Алгоритм динамического распределения задач по абонентскому обслуживанию клиентов веб-студии в режиме реального времени

С.Б. Бегенова[a], Т.В. Авдеенко[a]

[a] *Новосибирский государственный технический университет, 630073, пр. К. Маркса, 20, Новосибирск, Россия*

Аннотация

Многие владельцы веб-сайтов обращаются к веб-студиям для решения задач поддержки своих веб-проектов, чаще всего на абонентской основе. Если веб-студия профессионально занимается поддержкой веб-проектов, то у нее могут быть десятки и сотни клиентов, которые в месяц создают сотни заявок на выполнение работ. Трудовые ресурсы веб-студии, как правило, ограничены. Поэтому актуальной является создание алгоритма распределения ресурсов (программистов, занимающихся выполнением данного вида работ) между задачами - заявками клиентов. В настоящей статье рассматриваются два альтернативных подхода к составлению расписания абонентского обслуживания клиентов веб-студии. Реализован алгоритм динамического назначения работ в реальном времени с учетом динамического характера данного процесса. Исследуется эффективность предлагаемого подхода.

Ключевые слова: динамическая теория расписаний; математическая модель; обслуживание клиентов; назначения в режиме реального времени; веб-студия; распределение задач

1. Введение

При работе над проектами, реализуемыми веб – студиями, задействуются специалисты различной направленности: менеджеры, программисты, бизнес – аналитики, дизайнеры и другой технический персонал. Однако основной силой, на которой держатся проекты, и основным дефицитным ресурсом, по словам управляющих данными студиями, являются программисты.

Основными направлениями, над которыми работают программисты веб-студий, являются [1]:

- Создание сайтов (сбор требований, разработка прототипов ключевых страниц, разработка технического задания, разработка дизайна сайта, верстка дизайна сайта, программирование и настройка сайта, тестирование и запуск проекта в эксплуатацию);
- Гарантийное (абонентское) обслуживание.

Многие владельцы веб-сайтов обращаются в специализированные организации, веб-студии, для решения задач поддержки и развития своих веб-проектов, чаще всего на абонентской основе. Абонентское обслуживание подразумевает техническую поддержку сайта и решение поставленных разработчиком или клиентом задач в рамках оплаченных часов. Если веб-студия профессионально занимается поддержкой веб-проектов, то у нее могут быть десятки и сотни клиентов, которые в месяц создают сотни заявок на выполнение работ. При этом обычно в договоре поддержки фиксируется максимальное время реакции на заявку клиента. Трудовые ресурсы веб-студии, как правило, ограничены. Поэтому актуальной является оптимизация алгоритма распределения ресурсов (программистов, занимающихся выполнением данного вида работ) между задачами - заявками клиентов.

Традиционно для решения проблемы распределения ресурсов (машин) между задачами, которые необходимо выполнить, используется теория расписаний [2,3]. Однако в случае решения задачи распределения задач абонентского обслуживания применение статических методов теории расписаний недостаточно. В большинстве реальных сред, возникающие непредвиденные обстоятельства, требуют пересмотра или изменения расписаний. Такие обстоятельства могут касаться либо ресурсов, либо самих операций. События, относящиеся к ресурсам – это поломка машин, поломка инструментов, недоступность инструментов или человеческих ресурсов, нехватка материала или компонент, некачественный материал и др. События, связанные с операциями – это, к примеру, изменение дедлайнов, отмена заказов, поздно прибывшие заказы, изменения в процессе производства из- за замены ресурсов и т.д. Таким образом, расписание часто становится неактуальным еще до момента своего завершения.

Подобная ситуация подтолкнула к развитию так называемую динамическую теорию расписаний [4], набор подходов, которые реагируют на неожиданные события либо корректировкой существующего расписания, либо перепланированием оставшихся операций. В условиях современной динамической среды более эффективным является использование именно динамической теории расписаний. В статье рассматриваются и анализируются два альтернативных подхода к составлению расписания абонентского обслуживания клиентов веб-студии: динамическое планирование и распределение задач в режиме реального времени.

Оставшаяся часть статьи организована следующим образом. В разделе 2 «Анализ данных веб - студии» анализируется текущая загруженность программистов веб – студии на основе построенной по реальным данным диаграммы Гантта. В разделе 3 рассматриваются два динамических подхода к построению расписания: динамическое планирование и распределение задач в режиме реального времени, приведены описания и сам алгоритм. В разделе 4 приведены тесты реализованного алгоритма «Назначение работ в режиме реального времени». В разделе 5 формулируются выводы по работе.

2. Анализ данных веб - студии

Для решения проблемы распределения задач абонентского обслуживания были получены и проанализированы реальные данные конкретной веб–студии по задачам, выполненным за некоторый период времени в рамках абонентского обслуживания. Текущий принцип распределения задач заключается в использовании ПО, работающем на принципе приоритетности и «ручном» распределением задач абонентского обслуживания по ресурсам (программистам веб – студии) менеджером проектов. В данном случае задачи распределяются с использованием принципа приоритетности: задачам в зависимости от их срочности присваиваются следующие значения:

- нормальная
- срочная
- критичная

Следствием ручного распределения задач являются «простои» в расписании программистов, т.е. периоды времени, в которые задач для выполнения у программиста не было. Для анализа загруженности текущего расписания программистов была построена диаграмма Гантта на данных, полученных веб – студией (рисунок 1).

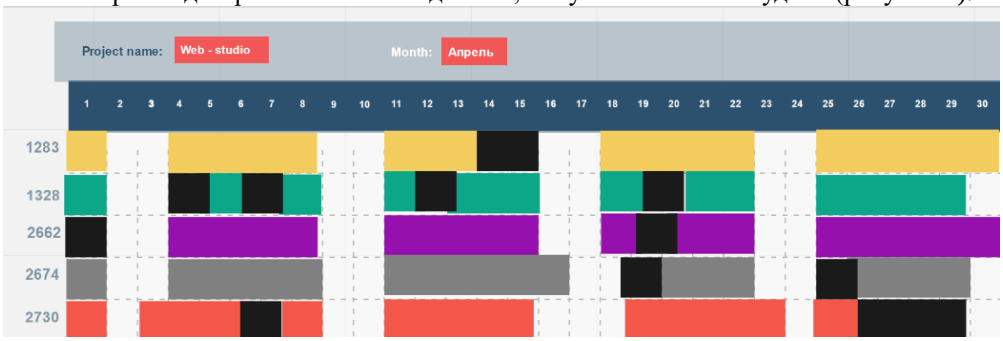

Рис. 1. Диаграмма Гантта для апреля 2016 г.

На данной диаграмме представлено расписание программистов веб – студии за апрель 2016 г. Черным цветом на диаграмме отмечены «простои», то есть дни, в которые у программистов на выполнении не было задач абонентского обслуживания. По вертикали расположены номера – идентификаторы программистов, по горизонтали – дни работы. На диаграмме видно, что у программистов с идентификаторами 1283, 2674 и 2662 простои в работе были зафиксированы 2 раза (простоем считается отсутствие работы в течение целого дня). У программистов 1328, 2730 такая цифра доход до 5. Учитывая, что в данные дни программисты могли, наряду с реализацией проектов, выполнять абонентские задачи, загруженность программистов по задачам абонентского обслуживания далеко не полная. Поэтому рассмотрим подходы к динамическому построению расписаний как способ улучшения и оптимизации текущего распределения задач абонентского обслуживания.

3. Динамическое планирование и назначение работ в режиме реального времени

В данном разделе рассмотрим два подхода к построению расписания, учитывающие временной фактор, предложенные в [4].

3.1. Динамическое планирование

В динамическом планировании используется подход классической архитектуры производства, подразумевающей, что производственная система представляет собой объединение отделов, которыми руководят разные менеджеры. Рассматриваемый подход заключается в том, что в начале периода управления строится статическое расписание с использованием методов классической теории расписаний. После того, как производство запускается в соответствие с построенным таким образом расписанием, возможно возникновение непредвиденных ситуаций, нарушающих стабильный процесс. В этих случаях расписание корректируется таким образом, чтобы наиболее оптимальным образом учесть нарушения.

Одним из методов динамической теории расписаний является использование следующих эвристик:

- по сроку получения задачи (правило FIFO, first in- first out)
- по количеству операций (задача, выполняющаяся наиболее быстро, имеет более высокий приоритет)
- по стоимости (с высоким приоритетом идут задачи, выполнение которых не в срок влечет наибольшие финансовые потери)
- по дедлайнам (с высоким приоритетом идут наиболее срочные задачи, выполнение которых безотлагательно)
- по времени выполнения операций

3.2. Назначение работ в режиме реального времени

Метод назначения в режиме реального времени был предложен в связи с распространением парадигмы цепей поставок. В цепи поставок каждый проект охватывает целый производственный цикл, начиная от требований

клиентов до окончательных расчетов. Кроме того, разнообразие продуктов, вовлеченных в проект, ограничено с точки зрения числа типов операций. Сегодня под проектом понимается непрерывный поток деятельности. В частности, производственные системы постепенно переходят от малосерийного производства к конвейерным линиям сборки, которые гарантируют не только производительность, но и гибкость системы. Таким образом, назначение работ в режиме реального времени заключается в назначении набора операций (работ) набору ресурсов по прибытии заказа в производственную систему. В случае нарушения выполнения какой-либо работы ранее запланированные незавершенные работы перераспределяются в порядке, соответствующем спросу. Цель данного подхода заключается в перераспределении операций в режиме реального времени.

Рассмотрим задачу назначения работ в режиме реального времени с зафиксированными предыдущими назначениями.

Предположим, что задача i может быть выполнена любым из программистов $\{m_i^1, m_i^2, m_i^3, ..., m_i^{K_i}\}$, и программист $m_i^k, km\{1,2,\ldots,K_i\}$ находится в режиме ожидания в периодах $I_i^k = \left\{\left[\alpha_{i,q}^k, \beta_{i,q}^k\right]\right\}_{q=1,2,\ldots,Q_{k,i}}$. Таким образом, K_i - максимальное количество программистов, которые в состоянии выполнять задачу i, а $Q_{k,i}$ - это максимальное число периодов простоя, доступных для задачи i у программиста m_i^k.

Для случая нескольких программистов, работоспособность которых идентична, сгруппируем периоды простоя, связанные с одной группой таких программистов, следующим образом.

Для $k_1 \neq k_2$, где $k_1, k_2 \in \{1,2,\ldots,K_i\}$, период $\left[\alpha_{i,q}^{k_1}, \beta_{i,q}^{k_1}\right]$,

$q \in \{1,2,\ldots,Q_{k_1,i}\}$ предшествует $\left[\alpha_{i,r}^{k_2}, \beta_{i,r}^{k_2}\right]$, $r \in \{q=1,2,\ldots,Q_{k_2,i}\}$ если:

1. $\alpha_{i,q}^{k_1} < \alpha_{i,r}^{k_2}$, или
2. $\alpha_{i,q}^{k_1} = \alpha_{i,r}^{k_2}$ и $\beta_{i,q}^{k_1} < \beta_{i,r}^{k_2}$.

Последовательность таких отсортированных периодов обозначается $\left[\alpha_i^s, \beta_i^s\right]$, где $s = 1,2,\ldots,\sum_{k=1}^{K_i} Q_{k_i,i} = Q_i$;

s_i – ранг периода простоя, назначенного i- ой задаче;
t_i –время начала выполнения задачи i;
θ_i – время, необходимое для выполнения задачи i;
δ_i – максимальная задержка выполнения задачи i, разрешенная для определенного программиста;
m – количество задач;
$\alpha_i^{s_i}$ – начальная точка s_i периода простоя, который может быть назначен задаче i;
$\beta_i^{s_i}$ – конечная точка s_i периода простоя, который может быть назначен задаче i;
В данном подходе используется следующий алгоритм:

1. Задать $s_i = 1$ для i =1,2,…,m;
2. Задать $p_1 = \alpha_1^{s_1}$;
3. Задать $p_i = MAX(\alpha_i^{s_i}, p_{i-1} + \theta_{i-1})$,i=2,…,m;
4. Задать $p_{m+1} = p_m + \theta_m$;
5. Задать $t_{m+1} = p_{m+1}$;
6. Задать $t_i = MAX(p_i, t_{i+1} - \theta_i - \delta_i)$ для всех i = m,m-1,…1;
7. Если $(t_{i+1} > \beta_i^{s_i})$ для всех i=1,2,…,m, оптимум достигнут;

Иначе для каждогоi для которого $(t_{i+1} > \beta_i^{s_i})$, задаем $s_i = s_i + 1$ и возвращаемся к шагу 2.

4. Исследования работы алгоритма

Приведем тесты программной реализации алгоритма «Назначение работ в режиме реального времени» в программной среде MATLAB.

Тестовый пример с 2 программистами

Рассмотрим тестовый пример, в котором задачи будут приходить на исполнение 2 программистам. Опишем начальные данные, необходимые для исполнения алгоритма.

Выберем следующее время выполнения, необходимое для завершения задач:

- Первая задача будет выполняться 1 единицу времени
- Вторая задача будет выполняться 2 единицы времени

Максимальное время ожидания, которое может быть у программиста – это 1 единица времени. Определим периоды простоя программистов, то есть те периоды, в которые они могут взять на выполнение задачи абонентского обслуживания.

Периоды простоя первого программиста- [0,3], [5, 10], [15,+∞].

Периоды простоя второго программиста- [0,6], [16, 19], [25,+∞].

s – результирующая переменная, которая отображает номер периода простоя в который была назначена конкретная задача.

Результаты выполнения алгоритма представлены на рисунках 2, 3. На рисунке 3 черными прямоугольниками представлены периоды занятости программистов, а зелеными – периоды времени, в которые будут выполнены задачи.

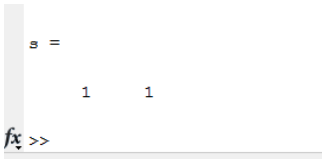

Рис. 2. Результат выполнения программы для первого теста.

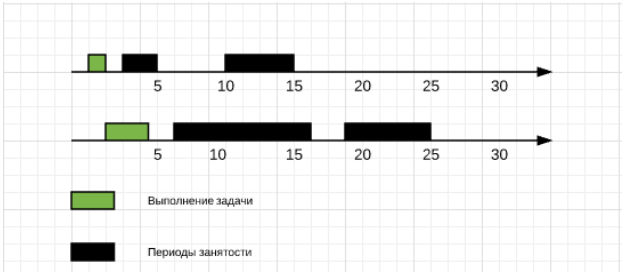

Рис. 3. Решение, найденное с использованием алгоритма.

Для данного теста две задачи будут последовательно выполнены в первые периоды простоя обоих программистов.

Тестовый пример с 4 программистами

В данном тестовом примере задачи будут распределяться между 4 программистами.

Время выполнения, необходимое для завершения задачи:

- для первой задачи– 3
- для второй задачи– 4
- для третьей задачи– 5
- для четвертой задачи– 5

Максимальное время ожидания для ресурса – 1.

Периоды простоя первого программиста- [0,2], [5, 15], [25,+∞].

Периоды простоя второго программиста–[0,15], [25,+∞].

Периоды простоя третьего программиста- [0,5], [10,25], [30,+∞].

Периоды простоя четвертого программиста- [0,10], [20,+∞].

Результаты выполнения алгоритма представлены на рисунках 4, 5.

Рис. 4. Результат выполнения программы для второго теста.

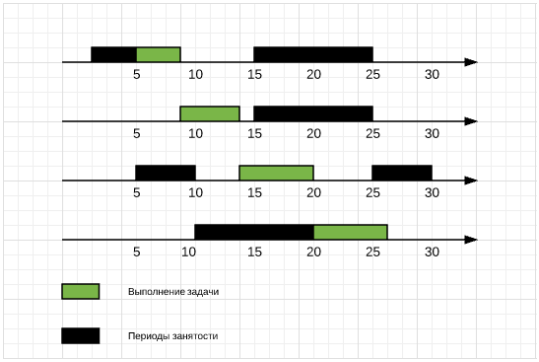

Рис. 5. Решение, найденное с использованием алгоритма.

Для данного теста задачи будут выполнены последовательно во вторые периоды простоя всех программистов, кроме второго.

Тестовый пример с 8 программистами

Рассмотрим результаты выполнения данного алгоритма при распределении задач для 8 программистов.

Время выполнения, необходимое для завершения задач:

- для первой задачи – 3
- для второй задачи– 2

- для третьей задачи– 3
- для четвертой задачи – 2
- для пятой задачи– 3
- для шестой задачи – 5
- для седьмой задачи– 4
- для восьмой задачи – 5

Максимальное время ожидания для ресурса – 1.

Периоды простоя первого программиста- [0,2], [5, 15], [25,30], [35,+∞] .
Периоды простоя второго программиста- [10,15], [25,27], [30,+∞].
Периоды простоя третьего программиста- [0,7], [13, 17], [20,24], [29,+∞] .
Периоды простоя четвертого программиста- [0,5], [10, 12], [16,22],[31,+∞] .
Периоды простоя пятого программиста- [10,15], [20,27], [30,+∞].
Периоды простоя шестого программиста -[7,11], [15,20], [25,30], [35,+∞] .
Периоды простоя седьмого программиста- [6,10], [14,24], [29,35], [43,+∞] .
Периоды простоя восьмого программиста- [0,5], [10,12], [16, 18], [20,22], [31,+∞] .
Результаты выполнения алгоритма представлены на рисунках 6, 7.

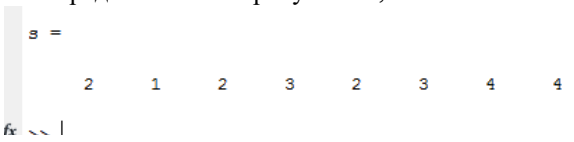

Рис. 6. Результат выполнения программы для третьего теста.

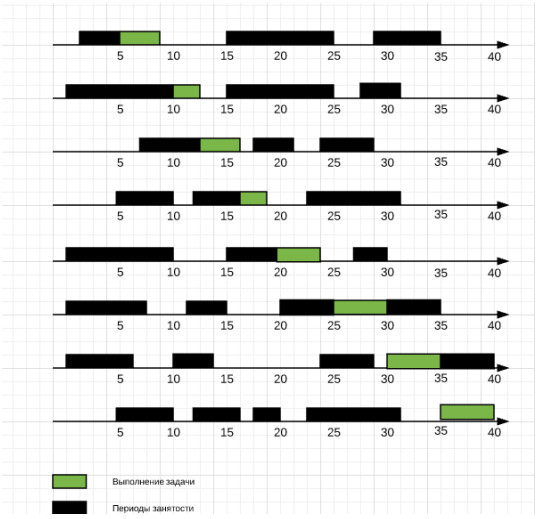

Рис. 7. Решение, найденное с использованием алгоритма.

В тестовых примерах задачи распределились равномерно с учетом периодов простоя программистов и их максимального времени ожидания. На тестовых примерах видно, что алгоритм проводит поиск наиболее подходящего периода простоя для выполнения задачи. В случае если в рассматриваемый период простоя задача не успеет выполниться либо максимальное время ожидание для ресурса будет превышено, то алгоритм начинает поиск другого ближайшего подходящего периода простоя.

Тестирование алгоритма «Назначение работ в режиме реального времени» выявило основные достоинства подобного подхода. Необходимость построения нового расписания в случае каких - либо поломок или любых других непредвиденных обстоятельств, которые делают существующее расписание неактуальным, теперь отпадает. Особенностью данного подхода являются зафиксированные предыдущие назначения. Начальными данными для алгоритма являются расписание программистов, периоды простоя программистов, а также поступающие на выполнение задачи с соответствующими временными оценками. Таким образом, данный подход позволяет выстроить расписание выполнения поступающих задач без изменения текущего расписания программистов. Также достоинством данного метода является низкая трудоемкость и высокое быстродействие, что, несомненно, важно для практического применения метода.

Анализ данных веб–студии в виде диаграммы Гантта показал, что текущее распределение задач в веб- студии недостаточно оптимально в плане загрузки программистов задачами в течение рабочего времени. Применение данного подхода позволяет заполнить и устранить периоды простоев в работе программистов.

На данном этапе исследований, подход «Назначение работ в режиме реального времени» рассматривает процесс выполнения задач как последовательную процедуру, что не всегда соответствует текущему принципу исполнения

задач в веб – студиях. Поэтому перспективным направлением дальнейшей работы является более точная адаптация алгоритма под проблему распределения задач абонентского обслуживания. То есть учет возможности параллельного выполнения задач программистами [3, 4], учет приоритетности задач, а также учет приоритетности назначения задач. В последнем пункте будет учитываться приоритетность программиста при распределении абонентских задач: программист, реализовавший веб – проект, имеет больший приоритет в очереди на абонентские задачи от данного проекта.

В алгоритме «Назначение работ в режиме реального времени с возможным изменением предыдущего расписания» в случае, когда на выполнение поступает задача с высоким приоритетом, возможен сдвиг на более поздние сроки других задач, выполнение которых не так срочно. Подобное изменение необходимо для немедленного выполнения срочной задачи. В сравнении с таким подходом, реализованный нами алгоритм «Назначения работ в режиме реального времени» не позволяет изменять установленное ранее расписание, что не всегда удобно в реалиях работы веб – студии.

5. Заключение

Анализ данных о загруженности работников задачами абонентского обслуживания на определенный период показал, что текущий принцип распределения задач дает неоптимальную загруженность сотрудников. Таким образом, проведенный анализ выявил актуальность исследования и разработки методов распределения задач абонентского обслуживания веб – студии.

В ходе исследования динамических подходов построения расписаний был реализован и протестирован метод назначения работ в режиме реального времени в программном пакете MATLAB. В результате проведенных исследований были выявлены текущие преимущества и недостатки данного алгоритма. Несомненным преимуществом рассмотренного подхода является возможность построения расписания работ программистов без изменения их текущего распорядка исполнения задач. Алгоритм позволяет наиболее оптимально заполнить периоды простоев программистов, таким образом позволяя равномерно распределить нагрузку на программистов и уменьшить периоды простоев.

В перспективе рассматривается возможность улучшения рассмотренного подхода «Назначение работ в режиме реального времени» путем добавления критерия, учитывающего большинство особенностей работы в сфере абонентского обслуживания. Такими особенностями являются установка и учет приоритетов при выборе ресурса (программиста) для определенной задачи, учет ограничений, связанных с отработкой оплаченных клиентом часов за абонентское обслуживание, а также учет самого критерия оптимальности модели – максимизации прибыли. Также в перспективе – интеграция алгоритма с различными эвристиками.

Благодарности

Работа поддержана грантом Министерства образования и науки РФ в рамках проектной части Госзадания, проект № 2.2327.2017/ПЧ «Интеграция моделей представления знаний на основе интеллектуального анализа больших данных для поддержки принятия решений в области программной инженерии».

Литература

[1] Авдеенко, Т. В. О возможностях применения методов и моделей теории расписаний для оптимизации работы веб-студии / Т. В. Авдеенко, Р. В. Петров // Сборник научных трудов Новосибирского государственного технического университета. - 2016. – № 2 (84). – С. 7–20.

[2] Brucker, P. Scheduling Algorithms (Fifth ed.) / P. Brucker – Springer, 2007. – 372 p.

[3] Pinedo, M. Scheduling Theory, Algorithms, and Systems (3nd. ed.) / M. Pinedo - Springer, 2008. – 672 p.

[4] Dolgui, A. Supply Chain Engineering - Useful Methods and Techniques / A. Dolgui, J.-M. Proth – Springer –Verlag, 2010. – 541 P.

[5] Avdeenko, T. V. Efficient approaches to scheduling for unrelated parallel machines with release dates / T. V. Avdeenko, Y. A. Mesentsev // IFAC-PapersOnline. - 2016. - Vol. 49, iss. 12. - P. 1743-1748.

[6] Авдеенко, Т.В. Мультиагентный подход с использованием нечеткого моделирования в задачах многокритериального принятия решений / Т.В.Авдеенко, М.А.Васильев // Научный вестник Новосибирского государственного технического университета. – 2010 . –№1. – С. 63-74.

[7] Авдеенко, Т.В.Проблемы параметрической идентификации в математическом моделировании процессов / Т.В.Авдеенко // Образовательные ресурсы и технологии.– 2014. – № 1(4).–С. 115-124.

[8] Лазарев А.А. Теория расписаний. Задачи и алгоритмы : учеб. пособие / А.А.Лазарев, Е.Р.Гафаров. – М.: Изд-во МГУ, 2011.– 222 с.

[9] Павлов, А.А. Модели и алгоритмы теории расписаний в задачах планирования и управления проектами / О.А. Павлов, С.К. Чернов, О.Б. Мисюра // Труды Одесского политехнического университета – 2006. – вып. 1(25).– С. 150-159 .

[10]Mezentsev, Yu.A. Scheduling and Optimization for Parallel-Serial Service Systems / Yu.A. Mezentsev, T.V. Avdeenko // Proceedings of 8th International Forum On Strategic Technology – 2013. – P. 271 - 275.