проблема не проявилась, так как объем печати там существенно меньше и новое значение по умолчанию соответствовало нагрузке на сервис печати.

## Заключение

В работе введено понятие простого обновления, рассмотрена стратегия обновления на основе использования простых обновлений, проведен анализ вероятности успеха обновления системы без использования стратегии простых обновлений и с использованием данной стратегии, доказано, что при использовании стратегии простых обновлений риск сбоя системы снижается, приведен пример, иллюстрирующий данный подход.

## Литература

1. IBM (1980). A Management System for the Information Business. White Plains. – New York: IBM.
2. Information Technology Infrastructure Library, Wikipedia [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/ITIL, свободный. Яз. англ. (дата обращения 19.02.2011).
3. COBIT, Wikipedia [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/COBIT#COBIT_and_ISO.2FIEC_27002:2007, свободный. Яз. англ. (дата обращения 19.02.2011).
4. Microsoft Operations Framework, Wikipedia [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Microsoft_Operations_Framework, свободный. Яз. англ. (дата обращения 19.02.2011).
5. ISO/IEC 20000, Wikipedia [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/ISO/IEC_20000, свободный. Яз. англ. (дата обращения 19.02.2011).

***Арустамов Сергей Аркадьевич*** – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, доктор технических наук, профессор, arustamov@gmail.com

***Генин Михаил Геннадьевич*** – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, gmg@rambler.ru