JIRA - Все, что пожелаешь, Хозяин.

JIRA: Все, что пожелаешь, Хозяин. Часть 1

Управление проектами - не простая наука, для этого нужны знания (и еще больше опыт) во множестве сфер связанных и с социальными, техническими, экономическими технологиями. Я ставлю перед собой более простую задачу: рассказать об "управлении желаниями" и программными продуктами, которые могут вам помочь вести учет пожеланий клиентов, контролировать их выполнение в сфере разработки ПО.

На рынке множество программных продуктов, которые позиционируют себя как "Project Management", "Issue Tracking", "Bug Tracking", "Customer Relationship Management". Взаимопроникновение этих продуктов и прагматичный подход "нашего человека" умеющего использовать для ведения проекта не только специализированный софт, но и все что угодно (я видел "чудеса", когда для общения между заказчиком и клиентом использовался обычный форум, и то какие проблемы приходилось решать чтобы построить график работ и следить за его соблюдением). Так вот все эти факторы часто приводят к тому, что люди часто путают эти приведенные выше понятия, пытаются решать некоторым программным продуктом несвойственные для него задачи. Хотя все приведенные выше категории ПО имеют общие черты, но "дьявол, как известно, кроется в мелочах". И какое бы определение и требования для системы управления проектом я не сформулировал. Всегда найдется человек (точнее предприятие со своим бизнес-процессом), который скажет: "нет, у нас не так". Я буду рассказывать об небольшой части управления проектами в области ПО: ведение учета ошибок и пожеланий клиентов. Если разработкой программы занимается не один человек, а команда, то важнейшей задачей становится обеспечение коммуникации между всеми участниками. Сценария, когда босс (заказчик) водит пальцем по мятому листку бумаги и цедит из себя, что-то вроде, хочу такое, ну ... этакое, ну ты, в общем, понял. Затем программист допридумывает себе задание, делает его в меру своего скромного представления, что же от него хотели. Тестировщики, в свою очередь, сравнивают "непонятно что" с "непонятно каким" эталоном, а затем, набросав пару строчек на листке бумаги, клеят его на монитор программистам и с чувством исполненного долга идут играть в counter-strike. Представили картинку, а, может, ... узнали что-то родное?

Нам нужен инструмент, с помощью которого мы определим перечень задач, которые должны быть выполнены. Нам нужен инструмент, где будет вестись перечень уже сделанных работ и затраченных на это ресурсов. Инструмент, в котором будет вестись журнал ошибок выявленных в ходе тестирования, и то какие из этих ошибок были исправлены. И, приходя утром на работу, каждый из участников проекта сможет просмотреть перечень назначенных дел (пришедших от заказчиков или тестеров), а, уходя вечером домой, выставит "галочки", мол, сделано. Давайте посмотрим, какие программные продукты помогут управлять Жизненным Циклом ПО. И чем нам поможет JIRA, не зря же я вынес ее в заголовок статьи.

Наиболее простым (и наиболее часто используемым) инструментом в разработке ПО, является "Bug Tracking". Они представляют собой базу данных, каждая запись в которой – сообщение об выявленной в ходе тестирования/эксплуатации ошибке. Пользоваться такими продуктами могут как ваши штатные тестировщики, так и бета-тестеры и даже конечные клиенты. Они заходят на специальную страницу вашего сайта, выбирают программный продукт, для которого они нашли ошибку, указывают ее категорию, описывают ситуацию, в которой возникла ошибка, версию О.С., программы, прикладывают файлы со скриншотами и т.д. Приходя утром на работу, вы просматриваете перечень добавленных за ночь сообщений об ошибках, определяете из них те, которые рапортуют об устраненных проблемах (может быть, клиент сообщает об ошибке, которая характерна для старой версии программы и ее нужно просто обновить). Возможно, сообщения об ошибках дублируются и вы уже решали ранее эту проблему. Отфильтровав ошибки, вы выставляете им приоритет и беретесь за работу. С тем чтобы к концу рабочего дня выложить в туже "Bug Tracking" систему описание способа решения проблемы. По крайней мере, вы всегда будете четко знать, что вы сделали за рабочий день и что осталось сделать завтра.... Не зря же говорят, что "след самых бледных чернил на бумаге ярче, чем самая крепкая память".

JIRA не только система учета ошибок, это система учета пожелания "Issue Tracking", у нее веб-интефейс, система дополнений и интеграция с другими программами поддерживающими Жизненный Цикл разработки ПО. Хорошая система "управления желаниями" должна уметь решать задачу согласования видений проекта у различных его участников. К примеру, заказчик проекта оперирует такими понятиями, как задача, степень ее готовности (в процентах), затраченные человеко-часы и деньги. С другой стороны для программистов важны другие критерии деления, например, сделать форму с двумя кнопками, сделать отчет с перечнем всех товаров, ... Подобное детальное представление работ мало интересует заказчика – он не должен вникать в то, что делается, ему нужен конечный продукт, программисты – наоборот. И хорошая система "управления желаниями" должна уметь сочетать эти две стороны. Кроме того важным является наличие средств интеграции со средствами автоматизированного тестирования, управления версиями файлов.

Я покажу как установить, настроить и использовать JIRA, тем кто с ней еще не сталкивался. За продвинутыми трюками и хаками прошу обращаться на сайт разработчиков JIRA (http://www.atlassian.com/software/jira/).

JIRA – веб-приложение, написанное на java. Значит для запуска ее нам нужен java хостинг. Есть множество веб-серверов способных запускать java-приложения (tomcat, jetty, resin ...). По адресу http://www.atlassian.com/software/jira/docs/latest/servers/index.html есть инструкции, рассказывающие как настроить jira для различных серверов. После того как вы скачали и разархивировали JIRA необходимо отредактировать файл с параметрами подключения к базе данных. По-умолчанию jira работает с базой данных hsqldb, хотя не обязательно, но рекомендуется ее заменить для повышения скорости работы. для этого я создал базу данных jiradb, пользователя jirauser (с паролем jirapass) и дал этому пользователю полные права доступа к базе jiradb:

```
CREATE DATABASE jiradb DEFAULT character SET utf8;

GRANT ALL ON jiradb.* TO jirauser IDENTIFIED BY 'jirapass';
```

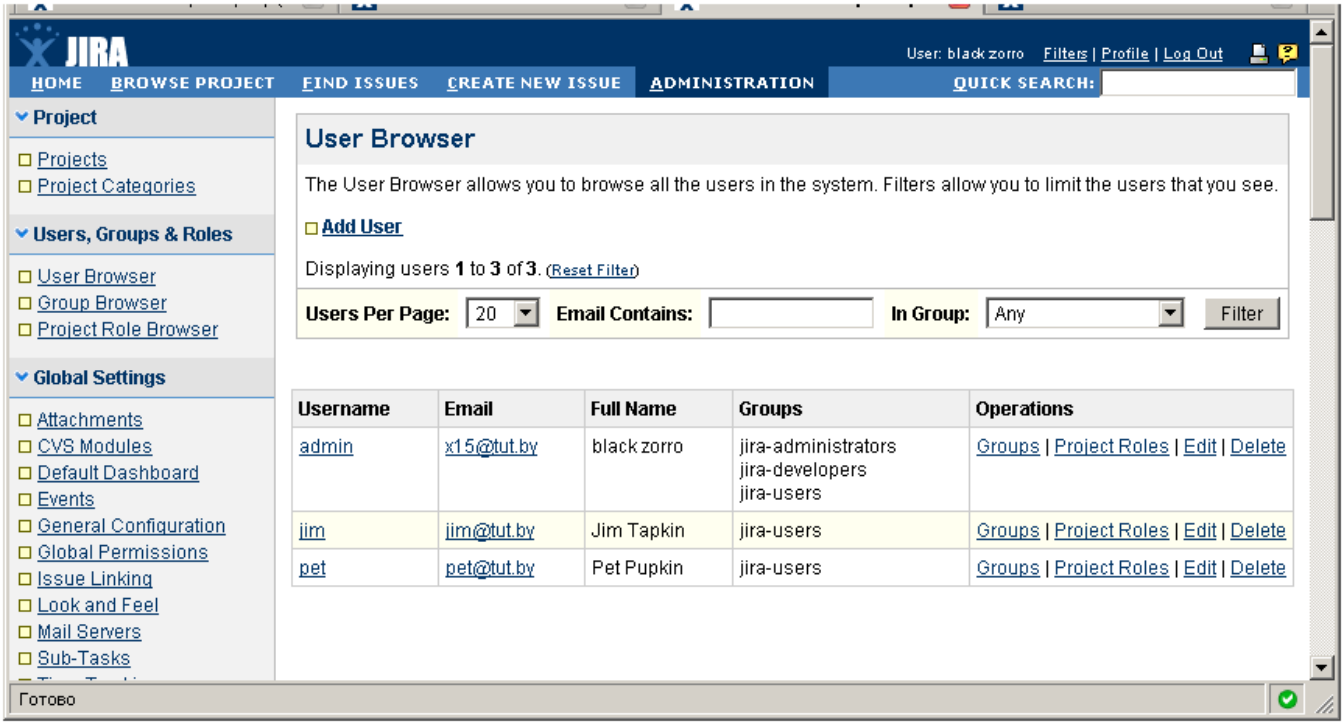
Теперь необходимо внести правки в файл /edit-webapp/WEB-INF/classes/entityengine.xml, найдите там строку field-type-name="hsql" и замените ее на field-type-name="mysql", также нужно удалить строку schema-name="PUBLIC". Затем вы запускаете процесс компиляции JIRA с помощью файла build.bat (он находится в корне распакованного дистрибутива jira). После чего в папке dist-tomcat/tomcat-6 будет сформирован файл atlassian-jira-НОМЕР-ВЕРСИИ.war его можно скопировать в папку webapps инсталляции tomcat, а можно не копировать (в дальнейшем примере при создании Context-а явно указывается путь к архиву). Осталось только настроить параметры подключения к БД и указать путь к веб-приложению. Для этого я скопировал файл jira.xml в папку conf/Catalina/localhost/ и отредактировал его. Прежде всего, обратите внимание на корневой тег Context и его атрибут docBase, здесь мы должны указать путь к созданному на прошлом шаге war-файлу. Значения атрибутов username и password необходимо поменять на имя и пароль созданной ранее учетной записи. Значение атрибута driverClassName меняется так чтобы указывать на имя драйвера для работы с базой данных mysql (com.mysql.jdbc.Driver), а атрибут url меняется так, чтобы указать на подключение к базе данных jiradb, например:

```
url="jdbc:mysql://localhost/jiradb?autoReconnect=true&useUnicode=true&characterEncoding=UTF8"
```

Настоятельно рекомендуется (а в случаях, когда при вводе текста на русском языке в JIRA появляются "кракозюбры", то обязательно) в файл conf/server.xml добавить следующую строку useBodyEncodingForURI="true" для тега Connector.
Раз мы работаем с базой mysql, то необходимо файл с драйвером (http://dev.mysql.com/downloads/connector/j/3.1.html) скопировать в папку WEB-INF/lib приложения. Еще с сайта http://www.atlassian.com/software/jira/docs/servers/jars/v1/jira-jars-tomcat6.zip вы должны скачать архив с нужными для работы JIRA над tomcat библиотеками и скопировать их в папку tomcat/lib.

Я пропустил в приведенной выше инструкции некоторые детали связанные с повышением производительности JIRA: они не столь важны для первого знакомства и в последующем вы сможете выполнить эти процедуры с помощью инструкций на сайте JIRA. Теперь осталось только запустить tomcat и открыть в браузере адрес http://localhost:8080/jira. Если процедура установки прошла без ошибок, то вы увидите мастера настройки JIRA, вам нужно будет пройти через три шага установки. На первом из них выбирается язык интерфейса (среди вариантов есть и русский, но его лучше не использовать – перевод не полный), название вашей компании, пути к трем каталогам, где будут храниться нужные для работы JIRA файлы и резервные копии. Там же нужно указать режим работы JIRA: public или private. Отличаются эти режимы тем, что в режиме public любой пользователь может зарегистрироваться и оставлять комментарии, в противном случае регистрация нового пользователя должна быть выполнена только администратором. Да ... самое главное, нужно ввести лицензионный ключ. Есть три версии JIRA: Standard, Professional, Enterprise, и хотя их стоимость зашкаливает за несколько тысяч "$", но JIRA отрабатывает вложенные в нее деньги до последней копейки, подтверждением этого будет многотысячный перечень организаций использующих JIRA и смежные с ним продукты (Confluence, Bamboo, FishEye ...). К тому же всегда можно получить пробную версию на 30 дней (сайтеhttp://www.atlassian.com). На второй форме процедуры установки вам нужно придумать имя и пароль для администратора. Третья форма служит для указания настроек почтового сервера. Все указанные настройки можно изменить в будущем.

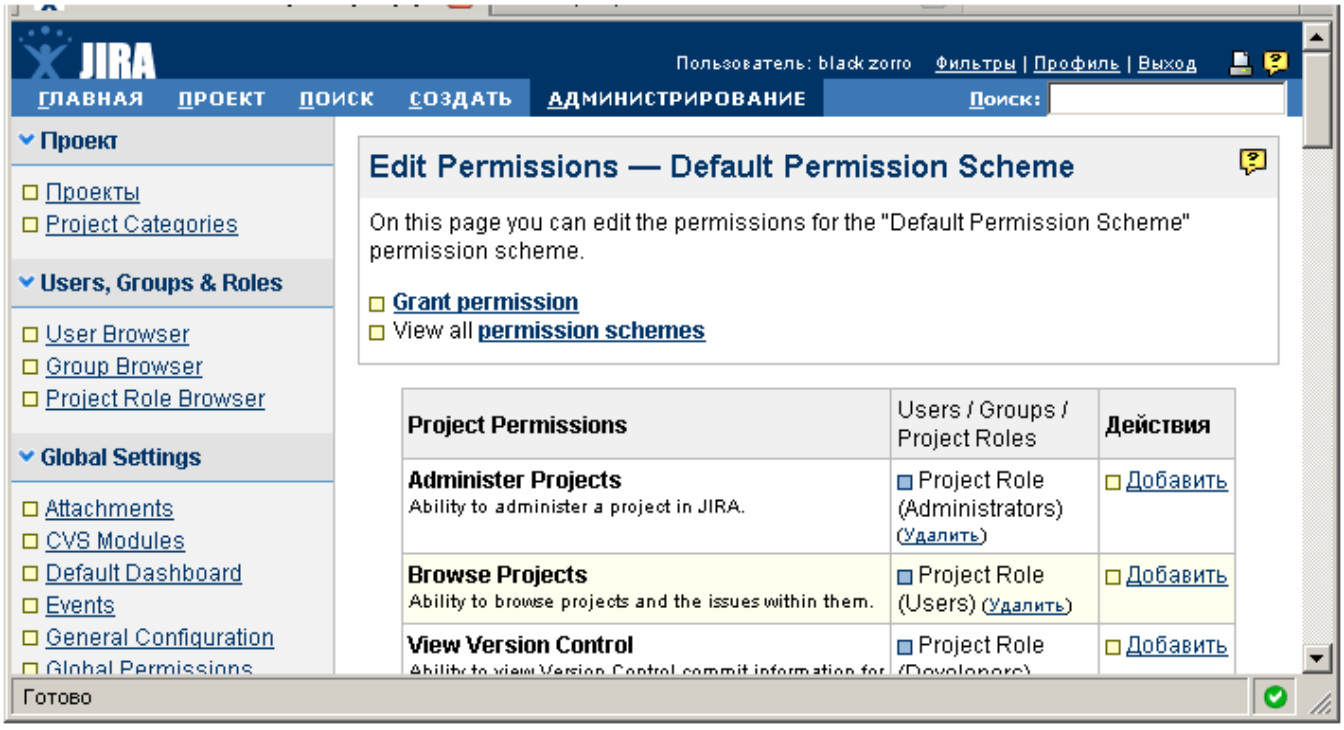
Объекты, которыми может управлять JIRA – это проекты, задачи, подзадачи. Прежде чем мы начнем создавать проект необходимо разобраться в терминологии JIRA, политике безопасности. Для этого завершив процедуру установки, вы возвращаетесь на первую страницу jira, и для входа в систему указываете имя и пароль администратора. Либо, воспользовавшись ссылкой signup, регистрируете нового пользователя (учтите, что создание новых проектов, равно как и задачи настройки JIRA может выполнить только администратор). Главное меню JIRA состоит из следующих пунктов: "HOME", "BROWSE PROJECTS", "FIND ISSUES", "CREATE NEW ISSUE", для администратора добавляется еще пункт "ADMINISTRATION", именно сюда вам и нужно зайти. Все функции администрирования перечислены в колонке слева и делятся на категории. Во-первых: управление учетными записями, группами пользователей и ролями. Затем изменение глобальных настроек JIRA; управление сценариями или схемами выполнения некоторых задач; средства для настройки и конструирования по своему вкусу "задач" из которых состоят проекты; функции экспорта и импорта информации и многое другое. Начнем мы с простого: разберемся с правами доступа. Политика безопасности JIRA оперирует тремя понятиями: пользователь, группа, роль. Пользователь – это человек, который может входить внутрь JIRA и выполнять какую-то работу, для того чтобы легче было назначить нескольким пользователям сходные права, мы объединяем их в группу, и затем привязываем права (роли) к этой группе (роль может быть назначена и непосредственно отдельной учетной записи). Для просмотра списка пользователей используйте меню "Users, Groups & Roles -> User Browser".



Так вы увидите табличку с перечнем пользователей, для каждого из них указано его ФИО, адрес почты (этот адрес важен, т.к. мы можем настроить рассылку извещений разработчикам проекта, дабы никто из них не сказал "я не сделал работу т.к. не знал, что мне нужно что-то делать ..."). Если вам кажется, что ФИО и email не слишком большой объем информации о человеке и хотите указать для пользователя его телефон, домашний адрес ..., то вам понравится "Свойства" (Properties). Так если вы на закладке "User Browser" нажмете на ссылку с именем пользователя, то откроется окно профиля, где можно изменить имя, пароль, членство в группах, роли и назначить человеку произвольный список свойств. Каждое свойство – это пара вида "ключ и значение", например, "phone = 12345".

Кроме того, JIRA отслеживает все операции с пользователями, и если вы захотите удалить кого-то, кому назначена "незавершенная работа", то операция удаления будет запрещена.

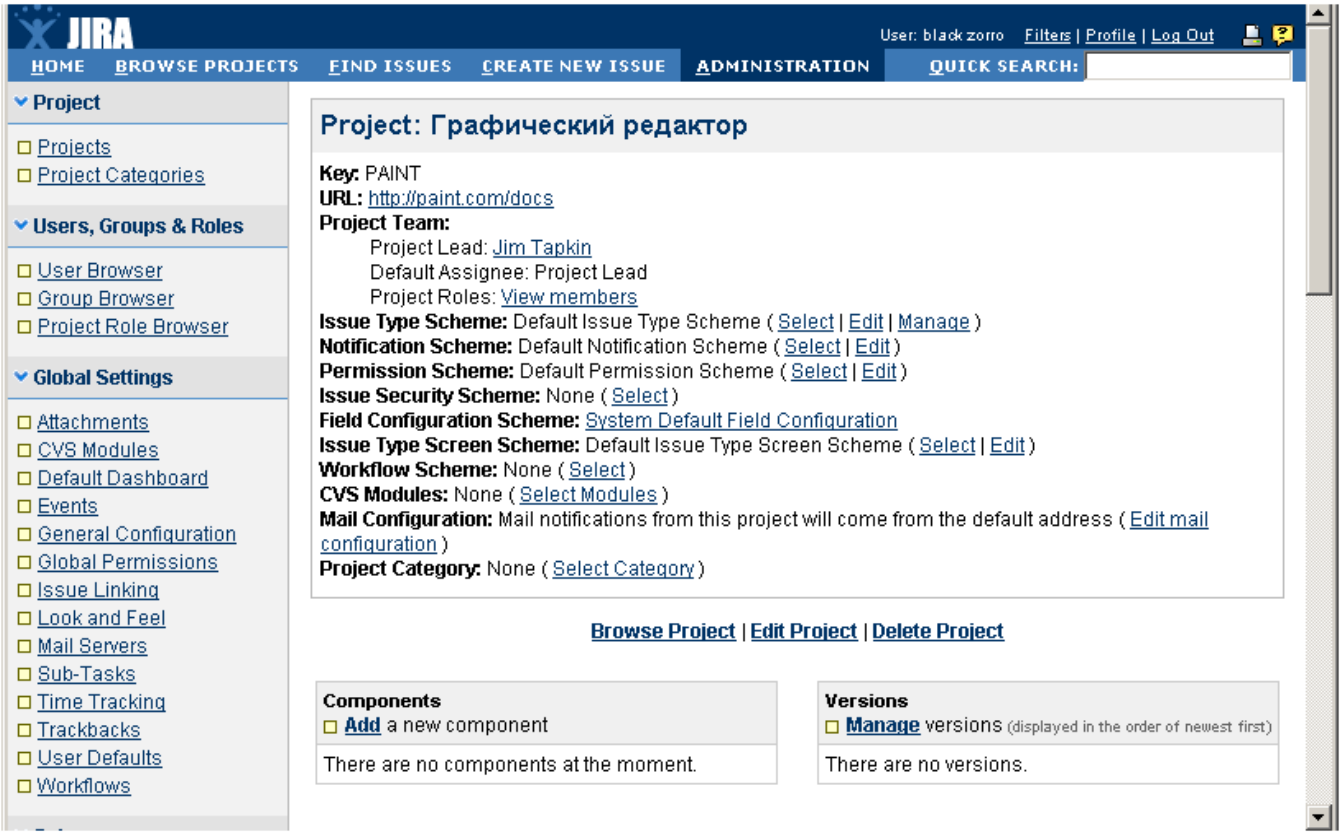
Для каждой учетной записи перечисляется список групп, в которые включен данный пользователь, и все, что было перечислено выше, мы можем изменить с помощью кнопок-ссылок "Groups", "Project Roles", "Edit", "Delete". Для добавления нового пользователя используйте меню ссылку "Add User". Никаких сложностей в создании и редактировании пользователей нет, поэтому переходим к созданию группы. Для этого в главном меню JIRA выбираем пункт "Users, Groups & Roles -> Group Browser" и видим табличку с перечнем групп (для каждой группы указывается сколько пользователей вошло в ее состав и можно просмотреть их поименно). По-умолчанию есть три группы: "jira-administrators", "jira-developers" и "jira-users", но вы можете создать и собственную группу. Все создаваемые пользователи изначально попадают в группу "jira-users". В группу "jira-developers" должны входить те пользователи, которые имеют право заниматься работой (брать на выполнение задачи проекта). Ожидаемо, что один пользователь может входить сразу в несколько групп. Также есть понятие "автоматического членства", когда каждый создаваемый пользователь автоматически будет входить в состав некоторой группы. Теперь разберемся с правами доступа (ролями), для этого переходим по меню "Users, Groups & Roles -> Project Role Browser" и видим привычную уже картинку с табличкой в которой перечислены роли. Есть три предопределенные роли: administrators, developers, users, которые назначены "почти-одноименным" группам (список ролей также не является фиксированным и вы можете создать собственную роль). И тут возникает вопрос, а что такое роли и как они отображаются на конкретные действия, которые можно или нельзя выполнять в JIRA. Дело в том, что система безопасности JIRA вводит еще одно понятие "разрешение" (Permission) и это те самые элементарные "кубики" из которых и складываются роли. Разрешения делаются на две категории: Глобальные и Проектные. Глобальные привилегии разрешают работать с JIRA в целом (например, входить в систему), проектные же привязаны к конкретному проекту (например, просмотреть список задач, редактировать их). Глобальные Разрешения назначаются конкретным группам пользователей, а Проектные Разрешения можно назначить не только группам пользователей, но их Проектным Ролям, конкретным пользователям. Проектные Роли и группы схожи, и те и другие являются средством привязать пользователя к Разрешениям. Вот только группы являются глобальными (Вася входит в группу программистов), а Проектные Роли специфичны для каждого из проектов (Вася в проекте "калькулятор" играет роль программиста, а в проекте "блокнот" играет роль "тестировщика"). Проектные Роли" являются общими для всех проектов и в Проектную Роль могут быть назначены пользователи и группы по-умолчанию, так что когда создается новый проект, то перечисленные люди/группы автоматически получат назначение (функции которые они должны выполнять) в новом проекте. Но это еще не все, чтобы окончательно вас запутать ввели понятие "Permission Scheme". Предположим, что если у вас есть команда разработчиков (с четкой структурой и разделением ролей), перед началом разработки проекта вы сгруппировали программистов в группы, назначили им роли. Когда проект был завершен, команда не была расформирована и в том же составе и с теми же функциональными обязанностями должна взяться за еще один проект. Так вот, для того чтобы каждый раз заново не назначать участникам проекта их роли, мы можем создать одну Permission Schema (таблицу в которой хранятся связи между пользователи-роли-группы) и назначить ее новому проекту. Для примера вызовите меню "Schemas -> Permission Schemas", там вы увидите таблицу с перечнем схем и того, к каким проектам они привязаны (пока список пуст). Нажав на ссылку "Permissions" вы попадете на страницу с перечнем всевозможных действий, которые можно выполнить над проектом, для каждого действия указаны пользователи и группы и роли, которые ими наделены.



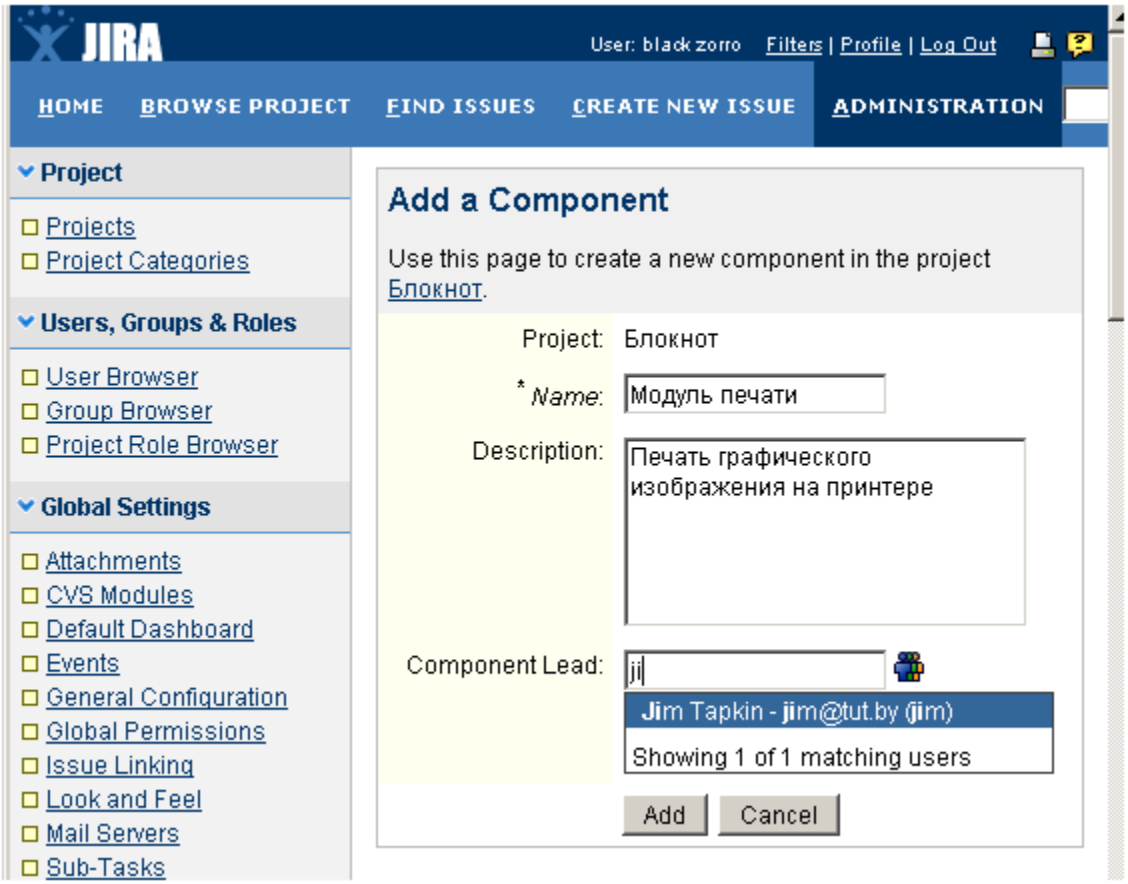
JIRA: Все, что пожелаешь, Хозяин. Часть 2

Я продолжаю рассказ об JIRA. Управление процессом разработки ПО, ведение журнала пожеланий, задач, обнаруженных багов – все это сфера применения JIRA. В прошлый раз я показал, как установить JIRA у себя на компьютере, рассказал о политике безопасности. Теперь самое время создать проект, разобраться в его составляющих, попробовать пройтись по шагам Жизненного Цикла.

Для создания проекта вам нужно войти в JIRA от имени администратора, затем перейти на закладку "ADMINISTRATION", затем по ссылке "Add Project". В появившемся окне вам нужно указать основные характеристики проекта: Name (Название проекта), здесь можно указать произвольный текст на русском языке. Затем идет Key(Код проекта) - это несколько букв на английском, например, для проекта калькулятор, его код может быть равен CALC. Если у вас есть уже подготовленная документация по проекту (его описание, Т.З.), то в графе URL вы указываете адрес сайта, где эта документация опубликована. Следующий шаг – указание Project Lead – ведущего разработчика проекта (в качестве project lead не может выступать пользователь не входящий в состав группы jira-developers или jira-administrators). Затем идет Default Assignee – по-умолчанию это тот же человек что и Project Lead. Т.к. проект в основе своей состоит из некоторого количества задач, то Default Assignee – это тот человек, который будет отвечать за выполнение каждой из поставленных задач (естественно, что в последующем он сможет делегировать свои обязанности кому-нибудь другому). Следующий параметр, который нужно указать при создании проекта – это Description (описание проекта). Затем идет Notification Scheme - это правила, по которым при изменении проекта (добавление новой задачи, завершение работы над ней) будут посылаться извещения пользователям JIRA. Также нужно указать схему прав доступа (Permission Scheme), она определяет, какой пользователь и что может делать в JIRA с этим проектом (подробнее о политике безопасности и схемах я писал в прошлой статье). Последняя характеристика проекта - Issue Security Scheme (схема безопасности для каждой из задач). Это специфическая функция только для версии JIRA Enterprise. Идея в том, что пользователи/группы/роли проекта делятся на уровни безопасности, затем при добавлении в JIRA новой задачи ей будет назначен некоторый уровень защиты. По-сути, Issue Security Scheme – неплохой способ защитить информацию, если организация разместила JIRA не в локальной сети, а в internet, так чтобы пользователи некоторого продукта могли добавлять свои пожелания, сообщать об выявленных ошибках. Важно, что одновременно эту установку JIRA используют и в самой организации, а значит что не все задачи проекта (те которые создают разработчики внутри самой организации) должны быть доступны для обычных пользователей internet. Пример того, как выглядит карточка проекта, показан ниже на рисунке:



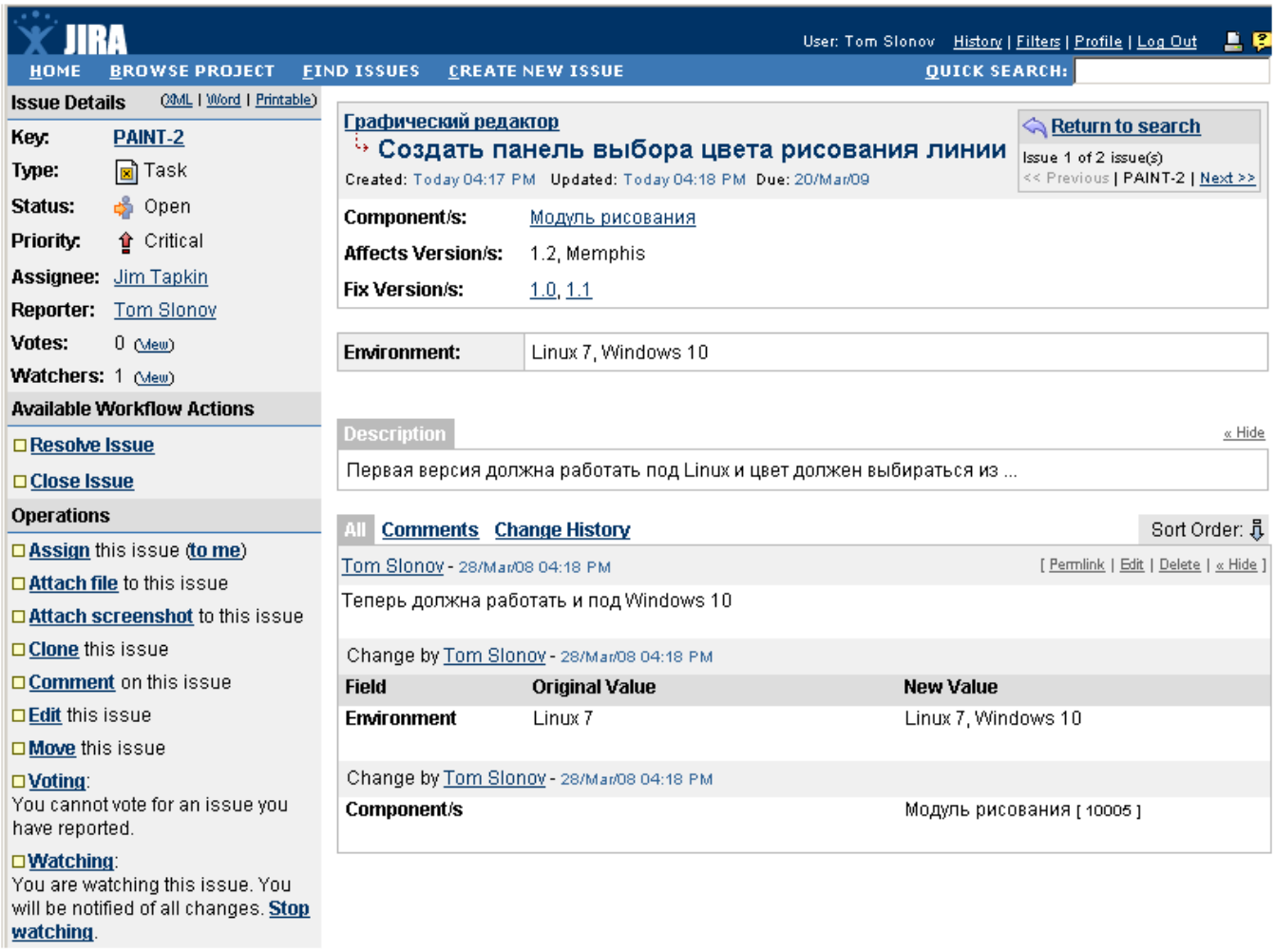
На этой карточке находится несколько полезных ссылок для "донастройки" проекта. Прежде всего, Project Category – создаваемые проекты можно группировать по категориям. Создать категорию можно на закладке "ADMINISTRATION", раздел "Project categories". В дальнейшем, если перейти на закладку "BROWSE PROJECTS", то увидите список проектов, сгруппированный по назначенным им категориям. Следующая опция в карточке проекта - CVS Modules, она служит для интеграции JIRA с системой управления версиями файлов (изначально в JIRA встроен модуль CVS, однако есть плагины для интеграции с SVN, Perforce). Следующая опция проекта – Workflow Scheme, ее назначение – настроить этапы, через которые может проходить решаемая задача (подробнее об Workflow Scheme чуть позже). Каждая задача, из которых состоит проект, имеет набор характеристик, например, название, категория (ошибка, или новая функциональность) ..., таких характеристик (в терминологии JIRA полей) много. Но не все ситуации можно заранее предусмотреть и возможно для некоторого проекта вам потребуется, чтобы к каждому из заданий было добавлено еще одно поле характеристики (например, признак того, что задание должно быть выполнено только в полночь пятницы 13 числа). Так вот подобная фунциональность обеспечивается с помощью Field Configuration Scheme. Помните только, что поведение JIRA зависят от того какая у вас версия: Standard, Professional, Enterprise (в младших версиях количество доступных функций меньше).
Создав проект, мы должны наполнить его содержимым, но до создания задач необходимо разбить проект на компоненты и указать версии. Компоненты это части проекта, например, проект графического редактора будет условно состоять из модуля загрузки/сохранения, модуля печати, модуля рисования. Чтобы создать компонент вы на закладке "ADMINISTRATION" переходите по ссылке "Projects", затем в таблице с перечнем проектов, вы выбираете проект, для которого хотите отредактировать его карточку и внизу карточки находится перечень компонент. Для добавления компонента используйте ссылку "Add a new component",



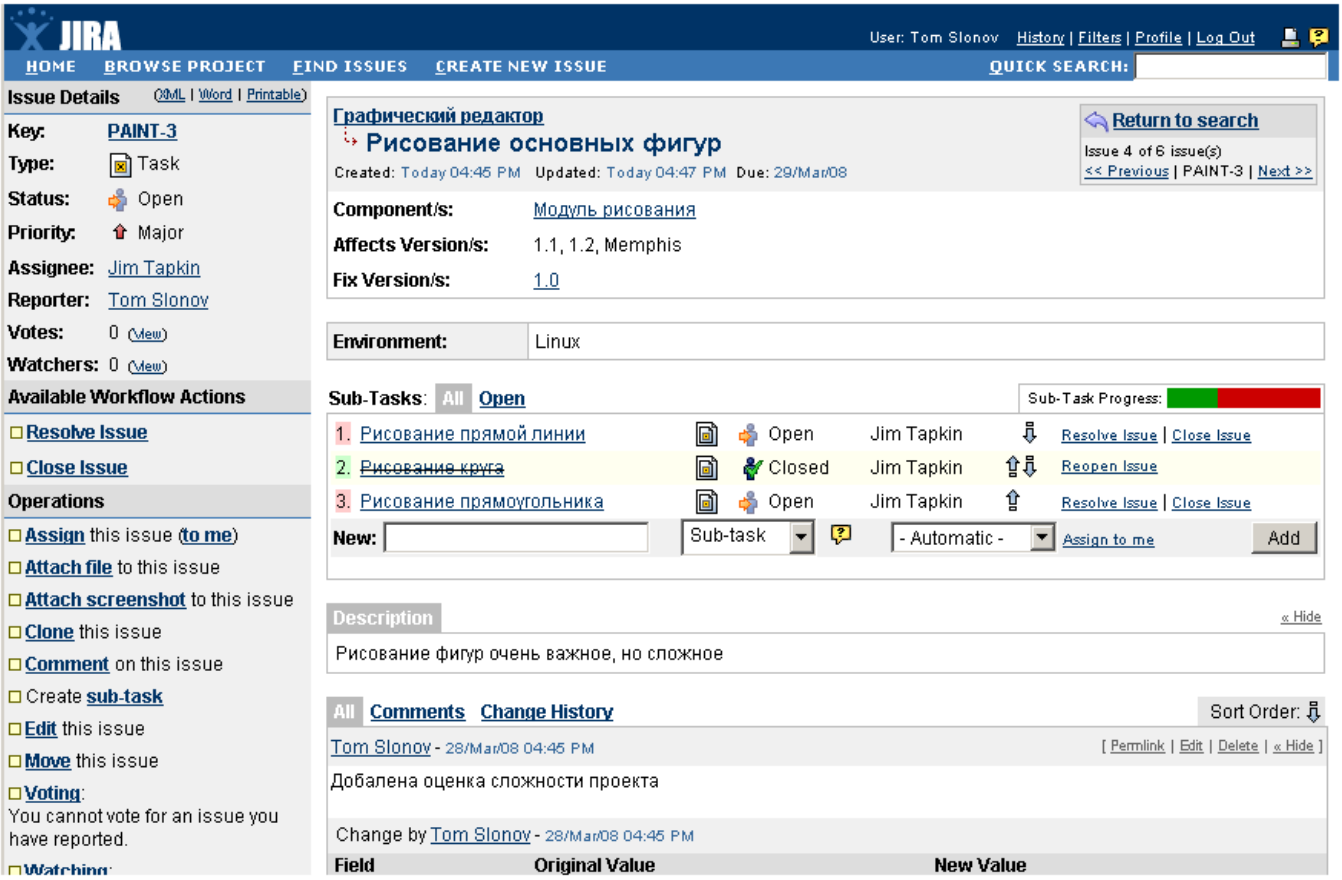
затем вы указываете название компонента, примечание к нему, а также выбираете кто из разработчиков будет за него отвечать (Lead). Создав несколько компонент? вы может выполнить донастройку проекта, и для каждого из компонентов указать лицо, которое будет играть роль Default Assignees (лицо ответственно за выполнение задачи добавленной в проект или его компоненту), для этого используйте ссылку "Select assignees for components". Проект не статическое образование, он развивается во времени, поэтому было введено понятие версий (редактировать версии нужно здесь же, в карточке проекта). При добавлении версии указывается, собственно, ее номер (возможно указать не только цифры, но и более сложные комбинации, например, 1.1.2, или кодовое название Chicago). Кроме номера версии, нужно указать планируемую дату завершения работ над версией (Release Date) и некоторый текст примечания к версии. Если вы будете добавлять вторую и последующие версии, то добавится еще одно поле Schedule, в нем нужно указать номер версии предшествующей добавляемой. Создав версии ими можно управлять, например, менять статус, с Release на Unrelease (версия вышла или в стадии разработки), выполнять слияние версий (в ходе этого выполняется перенос всех запланированных для этой версии задач в другую версию). В случае удаления версии, вас спросят, что делать с задачами назначенными для нее: следует ли эти задачи аннулировать или перенести в другую версию проекта?

Завершив создание проекта самое время перейти к наполнению его задачами. Для этого не обязательно быть администратором проекта: создавать задачи может и пользователь, включенный только в группу jira-users, однако для того, чтобы редактировать задачи, перемещать их ... нужно входить в группу jira-developers. Итак: после регистрации, вы заходите на закладку "CREATE NEW ISSUE", затем выбираете в падающем списке проект для которого хотите добавить задание, и тип задания. Среди предопределенных типов работы Bug, New Feature, Task, Improvement. Этот перечень не фиксированный, и его можно изменить, если в режиме администрирования перейти по ссылке "Issue Types". Возвращаясь к созданию задачи, давайте выберем тип добавляемой задачи как "Task", затем мы попадаем на форму где нужно указать для задачи все ее свойства. Прежде всего, Summary (название задачи), Priority (степень важности задачи). Она может варьироваться в диапазонах Trivial, Minor, Major, Critical, Blocker, соответственно, от минимальной важности до наивысшей и даже критической, такой, что до решения задачи дальнейшая работа над проектом нецелесообразна. Список приоритетов не является фиксированной величиной и его можно изменять (правда мне это ни разу не потребовалось, но все же). Итак, для редактирования перечня приоритетов в режиме администратора зайдите на страницу "Priorities". Возвращаясь к созданию задачи, теперь нужно указать перечень компонентов проекта, к которым привязана добавляемая задача (можно выбрать несколько компонентов). Графа "Affects Versions" служит для выбора тех версий проекта, к которым будет привязана создаваемая задача. А поле "Fix Versions" указывает то, для каких версий данная функциональность уже реализована. К каждой задаче привязан срок ее завершения (поле Due date в карточке задачи). Для детального описания задачи используется поле "Description" (в тексте можно использовать ссылки вида http://site.ru, они автоматически будут преобразованы в теги "a"). Поле Environment должно содержать информацию об окружении (операционная система и ее версия, тип браузера) для которого актуальная задача. Наконец, мы можем указать кто (какой разработчик) будет ответственным за выполнение задачи и планируемые затраты на выполнение работы.

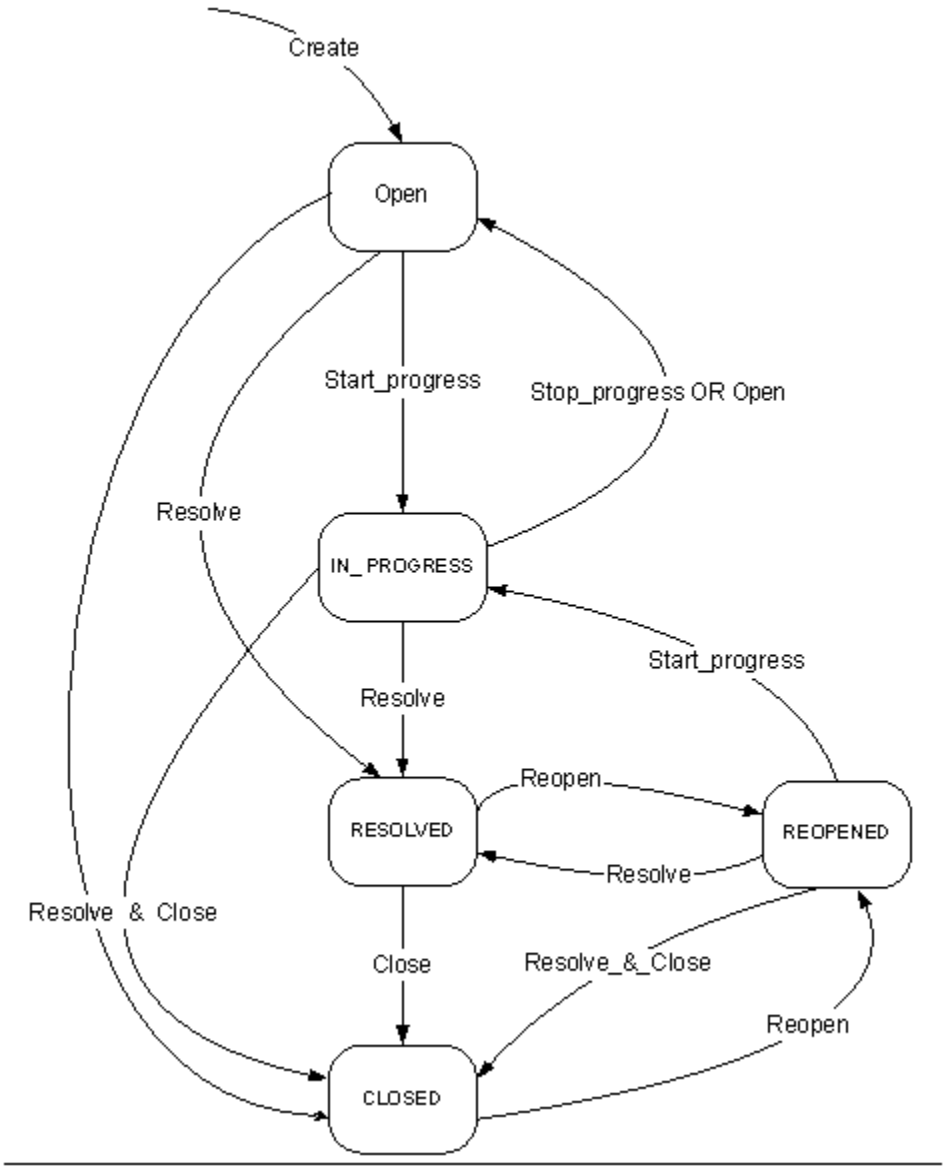
Завершив создание задачи, мы можем ее редактировать (Edit - в левой панели):



JIRA – замечательный инструмент: она ведет учет всех правок выполненных над задачей. К примеру, если я хочу добавить к списку операционных систем еще одну, то нажав на ссылку "Edit", ввожу новое значение в графу "Environment" и должен указать текст примечания к выполненной мною правке. Затем, вернувшись назад, в режим просмотра задачи я вижу под карточкой задачи перечисление всех правок, которые я выполнил (старое значение свойства и новое значение свойства), также вижу примечания к выполненным правкам. К задаче можно приложить файлы: ссылка "Attach file" служит для того, чтобы загрузить с вашего компьютера файл на сервер, а ссылка "Attach Screenshot" пригодится, если вы хотите приложить к задаче пример скриншота (например, вы тестер, нашли ошибку сделали скриншот, но не сохраняете файл с картинкой на диск а хотите чтобы на сервера сразу же отправилось содержимое буфера обмена). Задачу можно перемещать (ссылка Move) в другой проект. Также за задачу можно голосовать. Например, компания adobe создала общедоступную JIRA инсталляцию в internet, так чтобы пользователи продуктов adobe могли участвовать в их разработке. Но очевидно, что если некто, Вася добавил пожелание "чтобы в photoshop была большая красная кнопка", то не факт что эта функция нужна всем. А если разрешить голосование за задачу, то можно определить степень заинтересованности в этой функции у других пользователей. Функция Watching служит для подписки, так чтобы пользователю отправлялись сообщения об изменениях в состоянии задачи. Возможно, что решение некоторой задачи достаточно сложно и поэтому имеет смысл ее разбить на отдельные шаги (подзадания). Вообще-то, такая функциональность в JIRA по-умолчанию отключена, но это можно исправить в режиме администрирования, меню "Sub-tasks", затем ссылка "Enable sub-tasks". Теперь можно вернуться назад в режим добавления задачи, добавим новую задачу, например, рисование геометрических фигур, и теперь мы хотим ее детализировать, выделить подзадачи: рисование линий, прямоугольников, кругов. Для этого я перехожу в режим редактирования задачи "рисование фигур" и вижу что в расположенной слева панели инструментов добавились кнопки "Create Sub-task" и "Convert to Sub-task". Т.е. мы можем добавить к текущей задаче под-задачу или превратить текущую задачу как составную часть чего-то другого. Как будет выглядеть карточка задачи с подзадачами показано на рисунке:



Теперь пора разобраться с понятием Workflow. После создания задачи (отчета об обнаруженной ошибке) мы начинаем работу над ее реализацией (исправлением). Если задача не тривиальна, то на это требуется время, требуется участие других специалистов (например, тестеров) и не факт, что мы с первого раза решим поставленную задачу. Поэтому было введено понятие Workflow, т.е. это набор этапов, через которые проходит задача. Также Workflow - это набор правил перехода между этими состояниями. Это еще и набор действий, которые выполняются в ходе перехода, условий которые могут быть наложены на состояние и переход. В различных версиях JIRA (Standard, Professional, Enterprise) работа с Workflow выполняется по-разному, например, в старших версиях можно создавать собственный Workflow, и даже можно сделать так, чтобы задача одновременно проходила по нескольким Workflow одновременно. Сначала я приведу диаграмму (я взял ее в справке JIRA), где показаны состояния и переходы, реализованные в Workflow по-умолчанию:

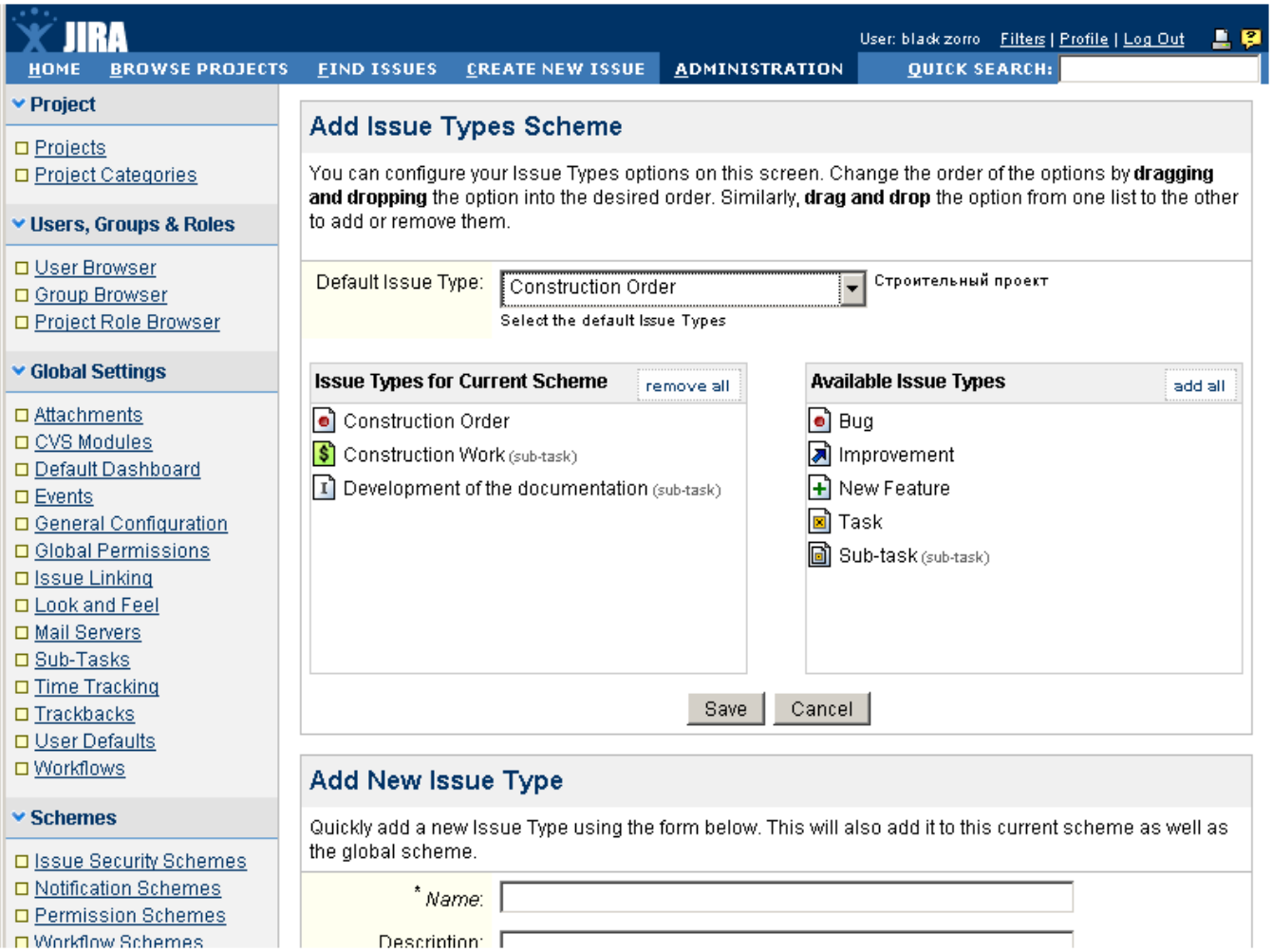

Create
Open
Start_progress
Stop_progress OR Open
Resolve
IN_PROGRESS
Start_progress
Resolve
Reopen
RESOLVED
REOPENED
Resolve
Resolve & Close
Close
Resolve_&_Close
Reopen
CLOSED

Сразу после создания задачи ей назначается статус OPENED (открыта и подлежит решению). Затем некто Вася, приходит утром на работу, смотрит то, какие ему были назначены задания, выбирает одно из них и погружается в работу. Для того чтобы злобный босс не кричал на Васю, чем же он занимается, Вася ставит задаче статус IN_PROGRESS (задача в разработке). Вечером Вася завершает работу и поставив для задачи статус RESOLVED (задача был решена) с чувством выполненного долга уходит домой. Однако, следующим утром когда Вася обнаруживает что за ночь тестеры проверили Васину работу и нашли массу недоделок, поэтому они перевели задачу из состояния RESOLVED в состояние REOPENED (открыта повторно). Так что Васе приходится исправлять свои ошибки и снова рапортовать RESOLVED. Только после того как задача была проверена и все ошибки исправлены, ей меняется статус на CLOSED (задач завершена). Если вы вернетесь в карточку задачи, то обратите внимание на набор ссылок помещенных в группу "Available Workflow Actions", там будут перечислены те действия, которые можно выполнить над текущим состоянием задания. Каждый раз, когда вы меняете это состояние вы должны ввести текст примечания. Механизм создания собственных Workflows очень важен, если модель разработки (взаимодействия между участниками) отличается от приведенной выше, например, если вы хотите применить JIRA не для управления проектом по разработке ПО, а для других видов проектов. Например, вы разрабатываете архитектурные проекты, и у вас есть шаги: "нарисовать дизайн", "начертить схему", "утвердить проект в санэпидемстанции", "утвердить у пожарников", ..., "заплатить бакшиш". Каждый из этих шагов может следовать только после другого шага (нельзя утвердить проект до того как будет создан чертеж). Попробуем создать собственную схему Workflow, для этого в режиме администратора переходим по ссылке "Workflows", там мы увидим описанную выше схему по-умолчанию (ее нельзя редактировать). Внизу страницы вы видите форму добавления новой схемы. Указав название (на латинице) и примечание к схеме, мы переходим на этап наполнения ее содержимым. Прежде всего, необходимо спланировать шаги (состояния обработки запроса). Каждый шаг имеет название и привязанный к нему статус (например, RESOLVED). Перечень статусов не является жестко фиксированным и его можно изменять (режим администрирования, ссылка Statuses). После того как вы создали нужные статусы и привязанные к ним состояния, необходимо создать переходы (Transitions). При создании перехода кроме указания начальной и конечной точки можно (хотя и не обязательно) указать еще и Экраны (например, для перехода от состояния "Дать бакшиш" к состоянию "Утвердить" нужно ввести примечание, что денег не хватило). Экран представляет собой html-форму с некоторым полями (да вы можете создавать собственные формы), которую нужно заполнить для успешного завершения перехода (Transition). Фактически JIRA представляет собой гибкий конструктор позволяющий создать систему управления проектами, выполняемыми работами и подстроить ее под вашу бизнес-модель. Если вам не будет хватить каких-то функций, то можно воспользоваться плагинами или создать собственное расширение возможностей JIRA.

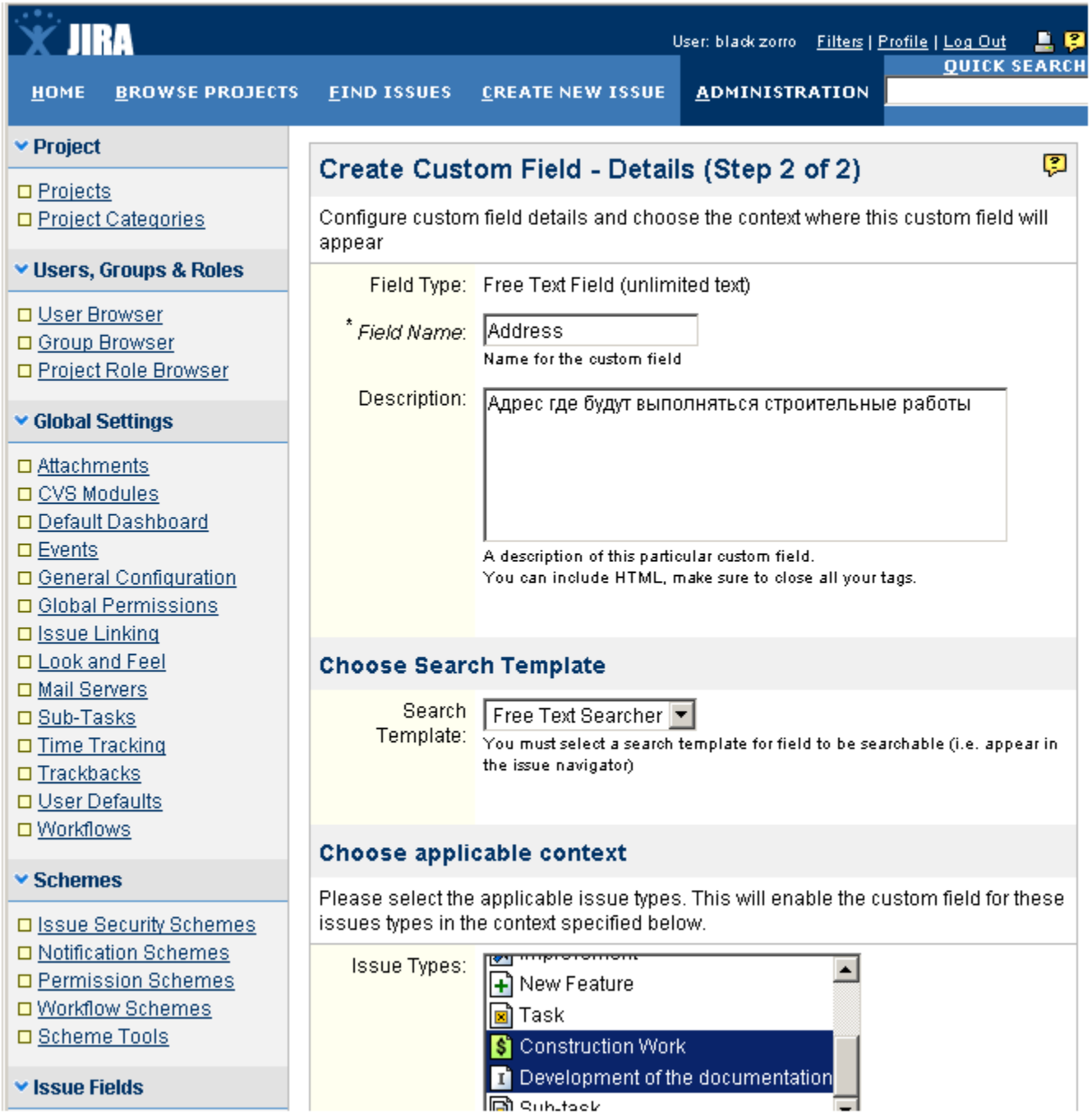
JIRA: Все, что пожелаешь, Хозяин. Часть 3

JIRA – это не просто система управления проектами, журнал учета обнаруженных ошибок, контроля за выполнением работ и средство для общения предприятия с клиентами – это настоящий конструктор, позволяющий создать приложение, путь не вашей мечты, но близкое к тому бизнес-процессу, который есть у вас. Основной акцент я делаю на применение JIRA в сфере разработки ПО. Но с таким же успехом можно применить JIRA и для управления другими видами проектов.

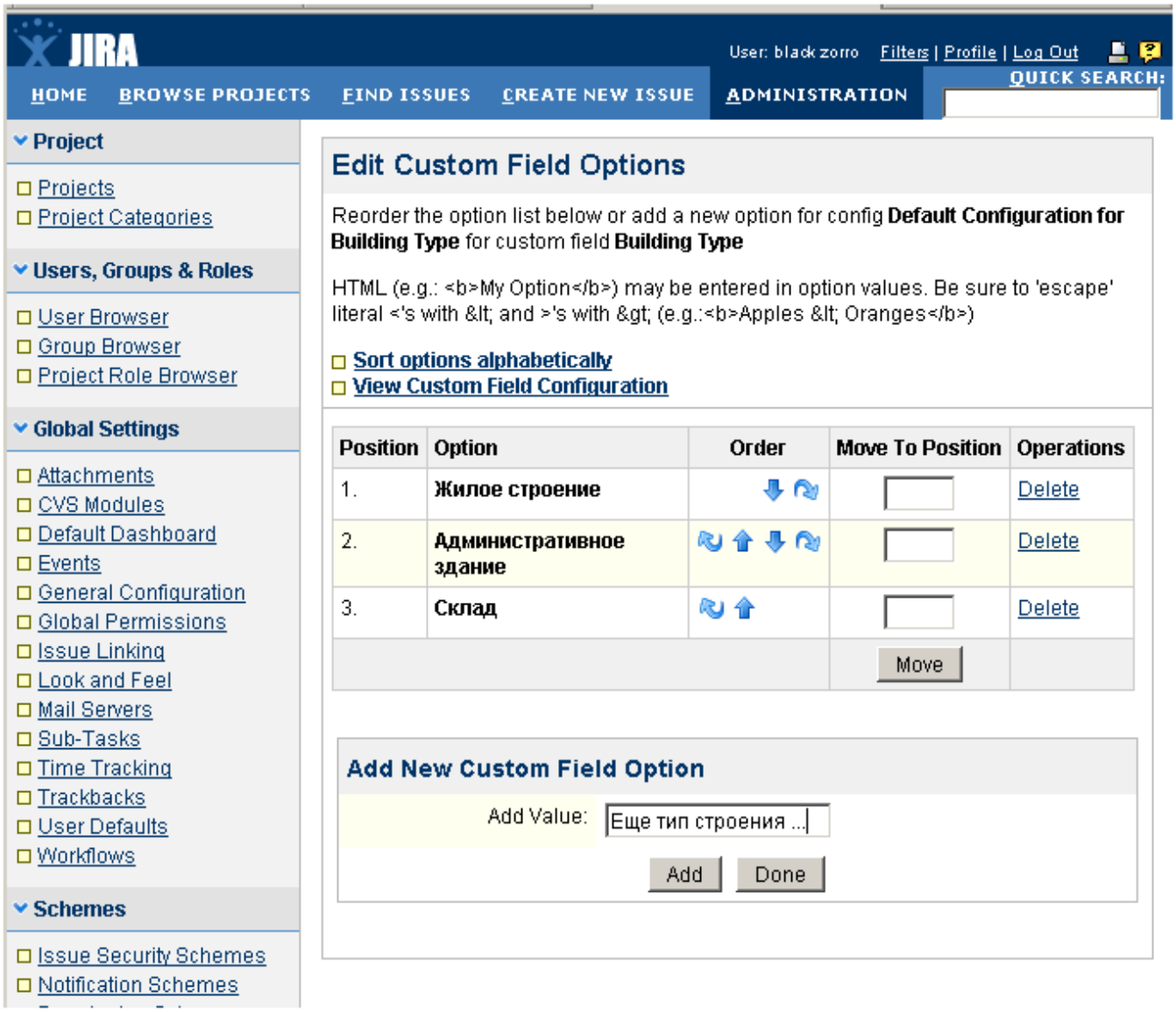
В прошлый раз я начал рассказ о методике создания Workflow, соответствующего особенностям тех проектов и этапов их разработки, которые приняты в вашей организации. Напоминаю, что Workflow состоит из этапов, правил перехода между этапами. Для примера я завершу создание Workflow некоторой строительной организации. Работы, выполняемые ею, включают составление дизайн-документа, разработку пакета чертежей, утверждение документов и выполнение, собственно, строительных работ. Примечание: в дальнейшем я буду часто употреблять словосочетание "в режиме администратора перейдите на закладку X". Здесь предполагается, что для выполнения некоторой операции вы должны не только войти внутрь JIRA от имени пользователя с административными привилегиями. Но следующий шаг заключается в переходе в главном меню на пункт "ADMINISTRAION" и выбор в расположенной слева экрана панели инструментов кнопки "X". Итак, начали. Первым шагом нужно в режиме администратора зайти на страницу "Issue Types", затем на закладке "Global Issue Types" мы должны создать виды Issue (заданий), которые специфичны для вас (те виды работ, которыми занимается наша строительная фирма). Например, я добавлю Issue: "Construction Order" (строительный заказ). В качестве параметров при его создании следует указать название, комментарий, иконку, а также тип (т.е. будет ли данный Issue служить для представления задач или подзадач). В качестве подзадач я создам еще два вида Issue, таких как "Разработка документации" (Development of the documentation) и "Выполнение строительных работ" (Construction Work). Затем нужно создать новую "Issues Types Schema" (схему типов задач), которая будет использована для управления некоторым проектом. Изначально в JIRA уже есть одна схема (Default Issue Type Scheme), которая описывает процесс разработки ПО. Она нам не очень подходит, поэтому я перехожу на закладку "Issues Types Schema". Затем говорю, что надо добавить новую схему (я назвал ее "Building Schema") и указываю то, какие типы Issue (задач) входят в ее состав:



Теперь я перехожу к настройке "Custom Fields" (пользовательских полей). Дело в том, что раз я планирую создать систему управления проектами специфическую для строительной организации, то мне нужно будет полностью перестроить экранные формы для добавления нового Issue (задания). Начну я с того, что создам несколько нестандартных полей. Дело в том, что обычные поля, такие как информация об аппаратном обеспечении, операционной системе (которые отображаются на форме создания нового Issue) мне не нужны. А вот поле "Тип Строения" с перечислением возможных вариантов (жилое, склад, административное) будет полезным. Также я хочу создать текстовое поле для хранения адреса, по которому будут выполняться работы. Итак, в режиме администратора я перехожу по ссылке "Custom Fields", там я вижу список пользовательских полей (изначально он пуст) и жму на кнопку "Add Custom Field". В появившейся форме я должен указать тип нового поля. Среди вариантов: и падающие списки для выбора значений (вот мое поле с перечнем типов строений), и список радиокнопок, и поле для ввода числа. Есть набор checkbox кнопок, найдется текстовое поле для ввода URL-адресов, также есть поле для ввода даты и поле, где можно ввести произвольный текст (ага, вот и поле для хранения адреса). Более того, часть вариантов типов полей тесно интегрирована с информацией хранящейся в JIRA. Например, есть падающий список для выбора пользователей, групп. Выбрав тип поля вы попадаете на второй экран, где нужно указать название поля и то к каким типам Issue, он может быть применим (я выбрал все три Issue созданные шагом ранее):



Завершающим этапом создания поля будет указать то, к каким Экранам оно будет привязано (здесь я ничего не выбрал т.к. методы работы с Экранами хочу показать чуть позже). При добавлении второго поля "Тип Строения" я выбрал тип "Select List". Затем, для того чтобы указать возможные значения для этого List-а, я после создания поля в таблице (где перечислены все пользовательские поля), в графе "Operations" нажал на кнопку "Configure", затем кнопка "Edit Options":



Теперь я хочу заняться созданием Workflow соответствующего работе строительной организации. Начну я с создания статусов. Статус – соответствует определенному этапу работы (составление дизайн-документа, разработку пакета чертежей, утверждение документов и строительные работы). Для этого я в режиме администратора перехожу на вкладку "Statuses" и добавляю туда эти четыре новых типа. Для каждого из них указывается название, комментарий, и иконка. Теперь мы готовы к созданию Workflow. Для этого я перехожу на вкладку "Workflows", там я увижу таблицу с одной единственной схемой jira (это стандартная схема по которой работает JIRA и ее можно использовать как основу при проектировании собственной схемы). Внизу страницы есть форма добавления нового Workflow, там нужно указать название и примечание к создаваемой схеме (я назвал схему "Architecture"). Когда мы сделаем это, то увидим, что в таблице Workflows будут уже две строки: jira и Architecture. Пора заняться наполнением схемы рабочего процесса конкретными состояниями и правилами перехода. Для этого я перехожу по ссылке "Steps" и попадаю на экран конструирования схемы. Устроен он очень просто: посередине страницы находится таблица с зарегистрированными состояниями (изначально в ней только один элемент "open" – т.е. проект был открыт). Я последовательно добавляю четыре новых этапа работ и привязываю их к созданным ранее шагам (создание дизайна, конструкторские работы, утверждение документов и строительные работы). Последним элементом Workflow я помещаю элемент "Closed" (проект был закрыт). Можно было бы развить пример и добавить состояния соответствующие исправлению ошибок на каждом из шагов, но я решил упростить схему и не делать этого. Конструирование workflow еще не завершено: нужно написать правила переходов из одного состояния в другое. Для этого в таблице с перечислением шагов, нужно нажать на ссылку "Add transition", так я попаду на экран планирования переходов. Каждый переход состоит из названия, примечания, шага, к которому будет выполнен переход, и Transition View (Экрану, который мы увидим при выполнении подобного перехода). Экран нужен для того, чтобы организовать ввод дополнительной информации, например, при переходе от стадии "Утверждения документов" к "Строительным работам" нужно ввести регистрационные номера утвержденных документов. Учтите, что переходы (Transitions) являются однонаправленными, например, можно выполнить переход от состояния "создание дизайна" к "чертежным документам", но вот обратный переход не возможен (как будто бы). Поэтому для сложного бизнес-процесса, будет просто огромнейшее количество переходов:

ECTS ПОИСК СОЗДАТЬ АДМИНИСТРИРОВАНИЕ Поиск:

View Workflow Steps — Architecture

This shows all of the steps for **Architecture**.

- View all **workflows.**
- View all **statuses.**

Step Name (id)	Linked Status	Transitions (id)	Действия
Open (1)	Open	Начать дизайнить (11) >> Дизайн	Add Transition | Delete Transitions | Пр. | View Properties
Дизайн (2)	Design	Начать чертить (21) >> Конструкторские работы	Add Transition | Delete Transitions | Пр. | View Properties
Конструкторские работы (3)	Drawing works	Утверждение документов (31) >> Утверждение документов	Add Transition | Delete Transitions | Пр. | View Properties
Утверждение документов (4)	The statement of the project	Строительные работы (41) >> Строительные работы	Add Transition | Delete Transitions | Пр. | View Properties
Строительные работы (5)	Construction Works	Приемка работ (51) >> Проект завершен	Add Transition | Delete Transitions | Пр. | View Properties
Проект завершен (6)	Closed		Add Transition | Пр. | View Properties

Add New Step

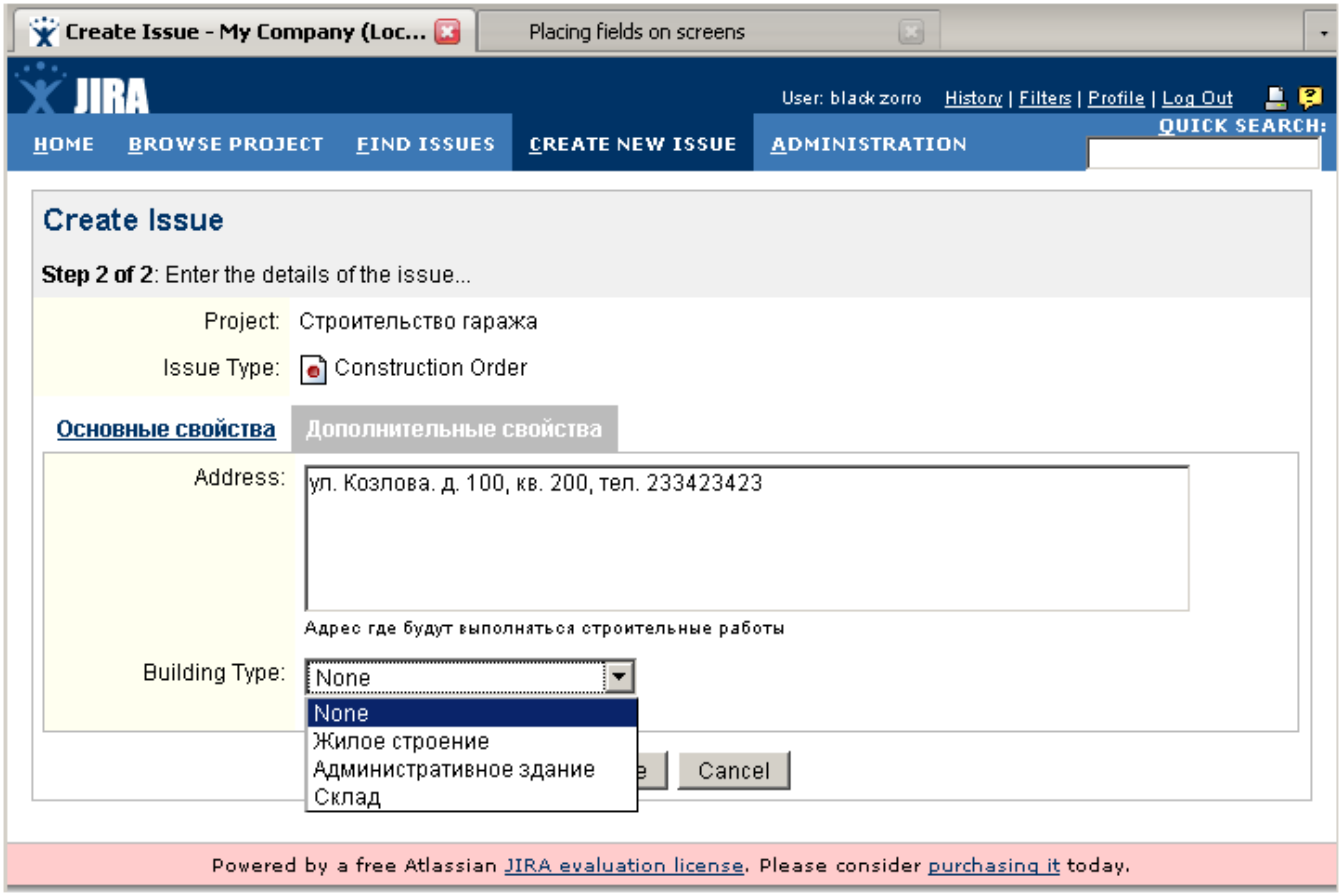
Завершив проектирование workflow, необходимо указать правила, по которым при создании проекта будет выбран соответствующий workflow. Итак, я в режиме администратора перехожу на закладку "Workflow Schemes", где вижу, что таблица схем пока пуста. Затем я нажимаю на ссылку "Add workflow scheme" и попадаю на форму создания новой схемы. Там нужно указать название схемы и примечание к ней (схему я назвал Building Schema). Следующий шаг – наполнить схему правилами (ищем ссылку "Assign Workflow"). Настройка ассоциации построена на выборе комбинации "Тип задания" (Issue) и "Workflow". В падающем списке я выбираю, соответственно, "Construction Order" и "Architecture". По аналогии я добавляю ассоциации и для двух оставшихся типов issue (строительные работы, разработка документации).

Теперь мы полностью готовы к созданию проекта выполняющегося по схеме строительных работ. На начальных шагах (ввод имени проекта, его кода, ответственных за выполнение работы лиц) нет никаких отличий от того, как мы создавали проект разработки программного обеспечения. Только после того как проект был создан, нужно перейти в его карточку, затем нажать в графе "Workflow scheme" кнопку "Select" и выбрать созданную на предыдущем этапе схему "Building Scheme". Поздравляю, первый этап пройден, и мы можем начать наполнять проект конкретными задачами.

Делается все это также как и в прошлый раз (как для задач разработки ПО). Вот только после создания задачи, если перейти в ее карточку, то в меню "Available Workflow Actions" будут находиться не команды "RESOLVE, REOPEN", а созданные нами действия "Начать дизайнерские работы", "Утвердить документацию" и т.д. Если я создам задачу "строительство гаража", то смогу поместить в нее подзадачи "создание проекта документов" и "строительные работы" (вернитесь к началу статьи, там я создавал именно вот эти типы Issue). Осталось только разобраться, как в карточку задания поместить пользовательские поля (адрес и тип строения). Пользовательские поля не являются самостоятельной сущностью – они должны быть добавлены на Экран (форму интерфейса JIRA, которая показывается при наступлении определенных событий). Начну я с того, что создам новый экран и наполню его нужными для меня характеристиками. Например, когда я создаю Issue проекта разработки, то такие поля, как версия программы, для которой данная задача актуальна или версии, для которых проблема уже решена, мне не нужны.

Снова в режиме администрирования я перехожу на закладку "Screens", там я вижу таблицу с перечнем Экранов (изначально там будут три экрана: Default Screen, Resolve Issue Screen, Workflow Screen) и того, где эти Экраны используются. Например, экран Workflow Screen используется когда нужно ввести данные о новой задаче. Редактировать данный Экран я не буду, чтобы не смущать тех, кто добавляет проект разработки ПО, таким полями как тип строения или его адрес. А вот внизу страницы есть форма для добавления нового Экрана. Использую ее, и создаю Экран с именем "Building Initiated Screen". После создания Экрана он появится в таблице, а я должен наполнить его полями (ссылка "Configure"). Форма редактора Экрана разделена на две части: поля и закладки. Я создаю две закладки: "Основные свойства" и "Дополнительные свойства". Затем, последовательно выбирая эти закладки, с помощью формы "Add Field" добавляю на каждую из них некоторые поля (в форме добавления поля есть список "Fields to add", где отображаются все доступные поля, как служебные, так и созданные мною). Интерфейс конструктора экрана очень удобен: легко добавлять, удалять закладки, менять их порядок расположения. Поля, которые были ошибочно размещены на какой-то закладке, без проблем перемещаются в нужное место. Момент в том, что вы не имеете права выбирать какие попало поля: есть поля обязательные и нет. Обязательно нужно чтобы в состав Экрана было включено поле: "Summary Of Issue". Завершив планирование экрана нужно выполнить его привязку к определенному этапу работы над проектом.

Делается это не просто, но привычно. Также как мы создавали новый тип Workflow, а затем схему его применения, так и здесь нужно создать новую Экранную Схему. Для этого переходим на закладку "Screen Schemes", там мы увидим таблицу со всеми доступными схемами управляющими, когда и какой Экран должен быть показан. Нам нужно добавить новую схему (внизу страницы), указав при этом ее название, примечание к схеме, и то какой экран будет показан по-умолчанию (указываю созданный на предыдущем шаге "Building Initiated Screen"). Новую схему я решил назвать "Building Screen Sheme". Если мы создали несколько Экранов (например, отдельный экран для начала строительного проекта, отдельный экран для каждого из шагов его выполнения или завершения), тогда нужно не уходя с закладки " Screen Schemes" нажать кнопку "Configure" и добавить там ассоциации "операция-экран". К этой работе я вернусь чуть попозже. Теперь будем привязывать созданную схему к определенному набору задач (Issue). Для этого снова в режиме администрирования переходим на закладку "Issue Type Screen Schemes". Там мы увидим ассоциации между Набором Экранов и списком проектов. Внизу страницы расположена форма добавления новой "Issue Type Screen Schemes". Используем ее, вводим название схемы (я назвал схему "Building Type Screen Sheme"), затем вводим примечание к схеме, и в падающем списке "Screen Scheme" выбираем созданную чуть ранее схему "Building Screen Scheme". После создания схемы мы видим в таблице уже две строки. Вот, правда, созданный на предыдущем этапе проект "Строительство гаража" привязан не к нашей схеме, а используемой при разработке ПО. Пора это исправить. Для этого переходим на закладку с перечнем проектов, далее переходим в карточку проекта и в графе "Issue Type Screen Scheme" используем кнопку "Select" для того чтобы выбрать "Building Type Screen Sheme". Теперь проверим, что же у нас получилось, для этого переходим на экран создания новой задачи, выбираем проект "Строительство гаража", затем тип задачи "Construction Order". И видим, что внешний вид экрана добавления задачи поменялся, теперь он состоит из двух закладок ("Основные свойства" и "Дополнительные свойства") и на второй из этих закладок находятся созданные мною в самом начале статьи пользовательские поля "Адрес" и "Тип строения":



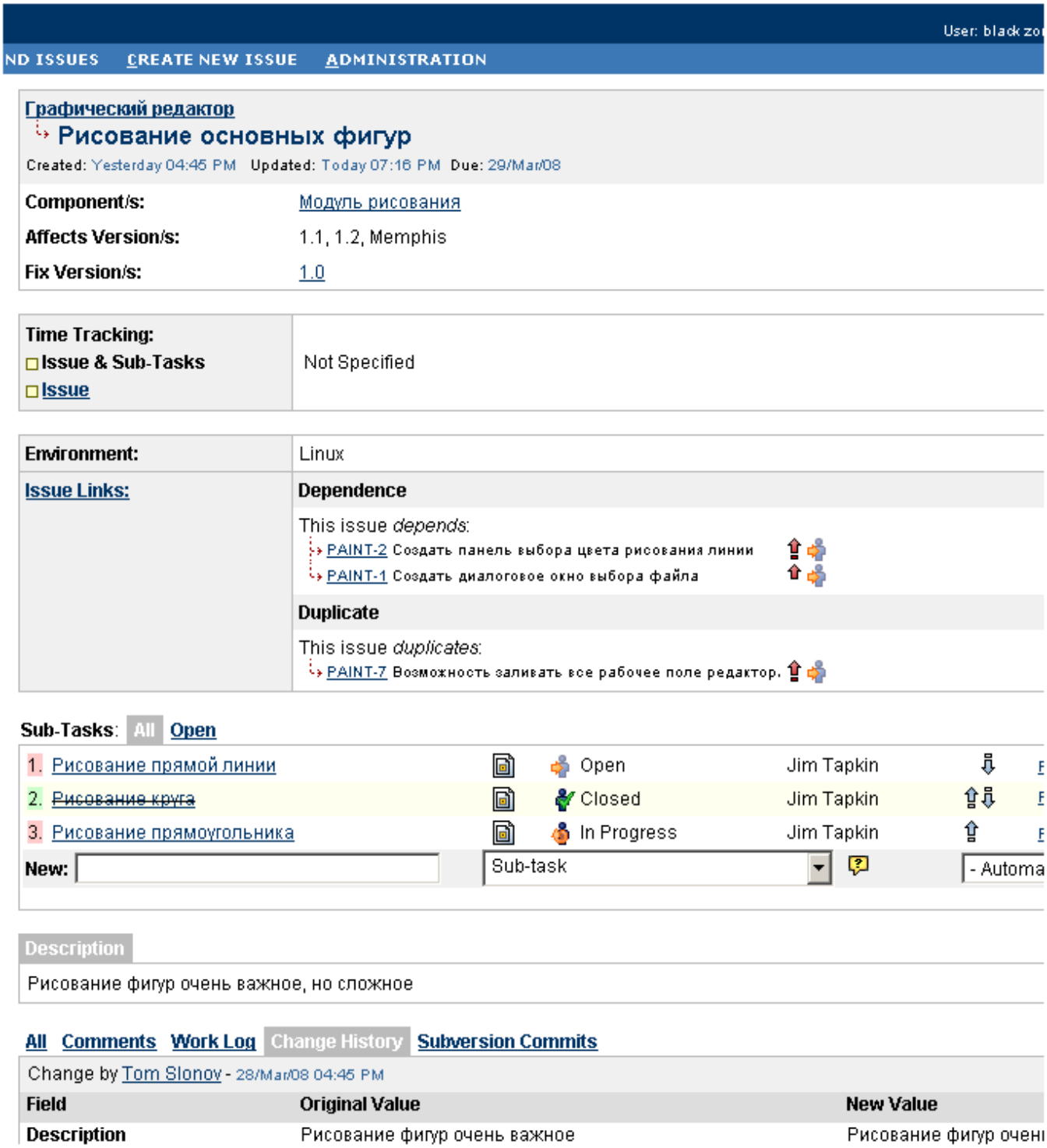
Попробуем начать работу над свежесозданной задачей и выполнить переход к состоянию "дизайнерские работы". Для этого я перехожу в карточку задания, в меню "Available Workflows Actions" жму на ссылку "Начать дизайнерские работы" и ... вижу совсем не тот экран, который хотел бы. Я снова вижу поля список версий ПО, для которых актуален этот переход, еще вижу требования к операционной системе, а вот полей адреса и типа строения нет. Будем исправлять. Для этого я должен вернуться на форму редактирования шагов Workflow (в административном режиме закладка "Screen Schemes"). Затем в таблице с перечнем схем я выбираю созданную мною же "Building Screen Sheme" и перехожу по ссылке "Configure". Теперь я должен с помощью расположенной внизу формы добавить ассоциацию между Экраном и конкретным этапом Workflow, когда появляется форма для ввода пользователем некоторой информации. Так я добавил связь между событием создание задачи ("Create Issue") и "Building Initiated Screen". Такой же Экран был присоединен и к операциям "Edit Issue" и "View Issue". Если теперь вернуться назад к карточке задания и попробовать ее редактировать, переходить с одного состояния в другое, то внешний вид экрана будет тот, что нужно: созданная мною форма, с двумя закладками и пользовательскими полями.

Сегодня я хотел показать гибкость и настраиваемость JIRA. Показать то, что ее можно применять не только в сфере разработки ПО, но и во многих других областях (если конечно вы сможете осилить покупку Enterprise версии JIRA за несколько тысяч условных зайчиков). В следующий раз я продолжу и завершу рассказ о JIRA, покажу как вести учет затраченного времени, как интегрировать JIRA и SVN.

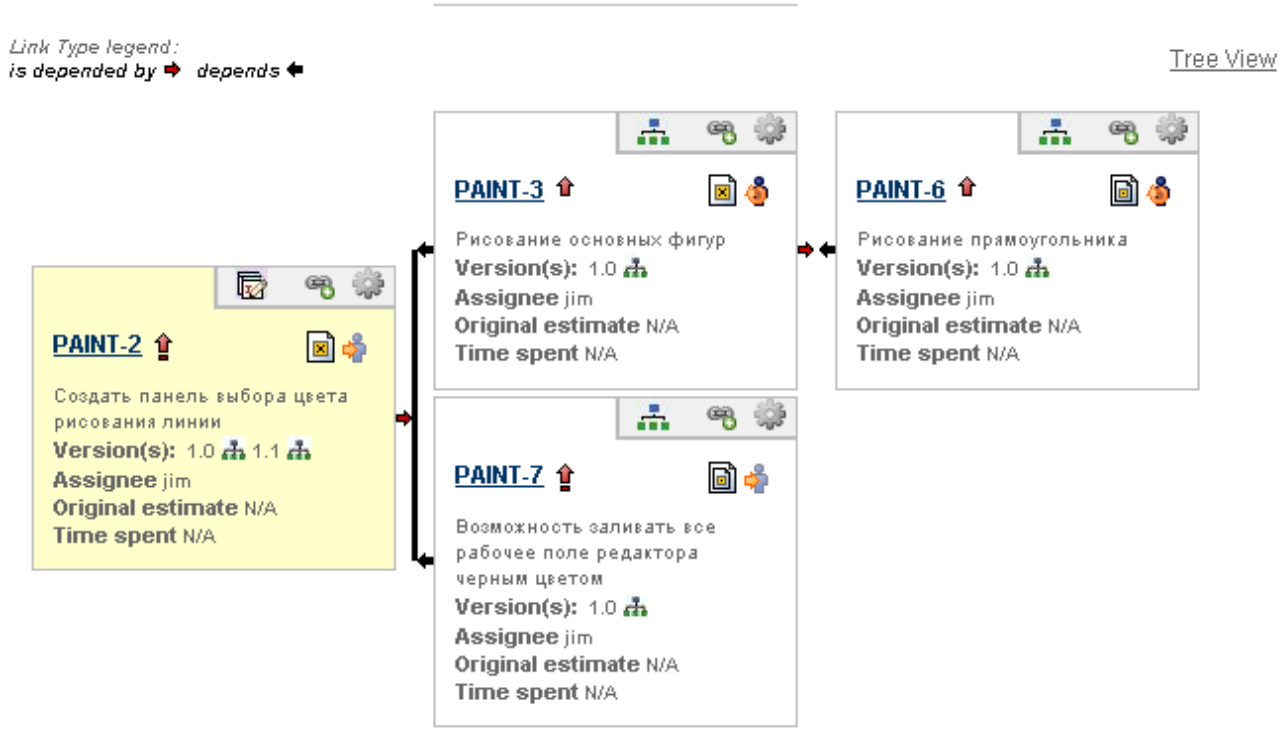
JIRA: Все, что пожелаешь, Хозяин. Часть 4

Сегодня я завершаю рассказ о JIRA, системе управления проектами, задачами не только в сфере разработки ПО но и во многих других областях. Основное внимание будет сегодня посвящено интеграции JIRA с системами управления версиями файлов и мониторингу за выполняемой работой, оценкой трудозатрат и планированию.

Прошлая статья была посвящена методике применения JIRA для моделирования бизнес-процессов строительной организации. Сегодня забудьте об этом – я посвящаю рассказ задачам программирования и только ним. Проект это набор задач и подзадач. Для каждой задачи есть статус, в котором она находится, есть еще и список всевозможных полей с характеристиками задачи, описанием того что, когда, кем было или будет сделано. В практике этого мало. Часто задачи связаны между собой не примитивным отношением "задача-подзадача", а произвольными связями. Например, с помощью "Issue Linking" можно ввести понятие зависимости задач друг от друга, подчиненности, дублирования задач. Так, при добавлении записи об обнаруженной ошибке, тестер может ошибиться и ввести запись об уже обнаруженной и внесенной в базу данных ошибке. Давайте рассмотрим, как настроить и использовать " Issue Linking ". Прежде всего, нужно в режиме администратора перейти на закладку "Issue Linking", затем с помощью кнопки "Activate" включить режим связи между задачами. После чего на этой же странице внизу появляется табличка с перечнем видов связей (изначально табличка пустая), мы должны ее заполнить. Например, я добавлю связь с именем "Duplicate" (признак того, что данный Issue дублируется другой Issue), в качестве значений поля "Outward Link Description" я ввожу слово "duplicates", а для поля "Inward Link Description" указываю значение "is duplicated by". Значения, которые вводятся в эти три поля – на ваше усмотрение, просто стремитесь к тому, чтобы это были корректные синтаксические конструкции: "связь от чего-то" и "связь к чему-то". Теперь попробуем сделать связь между двумя Issue. Для этого я с помощью меню "BROWSE PROJECT" перехожу к экрану с перечнем всех проектов, выбираю среди них любое задание, а теперь внимание: в карточке товара, на панели инструментов появился новый пункт "Link this issue to another issue". На появившемся Экране я в падающем списке "This issue" указываю отношение, которое существует между текущим Issue и другим issue (для выбора второго Issue используйте ссылку "select issue"). Удобно, что в поле "Issues" можно ввести сразу несколько кодов задач через запятую, например, "CALC-4,CALC-10" (здесь предполагается что CALC – это код текущего проекта). Также можно из задачи (Issue) для одного проекта сослаться на задачу, принадлежащую к другому проекту. Не забывайте, что отношение связи является двунаправленным и если вы для одной задачи сказали "duplicates", то та другая задача в своей карточке будет иметь значение "is duplicated by". Внешний вид карточки задачи со связями показан на рисунке:


User: black zo
ND ISSUES
CREATE NEW ISSUE
ADMINISTRATION
Графический редактор
Рисование основных фигур
Created: Yesterday 04:45 PM Updated: Today 07:16 PM Due: 29/Mar/08
Component/s: Модуль рисования
Affects Version/s: 1.1, 1.2, Memphis
Fix Version/s: 1.0
Time Tracking:
Issue & Sub-Tasks
Issue
Not Specified
Environment: Linux
Issue Links:
Dependence
This issue depends:
PAINT-2 Создать панель выбора цвета рисования линии
PAINT-1 Создать диалоговое окно выбора файла
Duplicate
This issue duplicates:
PAINT-7 Возможность заливать все рабочее поле редактор.
Sub-Tasks: All Open
1. Рисование прямой линии Open Jim Tapkin
2. Рисование круга Closed Jim Tapkin
3. Рисование прямоугольника In Progress Jim Tapkin
New:
Sub-task
- Automa
Description
Рисование фигур очень важное, но сложное
All Comments Work Log Change History Subversion Commits
Change by Tom Slonov - 28/Mar/08 04:45 PM
Field
Original Value
New Value
Description
Рисование фигур очень важное
Рисование фигур очени

Однако это еще не все, если ссылки между заданиями сложны, и вы хотите видеть не "ближайшие" прикрепленные задачи, но увидеть весь граф задач сразу, то встроенных средств JIRA не хватает. К счастью, для JIRA есть множество плагинов. Часть из них разрабатывается создателем JIRA, компанией Atlassian, часть разрабатываются посторонними организациями, часть плагинов идут "за бесплатно", а за некоторые нужно платить денежки. Вот адрес страницы, где можно найти перечисление доступных для JIRA плагиновhttp://confluence.atlassian.com/display/JIRAEXT/Plugin+Search. Для примера я загрузил плагин "Links Hierarchy Reports" со страницыhttp://confluence.atlassian.com/display/JIRAEXT/Links+Hierarchy+Reports. Плагин идет в виде jar-файла. Затем я устанавливаю плагин, для этого просто скопирую jar-файл в папку E:\Program_Files_2\apache-tomcat-6.0.14\webapps\jira\WEB-INF\lib (путь к самому tomcat, конечно, будет отличаться). В принципе, это стандартная методика установки и любого другого плагина, хотя прочитать инструкцию по установке никогда не вредно. Проверить то, что плагин был успешно установлен можно, если в режиме администратора зайти на вкладку "Plugins" и найти на странице с перечнем всех установленных в JIRA плагинов строку " Links Hierarchy Reports ". После того как были созданы несколько Issue, а затем связаны отношениями, давайте перейдем на страницу "BROWSE PROJECT". Там на правой боковой панели мы видим информацию о проекте и помимо привычных статистик, сколько в составе проекта задач и к каким категориям они относятся (открытые, решенные, в ходе работы) плагин "Links Hierarchy" добавил в раздел "Reports" два новых вида отчета: "Link Hierarchy Report For Versions" и "Link Hierarchy Report For Issues". Для построения первого отчета необходимо указать то, для какой версии проекта (вспоминайте, что проекты состоят из компонентов и версий) будет построен отчет, также укажите вид связи между issue (отчет нельзя построить сразу для всех видов "Link"-ов). Внешний вид отчета показан на рисунке:

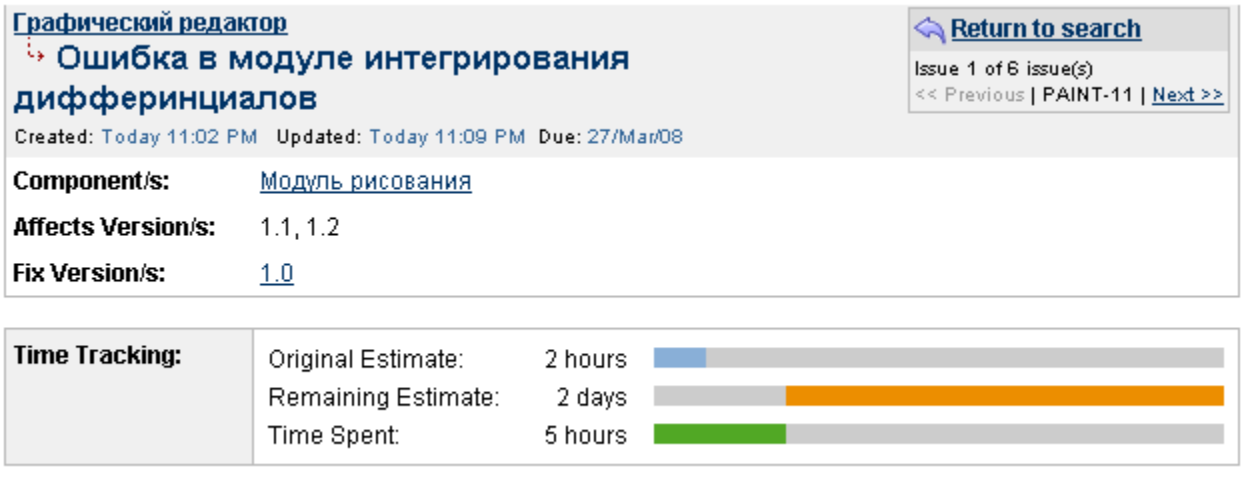


Созданный отчет не является статическим образованием: мы можем сразу, не выходя из него с помощью управляющих кнопок на карточке задания добавлять на диаграмму новые связи, также выполнять такие действия "закрытие issue", "начать работу". Второй вид отчета, который появился на карточке проекта "Link Hierarchy Report For Issues", приводит к появлению точно такой же диаграммы, но теперь отбор issue выполняется не на основании принадлежности к одной версии, а номер issue, от которого будет построен граф зависимостей, должен быть введен вами самими.

Есть особый вид связи между задачами, точнее, их текстовыми примечаниями, называемый "trackback linking". С этим термином вы наверняка уже встречались в сфере совершенно не связанной с JIRA и это ... блоги. Смотрите, когда некто Вася написал в internet на своем блоге классную статью, то ему очень хочется чтобы все ее читали и хвалили Васю. Для этого он ввел возможность комментировать свою статью (стоит упомянуть также и то, что часто эти комментарии публикуются в виде RSS-ленты). Но давайте ближе к "trackback linking": другой блоггер, Петя прочитал статью Васи, восхитился ею и пишет в своем блоге заметку, мол, я прочитал такую классную статью от Васи. А чтобы Васе знал, что его статью прочитали, то программное обеспечение Петиного blog-а посылает блогу Васи сообщение, мол, на тебя ссылаются. Фактически trackback-и упрощают разработку документации, так если вы на одной html-странице создали ссылку на другую страницу, то теперь вам не нужно открывать и редактировать текст второй страницы, чтобы показать, кто же ссылается на нее. Для работы с trackback нужно, прежде всего, в режиме администратора перейти на закладку "trackbacks". В появившейся форме с настройками trackback есть три параметра: будет ли JIRA принимать trackback-запросы от посторонних сайтов (по-умолчанию, включено), также будет ли JIRA отправлять trackback-запросы на другие сайты (по-умолчанию, выключено) и опция "URL Patterns to Exclude from Outgoing Trackback Pings". Эта опция необходима в случае, если вы включили отправку trackback-ов, но хотите чтобы на определенные адреса такие извещения не посылались. Одним словом, если вы укажите в графе "URL Patterns to Exclude from Outgoing Trackback Pings" некоторое регулярное выражение, то оно будет играть роль блокирующего фильтра для тех адресов сайтов, куда trackback-и посылать не нужно. Итак, я включил в настройках отправку trackback-ов по всем адресам. Теперь попробуем в JIRA создать две задачи (предположим, что их коды PAINT-2 и PAINT-3). Затем я перехожу в режим редактирования задачи PAINT-3 и в графе текста примечания пишу:

```
trackback-
  PAINT-2
  http://localhost:8080/jira/browse/PAINT-2
```

Обратите внимание, что ссылку на другую задачу в JIRA я могу создать не только путем указания полного http-адреса к ней, но и просто указав короткий код "PAINT-2". Jira проверяет все такие коды на предмет корректности. И если в локальной инсталляции jira есть задача под таким номером, то комментарий к issue будет преобразован в ссылку. Более того, если код принадлежит задаче, которая была разрешена, то она будет отображен в виде перечеркнутой надписи. Если перейти ко второй задаче, то увидите, что ее карточка изменилась и содержит в графе "External references" примечание о том, что другая задача ссылается на нее.
Важной частью управления проектами является вопрос оценки времени затраченных на выполнение работ. Хотя JIRA при изменении статуса issue сохраняет в карточке задачи время, когда статус был изменен. Но использовать эти даты/время для анализа ...ни за что! Начнем с того, что просто глупо подсчитывать время на выполнение работы путем отнимания от одной даты другой, практически невозможно вести учет времени, если задача несколько раз меняла состояние, переходя из OPENED -> RESOLVED и обратно. Для кардинального решения проблемы в jira был введен модуль "Logging Work on an Issue". Как и многие другие модули, его нужно предварительно включить. Для этого в режиме администратора переходим на закладку "Time-tracking". Затем жмем на большую кнопку с надписью "Activate", предварительно введя информацию о графике работы (количество часов в рабочем дне и количество рабочих дней в неделе). Теперь попробуем создать задачу и проверим то, как jira ведет учет рабочего времени. Однако перед этим, когда вы создаете новый issue, обратите внимание на то, что внизу формы есть поле для ввода "Original Estimate" т.е. здесь мы должны ввести планируемое время выполнения работы (например, 2h – работа займет два часа). После включения мониторинга за временем работы поменялся внешний вид карточки issue: на левой панели появилась новая кнопка "Log work done ". Фактически, всякий раз, когда вы завершаете какой-то этап работы над задачей, то должны нажать на эту кнопку, затем в появившемся окне указать то, сколько времени вы потратили на работу. Также нужно указать какая операция будет выполнена над предварительно спланированным временем работы (приемлемый вариант по-умолчанию, когда из времени Estimate будет вычтено указанное вами время работы). Внешний вид карточки задания также изменится: на ней появятся три разноцветные полосы с планируемым временем работы, с затраченным временем и оставшимся временем:



Для того чтобы получить подробную информацию о проекте, его составляющих, наилучшим выбором будет установить для jira плагинhttp://confluence.atlassian.com/display/JIRAEXT/JIRA+Charting+Plugin. После этого в карточке проекта в графе "Reports" появится "пачка" новых отчетов. Во-первых, отчет Recently Created Issues Report отображает в виде столбчатой диаграммы количество задач, которые были созданы и решены за последние X дней (количество дней указывается при создании отчета). Отчет Pie Chart Report отображает в виде круговой диаграммы соотношение между пользователями, работавшими над задачами проекта (можно оценить, кто сделал больше задач). Для оценки работы не по количественному показателю, а по затраченному времени используйте отчет User Workload Report. Отчет Version Workload Report строит таблицу с детальными и итоговыми сведениями, сколько работы еще осталось сделать для каждого из типов задач и для каждого из участников (грубо говоря, когда разработчики освободятся). Для того чтобы видеть не остатки времени, а реальное соотношение запланированных, выполненных работ и их остатков лучше всего использовать отчет Time Tracking Report:

Report: Time Tracking Report

Description:
This report shows the time tracking details for a specific project.

Excel View

Time Tracking Report for Графический редактор (1.0)

Including all sub-tasks

Progress: 25%

Accuracy: -105%

1d 2h completed from current total estimate of **4d 7h**
Issues in this version are **behind** the original estimate of **2d 3h** by **2 days, 4 hours**.

Key		Summary	Original Estimate	Σ	Est. Time Remaining	Σ	Time Spent	Σ	Accuracy	Σ
PAINT-3		Рисование основных фигур	-	-	-	-	-	-	-	-
PAINT-6		Рисование прямоугольника	-	-	-	-	-	-	-	-
PAINT-5		Рисование круга	-	-	-	-	-	-	-	-
PAINT-4		Рисование прямой линии	-	-	-	-	-	-	-	-
PAINT-11		Ошибка в модуле интегрирования диффе	2h	2h	2d	2d	5h	5h	-2d 3h	-2d 3h
PAINT-8		Интеграция со сканером	1d 2h	1d 7h	1d 2h	1d 4h	-	4h	on track	-1h
PAINT-9		Интеграция со сканером HP	2h	-	2h	-	-	-	on track	-
PAINT-10		Интеграция со сканером mustek	3h	-	0m	-	4h	-	-1h	-
PAINT-2		Создать панель выбора цвета рисования	-	2h	-	1h	-	1h	-	on track
PAINT-7		Возможность заливать все рабочее п	2h	-	1h	-	1h	-	on track	-
		Total		**2d 3h**		**3d 5h**		**1d 2h**		**-2d 4h**

Для работы остальных отчетов требуется выполнить дополнительные шаги: нужно добавить к задачам три пользовательских поля. Во-первых, Resolution Date (тип поля с таким же названием, и это не ошибка), затем поле Time in Status (тип поля тоже имеет название Time in Status), затем поле Date of First Response (и тип его Date of First Response). Когда мастер создания пользовательского поля будет просить вас привязать поле к каким-то Экранам, ничего не делайте. Теперь в режиме администрирования вы должны перейти на закладку "Indexing" и выполнить переиндексацию данных JIRA (просто нажмите кнопку "Re-index"). После чего вы можете вернуться обратно к карточке проекта и увидите, что остальные отчеты заработали. Так отчет "Created vs Resolved Issues Report" покажет соотношение между созданными задачи и тем, сколько их было решено. Отчет "Resolution Time Report" покажет время затраченное на разрешение задач. Отчет "Average Age Report" покажет вам то, сколько времени (дней) ожидалось решение задач.

Теперь рассмотрим методику интеграции JIRA и системы управления версиями (СУВ) документов. В состав JIRA изначально входит модуль связи с CVS, однако я с этой СУВ никогда не работал (да и в своих статьях описывал только SVN и Perforce). К счастью для JIRA есть плагины интеграции с SVN (даже два, только один нужно отдельно покупать). Сначала загрузите архив с плагином с сайта http://confluence.atlassian.com/display/JIRAEXT/JIRA+Subversion+Plugin (будьте внимательны, разные версии JIRA требуют разных версий плагина). Теперь скопируйте все файлы находящиеся в каталоге lib плагина в папку E:\Program_Files_2\apache-tomcat-6.0.14\webapps\jira\WEB-INF\lib, затем в папку E:\Program_Files_2\apache-tomcat-6.0.14\webapps\jira\WEB-INF\classes нужно скопировать файл subversion-jira-plugin.properties. Теперь этот файл нужно отредактировать. Внутри файла хранятся адреса подключения к SVN-репозиториям, к примеру:

```
svn.display.name=Repository for Project "A"
svn.root=svn://localhost/test/project
svn.display.name.1=Repository for Project "B"
svn.root.1=svn://localhost/test/project2
```

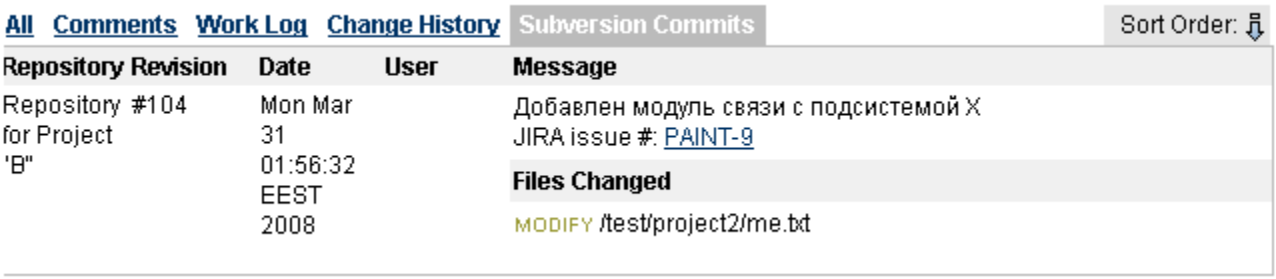
Здесь определены названия и пути к двум репозиториям (если потребуется третий, четвертый ... репозиторий, то их имена строятся по правилу svn.root.НОМЕР). Надо сказать, что при настройке репозиториев можно указать еще и путь к веб-интерфейсу для SVN, но раз я в серии про СУВ не рассказал ни об одном из таких продуктов, то пропускаю. Теперь перейдем к настройке СУВ, на примере, всеми любимой TortoiseSVN. В SVN для каждого файла или каталога введено понятие свойств. Проще говоря, мы можем присоединить к каждому файлу пары "ключ=значение". Есть небольшой список предопределенных свойств, и, кроме того, мы (или какая-нибудь программа, работающая с SVN) можем добавлять собственные значения. Итак, создайте в проводнике папку и выгрузите в нее с помощью checkout-а содержимое репозитория (одного из тех двух, которые я указал в примере файла настроек выше). Затем вызываем контекстное меню проводника "TortoiseSVN -> Properties" в появившемся диалоговом окне нужно указать параметры JIRA (просто выбираем в падающем списке название свойства и вводим его значение): **bugtraq:label** - Текст подсказки что-нужно вводить при отправке commit-а на сервер. Например, "enter JIRA issue #".

bugtraq:message - Текст, который будет показан в журнале правок. Например, "JIRA issue #: %BUGID%" (внимание: здесь и далее %BUGID% это место для подстановки кода задания).

bugtraq:url -- Адрес веб-страницы с карточкой задания. В моем примере это http://localhost:8080/jira/browse/%BUGID%.

bugtraq:number - Признак того будет ли код задачи только числовым (тогда значение этого свойства должно быть true, если же нет - false).

Теперь как только вы измените файлы и отправите их на сервер, то в появляющемся диалоговом окне commit-а, помимо поля примечания к правке, появится еще поле для ввода кода задачи, к которой относится данная правка. Изменения произойдут и в самой JIRA. Так в карточке задания появится закладка "Subversion commits", на которой отображается информация обо всех выполненных правках: название репозитория (где была выполнена правка), также дата и время правки, номер ревизии, сообщение (которое вы ввели как комментарий к commit-у), а также сведения о файлах, которые были изменены:



Оригинал статьи .