М.А. Плаксин, О.А. Пономарева[1]

Национальный исследовательский университет
«Высшая школа экономики»
(Пермский филиал)

mapl@list.ru
Ponomareva.olga.ru@mail.ru

РЕГИСТРАЦИЯ ЗАДАЧ И ОШИБОК В TEAM FOUNDATION SERVER КАК ИНСТРУМЕНТ УПРАВЛЕНИЯ ПРОГРАММНЫМ ПРОЕКТОМ

Введение

В ходе реализации программного проекта возникает необходимость в протоколировании процесса взаимодействия разработчиков. В качестве инструмента для этого часто для этого используется Team foundation server (TFS). В его информационной базе регистрируются такие действия, как создание новой задачи, назначение для нее программиста-разработчика, передача задачи на тестирование (назначение для нее тестировщика), обнаружение в ней ошибки, передача ошибки для исправления (назначение для нее программиста), фиксация исправленной ошибки, окончание тестирования задачи и др.

В большом числе случаев информационная база TFS используется как журнал регистрации событий и инструмент назначения работников для выполнения той или иной операции. Авторы данной статьи считают, что информация, собираемая TFS, может гораздо более плодотворно применяться для управления процессом реализации программного

[1] © Плаксин М.А., Пономарева О.А., 2016

проекта, оценки успешности этого процесса и выявления возникающих проблем, как во время проекта, так и по его окончании (на этапе пост-проектного анализа).

В данной статье рассматривается вопрос об использовании информационной базы TFS для управления программным проектом.

Применяемая модель процесса управления программным проектом

В рамках данной статьи применяется следующая модель управления программным проектом.

Проект реализуется по согласованию с неким заказчиком.

В реализации проекта принимает участие команда, члены которой исполняют четыре роли: руководитель проекта (менеджер проекта, ПМ), аналитики, программисты, тестировщики. На каждой роли, кроме ПМ, может находиться несколько человек (в команде может быть несколько аналитиков, несколько программистов, несколько тестировщиков).

Программный продукт разрабатывается в виде последовательности релизов. Для разработки каждого релиза назначается некоторое ожидаемое (планируемое) время. Планируемое время определяется совместно заказчиком, руководителем проекта и аналитиком. Согласованный релиз менеджер проекта регистрирует в информационной базе TFS.

Каждый релиз состоит из набора задач. Перечень задач определяется аналитиком по согласованию с заказчиком и руководителем проекта. Задачи отличаются приоритетами и сложностью. Высокоприоритетные задачи должны быть реализованы ранее низкоприоритетных. Сложность задачи выражается в ожидаемом (планируемом) времени ее реализации. Приоритет задачи определяется аналитиком по согласованию с заказчиком. Планируемое время реализации задачи назначается аналитиком экспертным путем. Если задача была запланирована на текущий релиз, но по какой-либо причине не была реализована, она переходит на следующий релиз.

Работа над релизом начинается с того, что аналитик регистрирует в информационной базе TFS задачи, реализация которых запланирована на данный релиз.

Реализацию каждой задачи руководитель проекта поручает одному из программистов. После того, как программист завершит реализацию задачи, руководитель передает ее для тестирования одному из тестировщиков. Если тестировщик в процессе тестирования задачи обнаружил ошибку, он регистрирует ее на TFS, указав «уровень серь-

езности» ошибки. Подразумевается, что более серьезные ошибки потребуют для своего исправления больше времени. Серьезность ошибки определяется тестировщиком экспертным путем. Кроме «серьезности» для ошибки может быть задан приоритет, который по аналогии с задачей означает, что высокоприоритетные ошибки должны быть исправлены ранее низкоприоритетных. Приоритет задается аналитиком.

Зарегистрированная тестировщиком ошибка передается менеджером проекта одному из программистов для исправления. Преимущественно, это должен быть программист, реализовывавший данную задачу. Но если программистов в проекте несколько, то менеджер может выбрать любого из них с учетом их текущей загрузки, серьезности ошибки и ожидаемых затрат времени на ее исправление. (Например, если ошибка сводится к неверному написанию слова на экране, то исправить ее может любой.)

Завершив корректировку задачи, программист регистрирует этот факт в информационной базе TFS. После чего руководитель проекта передает исправленную задачу тестировщику для повторной проверки.

Если тестирование задачи закончено, а ошибок не обнаружено, тестировщик регистрирует этот факт в TFS.

Работа над релизом заканчивается, если

– тестирование всех задач релиза закончено в отведенный для этого промежуток времени,

– истекло время, отведенное на разработку данного релиза, хотя какие-то из входящих в него задач еще не реализованы и/или не оттестированы.

Факт окончания работы над релизом регистрирует в информационной базе руководитель проекта. Если в релизе остались незаконченные задачи, эти задачи аналитик должен переназначить на новый релиз после того, как этот релиз будет зарегистрирован в TFS.

Состояние объектов информационной базы Team foundation server

В информационной базе TFS размещается информация о следующих объектах: релизы (release), задачи (task), ошибки (bug). В реальной жизни в TFS могут храниться также объекты иных типов. Но в данной статье мы ограничимся этими тремя.

Релизы и задачи имеют имена, ошибки – тексты сообщений.

Объекты объединяются в иерархию: релиз – задачи – ошибки. Каждая задач относится к одному из релизов. Каждая ошибка относится к одной из задач. На рис.1 приведен пример иерархии «релиз – задачи – ошибки».

Work Item	Title	State	Severity	Priority
Product Release	Выпуск версии 1.5	Closed	3 - Medium	1
Task	Поиск показателя деятельности	Closed		2
Bug	Поиск показателя деятельности	Closed	3 - Medium	2
Task	КЛ Реализовать проверки на даты создания пользователей подразделений	Closed		1
Bug	КЛ Присутствует возможность создания пользователя с датой создания меньше, чем у подразделе...	Closed	3 - Medium	1
Bug	КЛ При редактировании пользователя можно установить дату создания меньше, чем дата создани...	Closed	3 - Medium	1
Bug	КЛ При переводе пользователя в другое подразделение нет проверки на дату создания подразде...	Closed	3 - Medium	1
Bug	КЛ При создании подразделения отсутствует проверка на дату создания родителя	Closed	3 - Medium	1
Task	КЛ IM1915443, IM2021075, IM2021871, IM2036979 Разработка отчета Мониторинг	Clo sed		1
Bug	КЛ Некорректная сортировка пользователей в отчете Мониторинг	Clo sed	3 - Medium	1

Рис. 1. Пример иерархии «релиз – задачи – ошибки»

Любой из этих трех объектов в каждый конкретный момент находится в одним из четырех состояний: заведен (proposed), активен (active), разрешен (resolved), закрыт (closed). Общий смысл этих состояний следующий:

– заведен (proposed) – объект создан, но в данный момент над ним никто не работает. Когда объект «берут в работу», он переходит в состояние «активен»;

– активен (active) – над объектом в данный момент кто-то работает. По завершении работы объект переходит в состояние «разрешен»;

– разрешен (resolved) – работа нал объектом завершена, но еще не проверена. Объект находится в состоянии проверки. По результатам проверки объект может быть либо завершен окончательно (переведен в состояние «закрыт»), либо оставлен в состоянии «разрешен», либо возвращен на доработку (переведен в состояние «активен»);

– закрыт (closed) – работа над объектом завершена и проверена, дальнейших действия не требуется.

Более подробно смысл состояний для каждого класса объектов объясняется ниже.

Состояние верхних объектов иерархии «релиз – задачи – ошибки» определяется через состояние нижних: состояние релиза – через состояние задач, состояние задачи – через состояние ошибок.

Для ошибок состояния имеют следующий смысл:

– Заведена (proposed) – Возможны два варианта. Либо ошибка только что обнаружена тестировщиком и зарегистрирована им в информационной базе. Либо ошибка была взята в работу программистом (и перешла в состояние «активна»), но потом по какой-то причине программист от работы над ошибкой отказался (и ошибку из состояния

«активна» опять вернул в состояние «заведена». Соответственно, состояние может быть установлено тестировщиком или программистом.

– Активна (active) – Возможны два варианта. Либо для данной ошибки только что был назначен программист (и подразумевается, что он ведет работу над ее исправлением). Либо ошибка была исправлена программистом (перешла в состояние «разрешена»), была передана тестировщику для повторной проверки, но этой проверки не прошла (и из состояния «разрешена» была возвращена тестировщиком в состояние «активна»). Соответственно, состояние может быть установлено программистом или тестировщиком.

– Разрешена (resolved) – Программист объявил ошибку исправленной. Теперь она должна быть передана тестировщику для повторной проверки. Тестировщик проводит повторную проверку исправленной ошибки (по результатам которой ошибка будет либо закрыта, либо возвращена в состояние «активна»). Состояние может быть установлено только программистом.

– Закрыта (closed) – Тестировщик провел повторную проверку этой ошибки и признал ее исправленной. Состояние может быть установлено только тестировщиком.

Для задач состояния имеют следующий смысл:

– Заведена (proposed) – Задача только что зарегистрирована аналитиком в информационной базе. Состояние устанавливается аналитиком.

– Активна (active) – Программист начал реализацию данной задачи. Состояние устанавливается программистом.

– Разрешена (resolved) – Программист реализовал задачу. Задача проверяется тестировщиком. Обнаружение ошибок в задаче не влияет на ее состояние. Задача остается в состоянии resolved до тех пор, пока все относящиеся к ней ошибки не будут исправлены. Состояние устанавливается программистом.

– Закрыта (closed) – Тестировщик завершил проверку задачи. Либо ошибок не обнаружено, либо все обнаруженные ошибки исправлены, прошли повторную проверку и «закрыты» (переведены в состояние closed). Состояние указывается тестировщиком.

Для релиза состояния имеют следующий смысл:

– Заведен (proposed). – Релиз только что зарегистрирован менеджером проекта в информационной базе.

– Активен (active). Релиз находится в разработке. Релиз переходит в состояние active в тот момент, когда активной становится хотя

бы одна из его задач (хотя бы одна задача переходит в состояние active) и выходит из этого состояния в тот момент, когда все его задачи реализованы (переведены в состояние resolved или closed). Состояние устанавливается менеджером проекта.

– Разрешен (resolved). Все задачи и ошибки в релизе находятся в состоянии «разрешены» или в состоянии «закрыты». Это значит, что все задачи реализованы программистами, все обнаруженные к этому времени ошибки исправлены программистами. Однако часть задач еще находится на проверке у тестеров (первичной или повторной после исправления ошибок). Состояние устанавливается менеджером проекта.

– Закрыт (closed). Все задачи и ошибки в релизе закрыты. Состояние устанавливается менеджером проекта.

Доступ к информационной базе TFS имеют руководитель проекта, аналитик, программист и тестировщик. Заказчик доступа к TFS не имеет.

Действия руководителя проекта:

– регистрация релиза в информационной базе;
– назначение ожидаемого срока работы над релизом;
– назначение программистов для задач и ошибок;
– назначение тестеров для ошибок;
– назначение ожидаемого срока реализации каждой задачи;
– переназначение для задач, не реализованных в прежнем релизе и перенесенных в новый релиз, нового ожидаемого срока реализации.

Действия аналитика:

– регистрация задач, относящихся к данному релизу;
– указание приоритетов задач;
– переназначение в новый релиз задач, которые были первоначально зарегистрированы для другого релиза, но не были реализованы в срок;
– указание приоритетов ошибок.

Действия тестировщика:

– при обнаружении ошибки регистрирует ее в базе в состояния «заведена»;
– при повторной проверке исправленной ошибки (которую программист перевел в состояние «разрешена») переводит ее из состояния «разрешена» в состояние «закрыта», если он согласен с тем, что

ошибка исправлена, или возвращает в состояние «активна», если считает, что ошибка не исправлена;

– закрывает проверенную задачу (переводит проверенную задачу в состояние «закрыта», если он не нашел в ней нет ошибок или все найденные ошибки переведены в состояние «закрыта»).

Действия программиста:

– при получении задачи переводит ее из состояния «заведена» в состояние «активна»;

– при завершении реализации задачи переводит ее из состояния «активна» в состояние «разрешена»;

– при получении ошибки переводит ее из состояния «заведена» в состояние «активна»;

– при исправлении ошибки переводит ее из состояния «активна» в состояние «разрешена».

Часть информационной базы Team foundation server, используемая в данной статье

Для каждого из объектов в информационной базе TFS создается запись, называемая «рабочим элементом». Все записи имеют некоторый набор стандартных полей. При необходимости набор полей может быть расширен.

В табл. 1 перечислен ряд стандартных полей, которые упоминались выше или которые будут задействованы для целей управления проектом.

Таблица 1. Основные поля элементов в TFS, используемые в данной статье

Название поля	Перевод	Пояснение
ID	Номер	Уникальный идентификатор, создается автоматически. Используется для однозначной идентификации записи в информационной базе.
Work Item Type	Тип рабочего элемента	В нашем случае: релиз, задача, ошибка.
Title	Название	Заголовок релиза и/или задачи. Сообщение об ошибке для ошибки.
State	Состояние	Текущее состояние объекта

Название поля	Перевод	Пояснение
Reason	Причина	Поле указывается при изменении поля состояния. Например, разработчик может указать одну из следующих причин при разрешении ошибки: «Исправлено» (по умолчанию), «Отложено», «Дублирующаяся», «Не ошибка», «Не удается воспроизвести» или «Устаревшая». Каждая причина указывает на определенное действие для тестера в отношении ошибки
Severity	Серьезность	Указывается для ошибок. Может иметь следующие значения: Блокирующий (Blocker), Критический (Critical), Значительный (Major), Незначительный (Minor), Тривиальный (Trivial).
Priority	Приоритет	Указывается для задач и ошибок. Может иметь следующие значения: Высокий (High), Средний (Medium), Низкий (Low)
Customer	Аналитик	Аналитик, который зарегистрировал данную задачу.
Developer	Разработчик	Для задачи – программист, который назначен для реализации данной задачи. Для ошибки – программист, который назначен для исправления данной ошибки.
Tester	Тестировщик	Для задачи – тестировщик, который назначен для тестирования данной задачи. Для ошибки – тестировщик, который зарегистрировал данную ошибку.
Created By	Кем создано	Имя человека, создавшего объект
Activated By	Кем активировано	Имя человека, изменившего состояние рабочего элемента на активный.
Resolved By	Кем реализовано	Имя человека, изменившего состояние рабочего элемента на разрешенный.
Closed By	Кем закрыто	Имя человека, изменившего состояние рабочего элемента на закрытый.
Created Date	Дата создания	Дата и время создания рабочего элемент
Activated Date	Дата активации	Дата и время, когда рабочий элемент был переведен в состояние активного
Resolved Date	Дата реализации	Дата и время, когда рабочий элемент был переведен в состояние разрешенного.
Closed Date	Дата закрытия	Дата и время, когда рабочий элемент был переведен в состояние закрытого

Название поля	Перевод	Пояснение
Estimated Cost	Плановые затраты программиста	Время (в часах), которое было запланировано для программиста на работу с данным элементом: на разработку релиза, на реализацию задачи, на исправление ошибки.
Actual Cost	Фактические затраты программиста	Время (в часах), которое реально было потрачено программистом на работу с данным элементом.
Test Estimated Cost	Плановые затраты тестера	Время (в часах), которое было запланировано для тестера на работу с данным элементом: на проверку задачи.
Test Actual Cost	Фактические затраты тестера	Время (в часах), которое реально было потрачено тестером на работу с данным элементом.

Несколько замечаний.

Один и тот же элемент может многократно переходить из одного состояния в другое (например, задача из состояния «активна» в состояние «разрешена» и обратно). В базе сохраняются время и автор последнего изменения.

Аналогично, в разное время с элементом могут работать разные люди. В базе указаны имена, действительные в настоящее время.

Названия ряда полей явно неудачны. Прежде всего, это пары Estimated cost / Test estimated cost и Actual cost / Test actual cost. Во-первых, термин cost не вызывает ассоциаций с затратами времени. Во-вторых, в первом имени пары не хватает видового уточнения develop. Без него имя поля имеет вид родового имени.

В качестве единицы измерения времени TFS предлагает час. Любой промежуток времени меньше часа должен округляться вверх до целого часа. Такая единица представляется слишком грубой и осложняет анализ хода работы. Для новых проектов можно было бы рекомендовать использовать меньшую единицу (например, децичас).

Использование информационной базы TFS для управления программным проектом

Team Foundation Server позволяет выгрузить всю информацию о задачах и ошибках в Excel-файл. На основе этих данных можно получить информацию, которая характеризует ход разработки программного проекта:

1. **Разница фактического и запланированного времени.** На начальном этапе планирования создают итерации и соответствующие задачи, каждую задачу оценивают по времени, указав в рабочем элементе плановые затраты разработчика. После разработки рабочего элемента разработчик указывает фактические затраты разработки. Аналогично и при тестировании. Когда тестеровщик берет на тестирование задачу или ошибку, то перед этим необходимо указать плановые затраты, а после тестирования нужно указать фактические затраты. Разница между плановым и фактическим временем = плановые затраты – фактические затраты (рис.5).

Номер	Тип рабочего элемента	Плановые затраты разработчика	Фактические затраты разработчика	Разница между план.и факт.временем разработчика	Плановые затраты тестера	Фактические затраты тестера	Разница между план.и факт.временем тестера
1265789	Task	15	11	4	4	6	-2
1237410	Defect	2	2	0	1	1	0
1373842	Defect	3	2	1	1	1	0
1247444	Task	5	3	2	2	1	1
1366719	Defect	1	5	-4	1	1	0
1336660	Defect	1	3	-2	1	3	-2
1340759	Defect	1	1	0	2	1	1
				1			-2

Рисунок 5. Оценка фактического и запланированного времени

Рассчитав разницу между фактическим и запланированным временем можно узнать, насколько точно было запланировано время для задачи или ошибки, проанализировать, какие задачи и в связи, с чем заняли больше или меньше времени, чем было предположено. Если просуммировать разницу между плановым и фактическим временем, то можно понять насколько точно менеджер проекта запланировал время в целом и где допустил неточность в планировании.

2. Оценка ошибок по степени серьезности. При нахождении ошибки фиксируется степень ошибки. Наиболее распространена пятиуровневая система градации критичности дефекта:

- блокирующий (Blocker);
- критический (Critical);
- значительный (High);
- незначительный (Medium);
- тривиальный (Trivial).

На основе степени серьезности ошибок можно получить диаграмму, которая показывает каждую степень серьезности в процентном соотношении по проекту (рис.6).

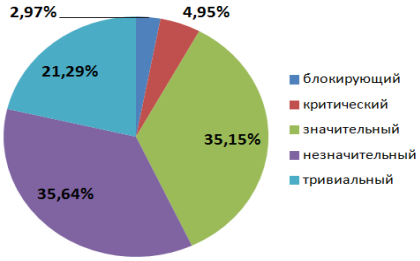

Рисунок 6. Оценка степени ошибок

По степени ошибок можно наглядно увидеть, каких ошибок больше всего и подумать почему. Например, если блокирующих и значительных ошибок больше других, то это может связано с трудной предметной областью, низкой компетентностью разработчиков или плохо написанными постановками.

3. Оценка задач и ошибок по состоянию рабочих элементов. При работе с рабочим элементом указывают его статус (рис.7), по которому можно отследить общий ход работ по проекту.

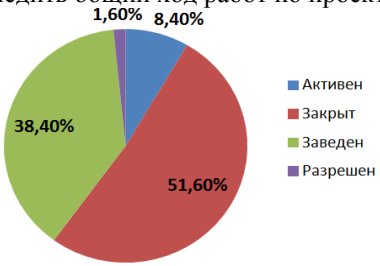

Рисунок 7. Оценка состояния рабочих элементов

По состоянию рабочих элементов можно проследить текущее состояние хода разработки. Если все рабочие элементы будут в состоянии «Закрыт», то это будет означать, что релиз готов к выпуску. Если же не все рабочие элементы закрыты, то менеджер проекта может проанализировать текущее состояние проекта (будет ли готов релиз в срок или нет) и принять необходимые меры (например, в случае нехватки времени перенести задачи на следующий релиз, увеличить число разработчиков).

4. Отслеживание работы каждого сотрудника. Для того чтобы узнать, какой сотрудник, сколько и каких дефектов создал, можно построить сводную таблицу на основе таких полей, как номер задачи или ошибки, тип рабочего элемента, кем создано, степень ошибки. На рис.

8 приведен пример отображения количества создания ошибок по каждому сотруднику в зависимости от степени серьезности ошибок (рис.8).

Названия строк	Blocking	Critical	High	Medium	Trivial	Общий итог
Dracheva, Kseniya V.				1		1
Gabov, Roman E.	1	2	11	19	7	40
Gorbunova, Veronika R.		2	5	2		9
Mysova, Mariya Yu.	4	4	22	15	4	49
Popov, Oleg A.		1	32	35	33	101
Shipilovskih, Christina V.		1				1
Smirnova, Yuliya I.			1			1
Uglitskih, Ilya A.	1					1
Общий итог	**6**	**10**	**71**	**72**	**44**	**203**

Рисунок 8. Количество созданных ошибок каждым сотрудником в зависимости от степени серьезности ошибки

По данному параметру можно узнать эффективность работы сотрудников, насколько хорошо они работает. Причем небольшое количество созданных дефектов сотрудником может быть связано со степенью ошибки.

Аналогичным образом можно посмотреть, как много ошибок или тасков закрыл, реализовал или активировал тот или иной сотрудник по определенному параметру (по степени серьезности, по состоянию рабочего элемента и т.д).

5. Отслеживание хода работ по проекту. Для начала нужно создать сводную таблицу, которая бы отражала количество ошибок в зависимости от состояния в промежутки времени. На основе этого можно вычислить нарастающим итогом количество созданных и закрытых ошибок. Количество созданных ошибок на текущую дату – это все ошибки, созданные вплоть до текущего дня = сумма созданных ошибок в прошлом + количество созданных ошибок сегодня. Аналогично считается количество закрытых ошибок (рис.9).

T26 =T25+Q26

	N	O	P	Q	R	S	T	U
24	Дата	Active	Closed	Proposed	Resolved	Общий итог	Создано ошибок	Закрыто ошибок
25	11.11.2014			1		1	1	0
26	13.11.2014			1		1	2	0
27	28.11.2014			3		3	5	0
28	01.12.2014	2		1		3	7	0
29	03.12.2014	1		2	2	5	8	0
30	05.12.2014	1	2	2	1	6	10	2
31	07.12.2014	2	3	2		7	12	5

Рисунок 9. Количество созданных и закрытых ошибок по проекту

На основе данной таблицы можно построить график, который отражает динамику закрытия и создания ошибок (рис.10).

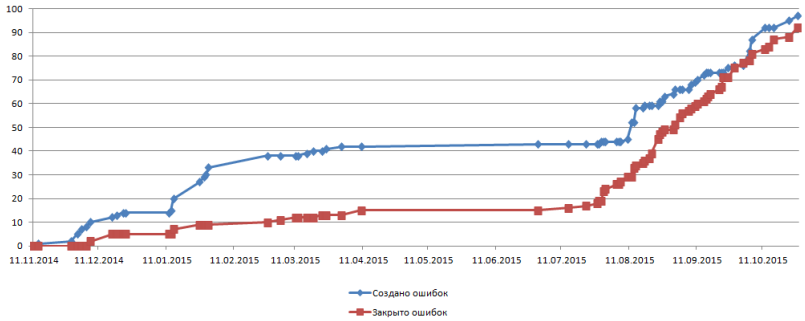

Рисунок 10. Количество созданных и закрытых ошибок по проекту

С помощью графика наглядно видно, когда активно начали работать на проекте, когда можно сдавать проект (т.е. количество закрытых ошибок должно быть равно количеству созданных ошибок).

Таким образом, с помощью на основе информации TFS можно получать необходимые данные для отслеживания хода разработки и работы сотрудников.

Заключение

В статье описана модель программного проекта, использующего информационную базу Team Foundation Server для регистрации таких проектных артефактов, как релиз, задача, ошибка. Описан порядок работы с информационной базой в ходе реализации проекта и возможность использования хранящихся в ней данных для управления процессом реализации программного проекта, оценки успешности этого процесса и выявления возникающих проблем, как во время проекта, так и по его окончании (на этапе постпроектного анализа).

Библиографический список

1. Project management is easy and efficient with Wrike // Wrike [Электронный ресурс] [Режим доступа: https://www.wrike.com/tour/] [Проверено 10.11.2016]

2. Мейер Д., Тейлор Д., Макман А., Бансод П., Кевин Д.. Командная разработка с использованием Visual Studio Team Foundation Server. М.: НОИ Интуит, 2016.
3. Степанченко И. В. Методы тестирования программного обеспечения – учебное пособие. – Волгоград: ВолгГТУ, 2006.
4. Использование рабочих элементов для управления проектом // Microsoft [Электронный ресурс] [Режим доступа: https://msdn.microsoft.com/ru-ru/library/hh409275.aspx] [Проверено 09.11.2016]