Experience in creation and implementation of integrated software development process automation system.

Grachev Anton
Luxoft Professional
Moscow, Russia
email: Agrachev@luxoft.com

Gavrilov Evgeny
Luxoft Professional
Moscow, Russia
email: Egavrilov@luxoft.com

Abstract (English)

The goal of the article is a LUXOFT experience presentation in development and implementation of LUXproject integrated development environment for software engineering process automation. Major issues of process automation tools are lack of integration, geographic team distribution, using different types of software engineering processes (therefore different sets of tools).

LUXproject is a "process environment" based on Atlassian products: JIRA and Confluence. Other tools required for process support could be integrated (or are already integrated) as well. Currently the solution implements RUP model (with CMMI support) and Agile model .

From the user's perspective LUXproject incorporates integrated products in a single Web interface and allows using the functionality of each integrated component in any projects. UI integration leads to a number of issues in keeping the uniform UI style as well as usability issues. Known solutions are described in the article.

Functional integration allows administering all components from a single place, activities monitoring, discovering artifacts links for future traceability and managing activities. System architecture based on "message bus" pattern and unified domain language. This will allow developing new functionality in future releases effectively.

Major implementation scenarios are presented: implementation of the process that is already used within an existing project as well as implementation of the predefined process templates to the new projects

***Keywords**: Automatization of softwore development process*

Опыт создания и внедрения интегрированной системы автоматизции процессов разработки программного обеспечения.

Грачев Антон
Люксофт Профешнл
Москва, Россия
email:Agrachev@luxoft.com

Гаврилов Евгений
Люксофт Профешнл
Москва, Россия
email: Egavrilov@luxoft.com

Abstract (Russian)

Целью статьи является представление опыта компании LUXOFT в создании и внедрении интегрированной системы для автоматизации процессов разработки программного обеспечения LUXproject. Описываются проблемы автоматизации процессов разработки: отстутствие интеграции, географическая распределенность команды и потребность использовать разные процессы разработки (следовательно, разные инструменты автоматизации).

LUXproject является «процессной оболочкой», которая базируется на продуктах Atlassian JIRA и Atlassian Confluence, интегрируя также и другие приложения, необходимые для поддержки процесса. В настоящий момент, LUXproject реализует модели процессов RUP (с поддержкой CMMI) и Agile.

С точки зрения пользователя, LUXproject объединяет интегрированные продукты в едином web-интерфейсе, предоставляя возможность пользоваться функциональностью каждого продукта во всех проектах. Это порождает проблемы поддержки единого стиля приложения и проблемы usability, решения для которых предложены в докладе.

Функциональная интеграция позволяет администрировать все компоненты из одной точки а также производить мониторинг активностей, отслеживать связи артефактов процесса и управлять активностями. Архитектура системы построена на основе шины сообщений и единого языка доменной модели, что позволяет быстро расширять функциональность продукта в дальнейшем.

Рассматриваются сценарии внедрения системы, как с учетом существующих работающих процессов в компании (отделе), так и для новых проектов с готовыми предустановленными в системе шаблонами процессов.

***Keywords**: автоматизция процессов разработки.*

1. Введение

Целью данной статьи является представление опыта компании LUXOFT в создании и внедрении интегрированной системы для автоматизации процессов разработки программного обеспечения.

Не секрет, что для многих компаний, занимающихся разработкой и внедрением программного обеспечения, стала стандартной практикой автоматизация процессов разработки. Для большинства компаний такие процессные области как, «управление проектами», «управление требованиями» «тестирование», «конфигурационное управление», уже немыслимы без автоматизации.

Однако, несмотря на довольно серьезную автоматизацию, существуют ряд проблем, которые не позволяют говорить о создании единой автоматизированной среды разработки и внедрения.

Для примера рассмотрим несколько ситуаций, иллюстрирующих острые проблемы автоматизации.

Первая из рассмотренных нами проблем, заключается в сложности быстрого получения информации о различных данных в проекте и ее сопоставления в едином визуальном «интерфейсе». Проиллюстрируем эту проблему на примере - в некоторой компании применяется программа для тестирования. Разработчик, получив данные о дефектах в коде разрабатываемого продукта, сталкивается с проблемой отождествления дефекта с описанием соответствующего требования, так как в системах тестирования в лучшем случае заносятся заголовки требований, а за подробным описанием приходится «залезать» в дебри уже программы по сбору требований.

Вторая проблема заключается в автоматизации распределенной разработки. Команды проекта, находящиеся в различных географических точках, могут испытывать проблемы, как с доступом к единой проектной базе, так и сложности с использованием систем у заказчика по политикам безопасности.

Третья проблема может заключаться в наличии различных подходов к разработке. В одном проекте используют Agile практики, в другом процессы RUP, соответственно множится и программное обеспечение для автоматизации процессов разработки, что влечет увеличение затрат Компании на автоматизацию.

Решением вышеописанных проблем, может стать единая интегрированная система разработки и управления проектом основанная на WEB решении для распределенного доступа.

2. Опыт компании LUXOFT в создании и внедрении интегрированной системы

Компания LUXOFT, в том числе и для преодоления вышеописанных проблем, создала свою интегрированную систему автоматизации процесса разработки - LUXproject ®.

При разработке данной системы были поставлены следующие цели:

- Комплексная поддержка жизненного цикла разработки программного обеспечения (от заключения контракта до поддержки);
- Поддержка распределенной разработки
- Поддержка различных систем разработки

По сути LUXproject является «процессной оболочкой», которая базируется на Atlassian JIRA и Atlassian Confluence. Благодаря данной «процессной оболочке» существует возможность интеграции с различными программными продуктами, как линейки продуктов компании Atlassian, так и продуктами других компаний.

В настоящий момент LUXproject поддерживает две модели процессов разработки: одна основана на процессах RUP (и поддерживает модель CMMI), вторая - на Agile практиках.

Для модели, основанной на процессах RUP, в настоящий момент реализован следующий процессный функционал:

- Управление задачами;
- Управление рисками;
- Формирование отчетности;
- Управление требованиями;
- Управление изменениями;
- Управление сборкой;
- Управление тестированием (включает управление test cases и дефектами);
- Управление качеством (процессные аудиты и анализ статистических данных);
- Управление конфигурацией;
- Управление коммуникациями (возможность вести базу знаний);

Для модели, основанной на практиках Agile реализован следующий функционал:

- Управление задачами (в части - ведение бэклога продукта, управление релизами,

ведение бэклога итерации, персональный план работ);

- Управление рисками;
- Управление дефектами;
- Управление коммуникациями (Scrum and retrospective meetings);
- Управление конфигурацией;

Основными пользователями системы являются, как исполнители проекта, так и заказчик.

Система поддерживает ролевой доступ, то есть каждый участник проектной команды в зависимости от проектной роли имеет тот или иной доступ к функционалу системы.

3. Интеграция интерфейсов пользователя

С точки зрения пользователя, LUXproject объединяет интегрированные продукты в едином web-интерфейсе, предоставляя возможность пользоваться функциональностью каждого продукта во всех проектах.

При интеграции нескольких web-приложений в единую систему встает вопрос о том, как организовать объединение пользовательских интерфейсов, чтобы, с одной стороны, получить всю функциональность приложений, с другой – создалось ощущение единства стиля и целостности всей системы.

Первая задача интеграции заключается в введении общих навигационных элементов (header, footer) и единого стилевого оформления на всех страницах системы.

Для решения поставленной задачи исследовались следующие подходы:

1. Использование существующих портальных решений позволяет относительно быстро сынтегрировать интерфейсы приложений. Но исследования показали, что они достаточно тяжелы и требуют адаптации приложений к использованию в виде портлетов.

2. Использование комбинации паттернов “Proxy” и “Decorator” позволяет перехватить контент приложения, обработать его (добавив и изменив необходимые элементы) и выдать конечному пользователю. Минус подхода – перехват и фильтрация контента достаточно дорога по ресурсам и существенно влияет на быстродействие системы; накладывание аспекта-декоратора на все страницы может вызвать большое количество разнообразных ошибок, т.е. требует большого объема тестирования. Плюсы – есть широкие возможности по контролю контента интегрируемого приложения, которые можно использовать для функциональной интеграции. Изменение оформления приложений достигается добавлением css-стилей в шапку в декоратор. В качестве реализации Proxy используется ProxyHttpServlet, декорирование обеспечивает фреймворк Sitemesh (http://www.opensymphony.com/sitemesh). Данный подход используется в LUXproject.

3. Поставленную задачу можно решить, внедрив в каждое приложение общие элементы (header, footer). Причем конечные пользователи будут ходить на те же приложения, что и раньше, но ссылки в header-е будут вести на разные приложения в интегрируемой системе. Этот подход имеет существенное преимущество в скорости доступа (фактически, интеграция почти не влияет на время отклика). Основной недостаток – необходимость изменять исходный код интегрируемого приложения, что не всегда представляется возможным. По сравнению с вариантом 2, теряется возможность контроля над интегрируемыми приложениями из одного (центрального) приложения. Этот подход был использован в продукте Atlassian JIRA Studio (http://jira.com).

После успешной реализации описанных задач, начинают возникать проблемы другого уровня сложности. Каждое приложение имеет собственную модель работы пользовательского интерфейса: названия и типовое расположение кнопок и других элементов управления, горячие клавиши, расположение и состав меню. Помимо этого, возможны опасные терминологические конфликты, когда под одними и теми же названиями в интегрируемых системых скрываются совершенно разные понятия. Причем все подходы абсолютно оправданы в конкретных приложениях (документная модель Confluence, ориентированная на контент, и табличная модель JIRA), но сбивают с толку пользователей, работающих в них всех, практически одновременно. Данная проблема встает все острее при развитии функциональной интеграции, когда все больше сценариев использования выполняется в нескольких компонентах, заставляя пользователя переключатся с одной модели интерфейса на другую.

Предлагается два пути решения проблемы. Первый путь – дать пользователю возможность четко осознавать, когда он находится в одном приложении, а когда уже переключился в другое, как это сделано в Atlassian JIRA Studio при помощи закладок. Это не решает проблемы, но меньше запутывает пользователя и позволяет быстрее понять модель взаимодействия приложений. Второй путь – выносить большую часть функционала в одно приложение, используя

другие в качестве движка. Например, если требования ведутся как wiki-страницы + JIRA items, возможно реализовать все операции с требованиями не выходя из wiki, используя JIRA только как движок Workflow и для формирования отчетов. В последнем случае, аналитику не придется переключаться между приложениями в большинстве случаев. Этот подход постепенно реализуется в LUXproject, где Wiki-движок Confluence используется как основа для интерфейса интеграции всех компонент (см. рис 1).

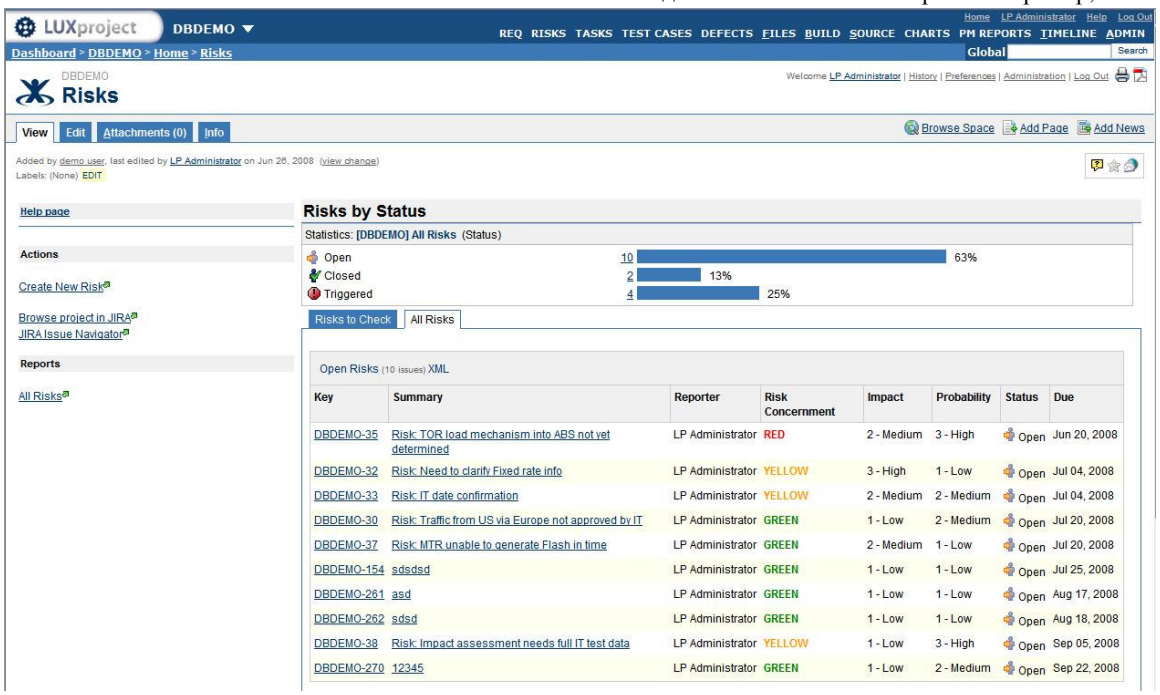

Рис 1. Иллюстрация интеграции JIRA функционала с помощью Wiki-движка Confluence

4. Функциональная интеграция приложений

Основной целью системы LUXproject является поддержка процессов разработки программного обеспечения. Для этого используются интегрированные приложения, выполняющие определенную роль в процессе. Каждое приложение по сути представляет собой полноценный продукт, которым можно пользоваться независимо, но их интеграция вместе дает значительно больший эффект практически для всех активностей процесса. Основные цели

функциональной интеграции – это единая точка администрирования всех приложений и отдельных проектов, мониторинг активностей, возможность отслеживать связи артефактов процесса и управлять активностями.

Как известно, интеграция нескольких приложений попарно рано или поздно приводит к проблемам слишком жестких зависимостей, поэтому эффективнее реализуется с использованием паттернов Enterprise Messaging и использования шины сообщений. В последнем случае, при возникновении события в одном из компонентов, он посылает в шину сообщение, на которое подписываются заинтересованные стороны (другие компоненты), что позволяет достаточно быстро расширять функциональность всей системы, не затрагивая другие модули.

Кроме этого, вводится единый язык доменной модели, которым оперируют все слои интеграции отдельных компонент, преобразуя свои объекты в объекты единого языка и наоборот. Например, проектные роли эффективно отображаются на глобальные роли в JIRA и на группы в Confluence и на соответствующие сущности в других компонентах

Язык доменной модели представляет собой основные обобщенные сущности модели процесса разработки: роль, активность, артефакт.

На эту модель накладывается множество типизированных связей и ограничений, представленных в виде правил. Такая модель позволяет отслеживать связи между артефактами, поставляемыми разными компонентами (например, требование в wiki, test case, defect в JIRA и изменения исходного кода в репозитарии Subversion), отслеживать целостность этих связей при помощи правил (например, запрещать вносить изменения в репозиторий, без указания дефекта в JIRA в соответствующем статусе).

Используемая модель и достаточно богатые функциональные возможности компонентов позволяют гибко настраивать систему под нужды конкретных заказчиков при внедрении.

5. Внедрение интегрированной системы

Внедрение интегрированной системы имеет некоторую специфику. В связи с тем, что зачастую, в различных проектах компании могут применяться различные методологии разработки ПО. Простая инсталляция системы, с заложенной в нее типовой процессной моделью, не эффективна, так как произойдет несовпадение схем работы в системе и в жизни.

Поэтому здесь нам видятся два подхода. Компания должна сама для себя ответить вначале на вопрос, насколько существующие процессы разработки ПО у нее эффективны (например, с помощью проектных аудитов).

Если компания пришла к выводу, что ее процессы не достаточно эффективны, то она может произвести «реинжениринг» процессов путем принятия процессной модели, заложенной в интегрированную систему. В дальнейшем, сотрудники проходят обучение по новым схемам процессов работы.

Если компания считает, что ее процессная среда достаточно совершенна и накоплен существенный багаж знаний и опыт по проектам (особенно это актуально если используется системы, подобные JIRA), то в начале внедрения проводится сбор требований для уточнения схем процессов, статусов, форм отчетности. В дальнейшем, внедряемая интеграционная система настраивается под требования заказчика, и вводится в эксплуатацию.

Однако может так оказаться, что в центре разработок или компании, применяется несколько методологий разработки программного обеспечения. Одни проекты идут по практикам Agile, другие используют RUP процессы. Как быть в данной ситуации?

В компании LUXOFT, в рамках системы LUXproject, данную проблему решили путем создания проектных шаблонов. Проектный шаблон – это совокупность настроек функциональных модулей системы, (состоящих из issues, wiki–контента, версионного репозитория, шаблонов документов и базы знаний) под конкретную методологию разработки. На рис. 2. представлен пример визуального интерфейса проектного шаблона для Agile проектов.

6. Заключение

Подводя итоги, можно сделать вывод, что описанная архитектура системы и схемы ее внедрения позволяют реализовать достаточно гибкий инструмент для управления процессами разработки ПО. Подход к интеграции компонентов системы, позволяет проектной команде управлять проектной средой заточенной под конкретную методологию разработки, а также производить оптимизацию процессов уже во время выполнения проекта.

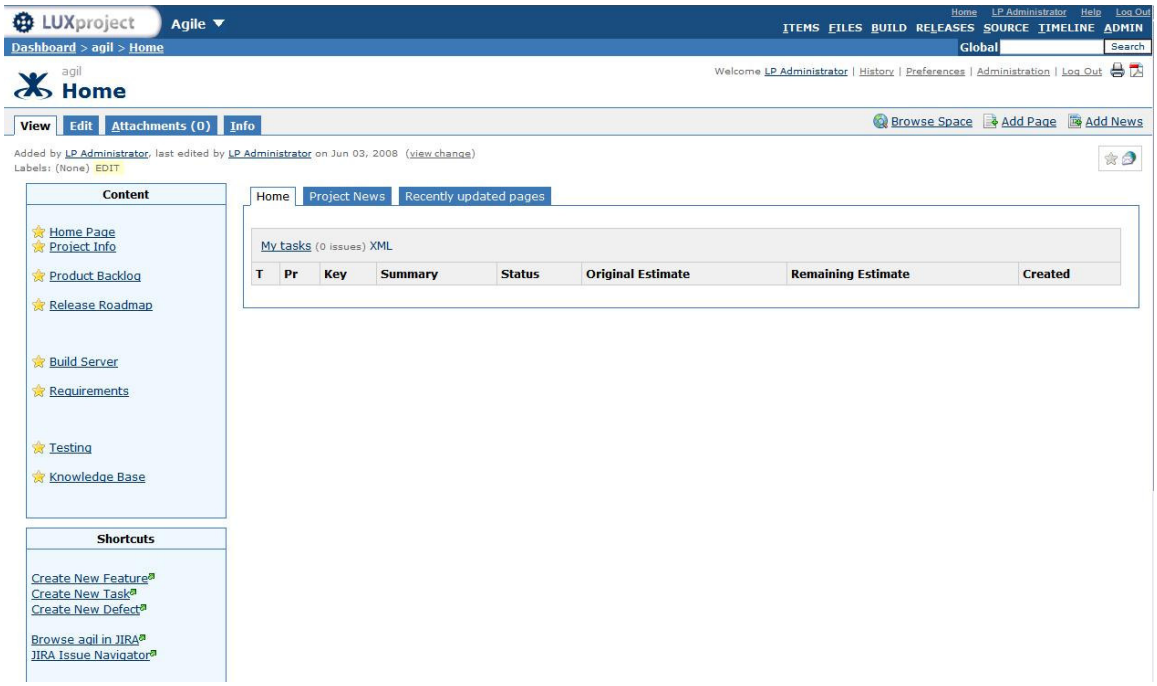

Рис. 2. Иллюстрация визуального интерфейса проектного шаблона “Agile”