<https://finecosoft.ru/ibm_rational_jazz_comp?ysclid=mcn9bwhq10895167200>

**Что такое Jazz от IBM - утопия или будущее?**

*Февраль 2011 г.*

**Аннотация:**

В данной статье автор пытается проанализировать свой опыт, полученный при внедрении традиционных средств IBM Rational и средств платформы Jazz ([http://jazz.net](http://jazz.net/)), и сравнить полученные знания и впечатления. Также выполнена попытка оценки потенциальных перспектив Jazz на ближайшее будущее.

**Автор:**

[Дмитрий А. Лесин](mailto:mail@finecosoft.ru), сертифицированный специалист IBM, опыт использования средств IBM Rational при разработке программных продуктов с 1999 года, практический опыт в области консалтинга, внедрения и использования средств IBM Rational Jazz - с 2008 года. Руководитель Департамента развития бизнеса [ООО "Финэкософт"](http://www.finecosoft.ru/) и преподаватель [Центра IT Обучения "Institutio"](http://www.institutio.ru/).

**Содержание**

* Вступление
* Типовое решение на базе традиционной линейки IBM Rational
* Как к этому пришли или немного истории
* Известные проблемы или как появился Jazz
* Так что же такое Jazz?
  + Архитектурное решение
  + Набор инструментов
  + Сообщество пользователей
* Окончательное сравнение или что не хватает для полного счастья
* Заключение
* Ссылки

***Вступление***

Итак, прошло уже несколько лет, как было во всеуслышание провозглашено появление новой технологической платформы Jazz под патронажем IBM. И, хотя не говорилось открыто, но, тем не менее, все мы отчетливо понимали, что наступили революционные изменения на рынке традиционных технологий и средств IBM Rational. И это в то время, когда, казалось бы, подразделение Rational компании IBM намного опередило других конкурентов в области построения управляемой автоматизированной среды разработки, благодаря созданию максимально полного комплекта технологий и средств, тесно интегрированных друг с другом. Что же скрыто за этой революционной сменой поколений, зачем все это вообще было начато и чем таким особенным отличаются новые технологии от традиционных технологий IBM Rational, а также от технологий других компаний, специализирующихся на создании решений в области управления разработкой ПО? Почему такая крупная компания на рынке, как IBM, в очередной раз решила рискнуть своим имиджем и, имея большую вероятность растерять свои конкурентные преимущества на рынке, приступила к продвижению Jazz.

Что же подумало большинство людей (а я основываюсь на наиболее популярном мнении, господствующим как среди наших заказчиков, с которыми мы имеем дело при проведении консалтинговых проектов, а также среди разработчиков нашей компании и слушателей курсов Центра обучения), которые уверенно внедрили в своих организациях или только собирались внедрять средства автоматизации разработки ПО на базе традиционных инструментов IBM Rational (далее - традиционной линейки IBM Rational), когда им вдруг объявили о появлении абсолютно новой платформы Jazz. Наверно среди большинства из нас появилось некоторое чувство непонимания и вопрос, а не запуталась ли сама IBM во всем этом. Затем это непонимание перешло в насмешку типа "IBM в своем стиле". А возможно такая насмешка возникла сразу, как это часто бывает, когда судят о ведущих брендах: IBM, Microsoft, HP и др.

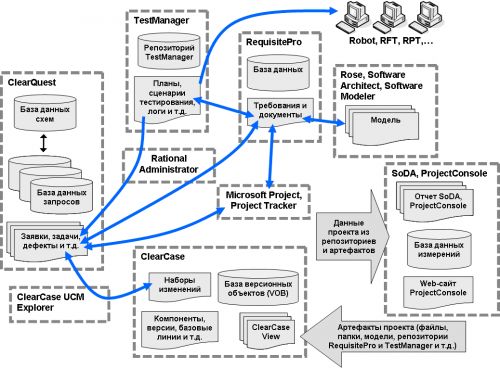
Но так ли легко оценить возникшую ситуацию, как это может показаться на первый взгляд? Ведь наверно даже такая крупная компания IBM, потратившая уйму сил и времени на маркетинг традиционных средств IBM Rational и добившаяся здесь огромного задела на ближайшие годы, вряд ли станет без причин все резко перестраивать и "с нуля"развивать иные направления.

На самом старте нашей дискуссии я сразу же хочу отметить, что не претендую на провозглашение истины в последней инстанции, что многие выводы здесь являются личным мнением автора. Я крайне уважаю собственные мнения читателей, даже если они коренным образом отличаются от высказываемых мной мыслей. При этом я обещаю стремиться к максимальной объективности, по крайней мере, насколько это возможно, когда выводы делаются всего одним человеком.

Для сравнения возможностей направлений в данной статье мы будем оперировать с последними версиями обеих линеек по состоянию на начало 2011 г.. Соответственно, для традиционной линейки IBM Rational (которая, правда официально уже давно распалась на отдельные продукты) - это v7.x, а для линейки Jazz - v3.0.x.

***Типовое решение на базе традиционной линейки IBM Rational***

Для начала вспомним, как примерно выглядит типовое решение, сформированное IBM Rational в течение ряда лет для автоматизации разработки ПО на базе традиционных средств. Здесь мы с Вами прекрасно понимаем, что практически все организации, внедрившие эти средства, отличаются друг от друга, имеют отличающиеся стартовые условия и в каждой из них необходимо учитывать разнообразные дополнительные ограничения. Тем не менее, в качестве своеобразного шаблона, можно для рассмотрения принять следующую схему автоматизированной системы, от которой обычно удобно отталкиваться при необходимости адаптации традиционной линейки IBM Rational в организациях (Рисунок 1).

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/f8e350940cd8fd32053619b1bf00fa09.gif)

**Рисунок 1. Типовая архитектура традиционной автоматизированной системы управления разработкой от IBM Rational**

На данном рисунке не показаны многие другие инструменты, которые также закрывают конкретные области разработки и имеют разнообразные интеграционные связи с традиционными средствами. Среди таких отсутствующих инструментов можно выделить, например, средство управления наработками и шаблонами IBM Rational Asset Manager, средство автоматизации управления сборкой в масштабе целого предприятия IBM Rational BuildForge, средство моделирования и оптимизации бизнес-процессов IBM Websphere Business Modeler и др. Здесь показаны только те инструменты, которые были отобраны предварительно с целью упрощения рисунка. А именно их мы выбрали потому, что они являются базовыми для автоматизации большинства типовых проектов разработки. Тем более, не стоит забывать, что основным акцентом данной статьи является сравнение не конкретных инструментов и оценка полноты пакетов автоматизации, а кое-чего другого. Того, что часто называется термином архитектура или архитектурное решение и сопровождается словами "надежная", "грамотная", "эффективная" и т.д. И как раз именно на анализе архитектурных преимуществ обеих платформ мы акцентируемся в наибольшей мере и постараемся сделать выводы о перспективах новшеств, продвигаемых сегодня IBM.

Итак, что же представлено на нашем рисунке? Как было сказано выше, здесь мы можем видеть средства автоматизации ключевых процессов разработки информационных систем и важнейшие интеграционные связи между ними, реализованные в традиционной линейке IBM Rational. В частности, на данном рисунке показаны:

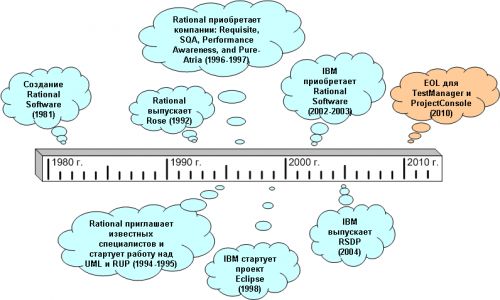
* **IBM Rational ClearQuest**, который является мощным средством управления запросами на изменения, позволяющим аккумулировать запросы различных типов, двигать их по состояниям жизненного цикла, организовывать эффективный контроль за реализацией запросов и гарантировать, что все они будут рассмотрены в необходимые сроки и учтены в той или иной степени. В частности, IBM Rational ClearQuest в рамках самой полной схемы "Enterprise" позволяет организовать:
  + управление начальными запросами заинтересованных лиц (заявками на разработку, запросами на улучшения, инцидентами и т.д.);
  + управление низкоуровневыми задачами, назначаемыми конкретным исполнителям;
  + управление ошибками, порождаемыми специалистами по тестированию на разных стадиях проверки качества выпускаемых программных продуктов или конечными пользователями при обнаружении проблем в уже поставленных продуктах;
* **IBM Rational RequisitePro** - средство для организации управления требованиями при командной разработке ПО, которое может оказать неоценимую помощь, когда необходимо подключить к работе аналитиков для проведения подробного анализа полученных запросов на изменения. Обычно в ходе такого анализа аналитикам требуется понять, уточнить и согласовать с заказчиками или их представителями, что конкретно должно быть сделано в рамках выставленных запросов. При этом часто запросы на изменения приводят к необходимости создания новых требований или коррекции уже существующих.
* В ходе выявления и уточнения требований может быть использована хорошо зарекомендовавшая себя техника визуального моделирования с помощью разнообразных средств типа **IBM Rational Rose**, **IBM Rational Software Architect**, **IBM Rational Software Modeler** и др. С помощью них могут быть сформированы модели на разных уровнях абстрагирования данных, например, модели сценариев использования, модели анализа и проектирования, модели данных и другие модели, которые определены в рамках нашего процесса разработки.  
  Средства разработки кода (Integrated Development Environment или IDE) в составе традиционной линейки IBM Rational изначально не были предусмотрены, хотя Eclipse, который лежит в основе IBM Rational Software Architect и IBM Rational Software Modeler уже включает встроенные возможности программирования на Java. Но при этом многие из перечисленных средств визуального моделирования могут в той или иной степени включать различные интеграции с популярными средствами разработки кода, например, Microsoft Visual Studio .NET, Microsoft Visual Studio v6, Delphi и т.д.
* **IBM Rational TestManager** является средством управления тестированием при разработке информационных систем. Тестирование заключается в том, что над выпускаемым продуктом выполняются предопределенные операций, описанных в т.н. сценариях тестирования. При этом эти операции могут быть запущены одновременно на нескольких тестовых стендах, параллельно или в необходимой последовательности, и позволяют осуществить проверку разрабатываемых систем по т.н. точкам верификации (иначе говоря, оценить соответствие разрабатываемой системы некоторым заранее предопределенным условиям, которые ожидаются нами по завершении различных операций).  
  Особенно важна роль средств автоматизации при организации функционального регрессионного тестирования, когда от итерации к итерации на повторяемой основе проводится полная проверка всего функционала разрабатываемой системы, что позволяет значительно повысить эффективность тестирования.  
  IBM Rational TestManager позволяет запускать тесты, относящиеся к различным типам тестирования. В частности, можно запускать функциональные и нагрузочные тесты IBM Rational Robot, функциональные тесты IBM Rational Functional Tester, тесты для контроля нагрузки и производительности системы IBM Rational Performance Tester, контроль утечек памяти с помощью IBM Rational Purify, контроль производительности с помощью IBM Rational Quantify, оценку полноты прохождения (или покрытия) с помощью IBM Rational PureCoverage и др.  
  Также следует отметить, что IBM Rational TestManager – это один из немногих инструментов традиционной линейки, у которого репозиторий создается с помощью фактически стороннего средства – IBM Rational Administrator. Кроме того, для продукта IBM Rational TestManager в конце 2010 г. объявлен статус EOL (End Of Life), что говорит о прекращении разработки инструмента и лишь временном продолжении его сопровождения для существующих пользователей. Вместо IBM Rational TestManager уже сегодня предлагается использовать средство IBM Rational Quality Manager из Jazz.
* Кроме всего прочего, в составе описываемой традиционной линейки крайне велика роль средства управления версиями и конфигурациями **IBM Rational ClearCase**. Как известно, при разработке систем любого уровня опасно напрямую менять проектные артефакты, т.к. можно потерять достаточно серьезные наработки и соответственно потерять время, а значит и деньги, если внесены некорректные изменения. Т.е. надо сохранять историю артефактов. Управление версиями и конфигурациями позволяет организовать внесение контролируемых изменений в данные проекта, благодаря формированию новых версий артефактов при внесении изменений. С помощью версионности всегда существует возможность умышленно откатить изменения назад к некоторой зафиксированной ранее точке и восстановить артефакты до требуемого состояния в прошлом.  
  При этом IBM Rational ClearCase позволяет организовать работу не только над отдельными артефактами, но и над целыми группами артефактов и даже репозиториями. Так, в нашей архитектуре под управление версиями и конфигурациями могут быть помещены различные файлы (документы, исходные тексты, скрипты) и сделаны снимки репозиториев IBM Rational RequisitePro и IBM Rational TestManager.  
  Кроме того, с помощью IBM Rational ClearCase обеспечивается параллельная работа нескольких разработчиков над одними и теми же артефактами. Специальные механизмы разрешения конфликтов позволяют объединять результаты работы нескольких специалистов и не потерять сделанные наработки.  
  IBM Rational ClearCase позволяет осуществить формирование базовой линии или зафиксировать текущую конфигурацию артефактов. Это позволяет при необходимости в любое время в будущем восстановить среду разработки, полностью идентичную текущей, если вдруг понадобится начать разработку параллельной конфигурации от этой точки или просто решить какие-то дополнительные задачи в рамках ранее зафиксированного релиза.
* **IBM Rational SoDA** и **IBM Rational ProjectConsole** обеспечивают прозрачность проекта и автоматизацию процедур генерации необходимых документов. Оба средства обращаются к разнообразным проектным данным, правда результаты их работы немного отличаются. Основная цель IBM Rational SoDA - создание электронных документов Microsoft Word с необходимым форматированием и представлением данных. А в основе IBM Rational ProjectConsole лежит Web-сайт проекта, на котором можно познакомится с аналогичными документами, сгенерированными в Web-формате, а также получить рассчитанные метрические показатели проекта. При этом можно настроить практически любые способы расчета метрик (поддерживается язык программирования Perl) и отобразить полученные значения метрик в виде красивых графических представлений (светофоров, спидометров или пользовательских элементов).
* Как известно, в традиционной линейке IBM Rational отсутствуют полноценные средства управления проектами. По этой причине в данной нише достаточно прочно обосновался классический продукт **Microsoft Project**. С помощью него формируется или уточняется план проекта, определяются задачи и ресурсы, которые будут назначены ответственными для выполнения задач.  
  Правда напомню, что всего пару лет назад был известен достаточно мощный продукт IBM Rational Portfolio Manager, предназначенный не только для организации управления отдельными проектами, но для автоматизации целого портфельного управления проектами с мощным функционалом отбора кандидатов в проекты. Но в числе многих других направлений для IBM Rational Portfolio Manager был объявлен EOL. Скорее всего, это произошло потому, что архитектура данного инструмента и трудоемкость его сопровождения не вписывались в стратегию IBM в области развития средств автоматизации разработки информационных систем.  
  Также следует отметить, что не всегда Microsoft Project используется в составе традиционной линейки. В этом случае для формирования задач на основе выявленных требований можно использовать прямую интеграцию между IBM Rational ClearQuest и IBM Rational RequisitePro.
* Небольшой довесок**IBM Rational Administrator** применяется для создания некоторых репозиториев и настройки интеграционных связей между некоторыми средствами традиционной линейки IBM Rational.

Когда мы говорим о средствах IBM Rational, то не в последнюю очередь имеем в виду интеграционные связи между ними. Какие же интеграции доступны в традиционной линейке IBM Rational? Кратко перечислим их:

* **Интеграция между IBM Rational ClearQuest и IBM Rational RequisitePro**, которая позволяет связывать запросы на изменения с требованиями и сохранять информацию о том, какие требования были затронуты при реализации тех или иных запросов и, наоборот, какие запросы оказали влияние на то или иное требование в ходе разработки.  
  А при отсутствии Microsoft Project данная интеграция позволяет создавать связи между задачами IBM Rational ClearQuest, которые были сформированы для реализации конкретных требований IBM Rational RequisitePro, и этими требованиями.
* **Интеграция между IBM Rational RequisitePro и Microsoft Project**, которая позволяет создавать новые задачи в плане проекта, нацеленные на реализацию конкретных требований. Благодаря этому можно уточнить время выполнения задач по реализации требований в текущей итерации, определить ресурсы для таких задач, выполнить необходимые денежные и иные оценки для оптимизации работы проектной команды. При этом всегда можно оценить, зачем появились те или иные задачи, какие требования были реализованы с помощью них или какие изменения были реализованы в этих требованиях.
* **Интеграция Microsoft Project и IBM Rational ClearQuest**, позволяющая экспортировать задачи плана проекта в IBM Rational ClearQuest и, таким образом, доводить до исполнителей необходимые назначения по задачам. Данный функционал реализуется с помощью специального модуля Project Tracker, который интегрируется напрямую в Microsoft Project при инсталляции средств IBM Rational.  
  В свою очередь, информация о реализации задач может быть обратно отражена в плане проекта Microsoft Project, благодаря чему в нем обновляется процент выполнения задач на привычных для руководителей проекта диаграммах Ганта.
* **Интеграция между IBM Rational RequisitePro и средствами визуальной разработки**, которая значительно облегчает синхронизацию требований при изменении визуальных моделей и синхронизацию визуальной модели при изменении требований.  
  При этом некоторые средства визуального моделирования позволяют достичь более высокой степени интеграции моделей с требованиями, а другие - меньшей. В частности, с помощью IBM Rational Software Architect можно практически любой значимый элемент визуальной модели связать с требованиями. В дальнейшем, любые изменения в модели выведут нас только на те требования, которые следует исследовать на предмет корректности их формулировок. Обычно таких требований оказываются единицы или десятки при небольших итеративных модификациях модели, но никак не все требования в репозитории, число которых может доходить до нескольких тысяч и более. Вспомним известный пример создателей продукта IBM Rational RequisitePro Дина Леффингуэлла (Dean Leffingwell) и Дона Уидрига (Don Widrig) из компании Requisite в своей книге "Принципы работы с требованиями к программному обеспечению. Унифицированный подход" ("Managing Software Requirements. Unified Approach"), что при проектировании самолета Boeing-777 было определено порядка 300.000 требований.  
  А вот классический инструмент визуального моделирования IBM Rational Rose, к сожалению, не позволяет достичь такой же высокой степени интеграции с IBM Rational RequisitePro. С помощью IBM Rational Rose можно лишь определять связи от заголовков самих сценариев использования или их реализаций, но никак не от отдельных элементов в их описаниях.
* **Интеграция между IBM Rational RequisitePro и IBM Rational TestManager**, гарантирующая возможность проверки того, насколько конечный продукт соответствует выставленным требованиям. Требования IBM Rational RequisitePro связываются со сценариями тестирования IBM Rational TestManager и такая связь позволяет отчитаться по реализации требований перед различными заинтересованными лицами (например, перед заказчиками или руководителями разработки). Таким образом, если по результатам тестирования мы получили успешно пройденные сценарии тестирования (успешно пройденные - значит для этих сценариев тестирования достигнуты поставленные ранее цели), то связанные с этими сценариями требования в IBM Rational RequisitePro можно считать реализованными.
* **Интеграция между IBM Rational TestManager и IBM Rational ClearQuest**, благодаря которой организуется эффективный дефект-трэкинг. Иначе говоря, прошло тестирование и получили лог с интерпретацией всех имевших место событий. Просматриваем лог и, если считаем, то некоторое событие говорит о наличии проблемы в конечном продукте, то добавляем описание соответствующего дефекта в IBM Rational ClearQuest. При этом, от этого дефекта всегда можно вернуться к исходному событию в логе и узнать, в каком сценарии тестирования и для каких требований все это произошло.
* **Интеграция IBM Rational ClearCase и IBM Rational ClearQuest**, которая позволяет достичь высокого уровня контроля за внесением изменений в проектные артефакты. Это происходит, благодаря возможности связывать запросы на изменения IBM Rational ClearQuest с наборами изменений в артефактах IBM Rational ClearCase в рамках работы механизма Unified Change Management (UCM). Как результат, можно увидеть что именно и в каких именно артефактах было изменено при выполнении поставленной задачи конкретным исполнителем.  
  Особенно полезен данный механизма для менеджера проекта, который всегда будет иметь информацию о том, кто и что сделал в проекте.
* **Интеграция средств традиционной линейки с IBM Rational SoDA и IBM Rational ProjectConsole** позволяет последним двум инструментам понимать формат анализируемых данных и обеспечить их представление в удобном для анализа виде. В частности, эти средства понимают данные IBM Rational ClearQuest, IBM Rational RequisitePro, IBM Rational Rose, IBM Rational Software Architect/Modeler (за исключением IBM Rational ProjectConsole), IBM Rational TestManager, IBM Rational ClearCase, Microsoft Project, Microsoft Word, простые текстовые и некоторые графические файлы.

***Как к этому пришли или немного истории***

Давайте оглянемся немного назад и вспомним, как все представленное выше появлялось и развивалось в течение целого ряда последних лет! При этом будем брать только достаточно значимые для нас события в ракурсе написания этой статьи.

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/c7d0c133e86ae82c04acb51589dad0e7.gif)

**Рисунок 2. Ключевые события Rational**

Итак, компания **Rational Software** появилась в 1981 г. Изначально она называлась “Rational Machines”, но позднее была переименована в более известную Rational Software. Это было сделано для того, чтобы исключить двусмысленность понимания целей компании, когда можно было подумать об ее ориентации на аппаратное обеспечение.

Ключевой целью Rational Software всегда являлось создание средств управления разработкой ПО. Первым собственным продуктом компании стал инструмент, сегодня более известный под названием APEX. Его назначение - разработка информационных систем на Ada.

Создание **Rational Rose** стартовало в начале 90-х г.г. В 1992 г. вышел первоначальный вариант этого инструмента. Но все же в то время он продавался не очень хорошо. Пик расцвета Rose пришел несколькими годами позднее, когда в нем была реализована нотация визуального моделирования UML. Позже, в 1999 г. был выпущен инструмент для визуального моделирования систем реального времени **Rational Rose RealTime**, который объединил классический Rational Rose с технологией Брана Селика (Bran Selic) из компании ObjectTime Corp, нацеленной на проектирование систем реального времени.

Тем не менее, надо признать, что по настоящему популярным Rational Rose так и не стал. Основной проблемой здесь явилось то, что разработчики для моделирования в Rational Rose совсем не стремились покидать их привычные средства разработки (Integrated Development Environment или IDE), где они свободно оперировали с кодом. По этой причине Rational Rose в конце концов был практически полностью заменен инструментами Rational Software Modeler и Rational Software Architect, созданных на базе платформы Eclipse, архитектура которого изначально ориентирована на максимальную интеграцию инструментов для различных участников проектов: аналитиков, разработчиков, тестировщиков и др.

В 1994-95 г.г. для работы в Rational Software были приглашены специалисты т.н. “банды трех” (“the gang of three”): Гради Буч (Grady Booch), Джеймс Рамбо (James Rumbaugh) и Айвар Джекобсон (Ivar Jacobson). При их прямом участии был определен набор элементов и правил их применения в составе языка визуального моделирования **Unified Modeling Language (UML)** и прообраз **Rational Unified Process (RUP)** – технологии описания процессов от приобретенной компании Objectory. Позднее к Rational Software присоединился Филипп Крачтен (Philippe Kruchten), наработки которого также послужили основой в области разработки базы знаний **Rational Unified Process (RUP)**.

Создание инструмента управления требованиями **Rational RequisitePro** началось в1993 г. в компании Requisite. Впоследствии, в 1997 г.компания Requisite была приобретена компанией Rational Software, а продукт Rational RequisitePro вошел в пакет предложений Rational Software, как одно из наиболее удачных решений, позволяющих эффективно автоматизировать процесс управления требованиями при разработке ПО.

Примерно в это же время Rational Software объединяется с компанией SQA, разрабатывающей инструмент SQA Suite для автоматизации тестирования, и покупает компанию Purify Awareness, разработки которой позволят расширить SQA Suite для обеспечения нагрузочного тестирования. Этот комплексный SQA Suite на долгие годы станет вершиной предложения в составе набора средств Rational TeamTest (который, в свою очередь, вошел в состав Rational Suite Enterprise) для организации функционального и нагрузочного тестирования, где наиболее известными продуктами станут **Rational TestManager**, **Rational Robot** и др.

Также 1997 г. приносит Rational Software новое крайне перспективное приобретение в виде компании Pure-Atria, в результате чего в пакете предлагаемых средств появляется мощнейший инструмент **Rational ClearCase**, предназначенный для управления версиями и конфигурациями, и **Rational Purify** для анализа утечек памяти в коде по методу "белого ящика" с поддержкой наиболее популярных технологий разработки (Java, C, C++). Надо отметить, что **Rational ClearCase** был создан еще в 1992 г. на Unix с функционалом командной строки и только позднее был адаптирован под Windows, где и началось развитие его графического интерфейса пользователя (GUI).

В декабре 1998 г. IBM запускает проект, результатами которого становится базовая платформа для создания средств разработки **Eclipse**. Этот инструмент станет в ближайшие годы основой для создания большинства инструментов разработки ПО от IBM и от Rational Software после приобретения ее компанией IBM. Среди таких инструментов наиболее известными стали средства Rational XDE Modeler и Rational XDE Developer (правда, их жизненный цикл также уже завершен), IBM Rational Software Modeler (RSM), IBM Rational Software Architect (RSA), IBM Rational Application Developer (RAD), IBM Websphere Business Modeler, IBM Websphere Integration Developer, "толстый клиент" IBM Rational Team Concert и др.

Упомянутые выше средства на базе Eclipse IBM Rational Software Modeler и IBM Rational Software Architect для визуального моделирования, IDE разработки IBM Rational Application Developer, инструмент для записи и запуска функциональных тестов IBM Rational Functional Tester (RFT), инструмент для формирования модели нагрузки и ее воспроизведения на тестируемой системе IBM Rational Performance Tester (RPT) и др. были выпущены в конце 2004 г. в составе т.н. **Rational Software Development Platform (RSDP)**. Совместно с традиционными средствами IBM Rational это наиболее был передовой комплект того времени, предназначенный для организации разработки ПО с поддержкой большинства ее процессов.

Если все же вернемся немного назад, то легко вспомним, что в конце 2002 – начале 2003 г.г. компания Rational Software была, в свою очередь, приобретена компанией **IBM** и стала пятым крупным брендом в составе последней наравне с Websphere, Tivoli, Lotus и DB2. А к инструментальным средствам Rational было добавлено наименование IBM и теперь они стали называться IBM Rational.

К сожалению, как я ни старался, мне не удалось найти более подробных сведений о происхождении инструментов **Rational ClearQuest**, **Rational ProjectConsole** и **Rational SoDA**. Известно лишь, что история Rational ClearQuest и Rational SoDA уходит далеко в 90-е годы. При этом Rational ClearQuest изначально позиционировался лишь как средство автоматизации дефект-трэкинга и лишь потом перешел в статус средства управления запросами на изменения (Change Request Management или CRM). Что касается Rational ProjectConsole, то этот инструмент частично базировался на функционале Rational SoDA и был впервые анонсирован Rational Software в составе Rational Suite v2002.

Понятное дело, что значимых событий в Rational Software и IBM было намного больше, чем перечислено здесь, но полагаю, что приведенной здесь информации вполне достаточно, чтобы уже начать делать какие-то оценки и выводы об эволюции предложений обеих компаний. В любом случае мы видим, что вся история формирования традиционной линейки IBM Rational - это классическая история бизнеса большинства успешных компаний мирового уровня, которые активно занимаются приобретением перспективных технологий с целью формирования собственных более конкурентных предложений! Плохо это или хорошо, конечно же, вопрос интересный, но результат очевиден. Данная традиционная линейка от Rational Software сумела занять огромный сектор рынка средств автоматизации разработки ПО и стала на долгие годы эталоном для конкурентов!

***Известные проблемы или как появился Jazz***

А что вообще надо нам, разработчикам? По-видимому, основная цель для нас не следовать слепо каким-то известным методологиям и использовать популярные инструменты, а хорошо делать наше непосредственное дело - прямо или косвенно зарабатывать деньги на разработке программного обеспечения. При этом для нас важно делать свой бизнес эффективно, а чтобы получать больше выгоды при минимальных затратах.

Чтобы компания могла делать свой бизнес эффективнее, каждый ее сотрудник должен эффективнее выполнять свою работу. Но что это означает в более конкретном понимании? Наверно здесь можно перечислить многие условия, которые известны практически каждому из нас из своего собственного опыта:

* Каждый специалист должен полностью вкладываться в работу, за которую ему платят, и можно легко вычислить тех, кто ничего не делает или делает мало и смотрит видеофильмы в рабочее время
* Необходимо максимально уменьшать вероятность переделок и повторных затрат на реализацию одного и того же решения или его части
* Надо максимально распараллеливать работу исполнителей и облегчать их взаимодействие, исключая неэффективные простои
* Обеспечить руководителя возможностью всегда быть в курсе событий, чтобы тот смог быстро идентифицировать потенциальные проблемы и оперативно и грамотно на них отреагировать
* Минимизировать число исполнителей на выполнение конкретной работы и добиваться достижения наибольших результатов наименьшими силами
* Каждый участник проекта должен четко видеть все, что творится в проекте, и понимать всю важность своей работы, всю свою ответственность за общий успех, а информация должна доводиться до исполнителей четко и быстро
* Минимизировать рутину и повышать степень автоматизации для повторяющихся операций и, таким образом, содействовать развитию творческого начала, что в силу специфических особенностей строения человека способно резко поднять качество выполняемых операций
* Обеспечить руководителям свободу принятия решений в каждом проекте, в том числе в выборе инструментария, применять наиболее эффективные в конкретном проекте средства и не быть связанными по рукам и ногам инструментами от одного вендора
* И т.д.

Вне всяких сомнений, первостепенным фактором для обеспечения успеха всего проекта является грамотная организация работы или заложенные в основе процесса методологические принципы. Но в дополнение к правильно организации работы очень эффективной может стать и грамотная автоматизация процесса. При этом, как и в случае с методологией, можно определить некоторые базовые принципы эффективной автоматизации, которые, в целом, не зависят от выбранной методологии или определенных в процессе дисциплин. Давайте попытаемся сформулировать такие принципы успешной автоматизации, которые будут максимально содействовать повышению эффективности нашего бизнеса, претворяя в жизнь уже сформулированные нами выше положения:

* средства автоматизации должны быть **максимально интегрированы** и позволять создавать связанные друг с другом проектные артефакты, чтобы можно было по цепочке отследить причину создания артефактов и поиск уже существующих наработок
* интеграция средств должна обеспечивать **"прозрачность" данных**, чтобы с данными, относящимися к одному инструменту, можно было работать из любого другого средства без необходимости развертывания первого на каждом рабочем месте
* средства должны иметь **унифицированные интерфейсы GUI**, чтобы минимизировать затраты на обучение специалистов или зависимость от конкретных, наиболее продвинутых исполнителей
* средства, включенные в автоматизированную систему, должны использовать **единые механизмы развертывания и администрирования** (настройки, управления пользователями и правами, лицензирования и т.д.)
* средства автоматизации должны быть **легко адаптируемыми к увеличению размеров команд**, чтобы обеспечить эффективное взаимодействие любого числа участников в проектах любого размера
* должны быть **унифицированы правила добавления нового инструментария** в среду, чтобы исключить зависимость от конкретного вендора и обеспечить возможность "бесшовного" подключения любых инструментов, соответствующих некоторым предопределенным стандартам

Непрерывный поиск Rational Software наиболее оптимальных путей и приобретение готовых продуктов для расширения пакета собственных предложений, с одной стороны, позволили достаточно быстро создать эффективное решение в области автоматизации разработки ПО, которое мы в рамках данной статьи назвали традиционной линейкой IBM Rational. С другой стороны, такая политика предопределила недостаточную гибкость задействованных инструментальных средств, как уже не отвечающую духу сегодняшнего времени. Архитектура, собранная по принципу мозаики не позволяла справиться с проблемами и ограничениями, среди которых следует отметить:

* Разносторонние и разбросанные механизмы интеграции, т.к. для связывания этих средств понадобилась разработка уникальных мостов с учетом специфики каждого инструмента
* Специфичные данные для каждого инструмента, когда возможности доступа к данным других средств определены лишь в рамках ограниченного функционала конкретного интеграционного моста
* Отсутствие единообразия графических интерфейсов пользователей GUI в инструментах, заметно отличающихся от средства к средству
* Разбросанные физически по разным инструментам, но при этом дублирующие друг друга по смыслу функции администрирования (управление пользователями и правами, создание и настройка репозиториев и т.п.)
* ориентированность интеграционных механизмов на одного вендора (IBM, Rational Software) и зависимость от компаний, которые разрабатывали мосты для встраивания в среду сторонних средств (лично мне здесь сразу же вспоминается классическая связка 10-летней давности из собственного программистского прошлого: Rational Rose - Rose Link for Delphi от компании Ensemble Systems - Borland Delphi)

Таким образом, в некоторый момент времени компания IBM столкнулась со значительными архитектурными ограничениями существующих инструментов. В результате IBM переориентировалась на создание абсолютно новой платформы, в которой максимально полно учитывались веяния времени и пожелания мирового сообщества разработчиков ПО. Теперь ставка была сделана на разработку эффективной архитектуры, а уже потом – на ее реализацию, а не наоборот, как это ранее вынужденно делала Rational Software, стремясь захватить свою долю на рынке.

Так зародилась идея новой платформы в области управления разработкой ПО, которую сегодня нам известна под брендовым наименованием Jazz. Коренным отличием Jazz от традиционных технологий IBM Rational является то, что средства этой платформы базируются на собственных разработках IBM и продуктах Open Source.

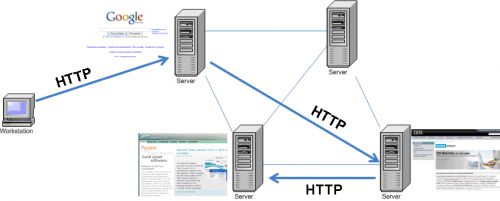
Название "Jazz" и сама идея участия в проекте в рамках этой технологии очень тесно перекликаются с представлением о функционировании музыкальных групп, играющих музыку в стиле Jazz. Вполне вероятно, что среди зачинателей этого направления в IBM были огромные поклонники музыкального стиля Jazz, которые и предложили название для нового бренда. Дело в том, что хорошая джаз-банда (jazz band) в своей работе опирается на следующие ключевые концепции:

* **прозрачность** (каждый участник отлично видит, что делают другие)
* **взаимодействие** (все отлично чувствуют друг друга и прекрасно взаимодействуют)
* **осведомленность** (каждый знает, что должен делать именно он)

При этом наличие хорошего инструментария значительно усиливает возможности группы создавать шедевры в своей области.

А вот архитектурным прототипом для Jazz послужило что-то такое, что уже давно существует рядом с нами. То, без чего нам уже трудно представить наш мир. Я имею в виду Интернет или всемирную информационную паутину! Что же такого особенного в Интернете, что его можно использовать в качестве прототипа архитектурного решения?

В Интернете существует множество всевозможных сайтов, а на этих сайтах – огромное число разнообразных Web страниц. Можно легко расширить сеть, добавляя новые сайты или новые страницы на отдельных сайтах. При этом совершенно не стоит опасаться того, что сеть не выдержит нагрузки или рухнет из-за ошибок на одном из сайтов!

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/bdb07e27c3cedf484605dc7efb71970e.gif)

**Рисунок 3. Интернет - прототип Jazz**

Любые Web страницы идентифицируются по URL (Uniform Resource Locator) и могут включать гиперссылки на другие Web страницы на любых Web сайтах. При этом совсем не важно, где физически размещены эти Web-сайты и Web-страницы, а важно лишь наличие подключения к сети через одного из ее участников! Можно даже полностью перестроить любой сайт и его интерфейс GUI, но при сохранении старых URL для Web-страниц мы гарантируем, что пользователи по-прежнему будут попадать на эти страницы и работать с опубликованными на них данными.

Правила наполнения содержанием Web-страниц стандартизованы. По страницам можно свободно перемещаться, не задумываясь, где (или на каком сайте) Вы находитесь и куда окончательно попадете в результате своего путешествия по громадному информационному пространству. При этом Интернет можно также сравнить с роем пчел, который будет сохранять свою функциональность, даже если какие-либо из пчел умрут и по любым причинам прекратят свое существование.

Специальная программа (т.н.Web-браузер) отображает Web страницу в рамках стандарта HTML (HyperText Markup Language) и позволяет пользователям перемещаться (осуществлять навигацию) по страницам.

Весь описанный выше опыт, связанный с продвижением продуктов традиционной линейки IBM Rational и бурным ростом Интернета, и привел к появлению технологии Jazz.

***Так что же такое Jazz?***

В технологии Jazz определены три ключевые компоненты:



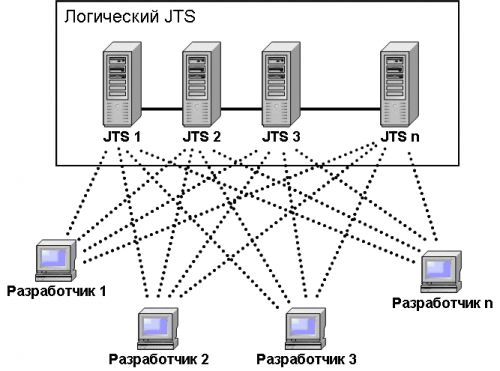
* **Архитектурное решение**, определяющее набор унифицированных стандартов с учетом возможного расширения набора средств
* **Базовый набор инструментов**, реализованных в строгом соответствии с этими архитектурными стандартами
* **Сообщество пользователей**, от которых аккумулируется обратная связь, двигающая и направляющая разработку пакета средств автоматизации Jazz

**Архитектурное решение**

Архитектурное решение Jazz реализовано на принципах открытости платформы и гибкой адаптации к уже существующим условиям разработки.

Jazz – это клиент-серверная система. Центральным элементом в ней является **Jazz Team Server (JTS)**. В зависимости от специфики деятельности для работы с JTS можно использовать либо “толстые клиенты” (клиентские средства, реализованные на базе Eclipse), либо “тонкие клиенты” (Web-браузеры, взаимодействующие с Web-приложениями, развернутыми на JTS).

При необходимости можно развернуть любое число JTS, которые могут быть интегрированы друг с другом. Таким образом, достигается расширяемость автоматизированной системы до корпоративного уровня, когда в проекте может принимать участие множество удаленных друг от друга команд с любым числом разработчиков.

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/52a84c6352ad5e07bf2e98303015a663.gif)

**Рисунок 4. Архитектура логического сервера JTS**

В единой системе все JTS интегрированы друг с другом. Разработчики могут подключаться к любому JTS, но при этом смогут работать с данными других JTS, если, конечно, им разрешен к ним доступ. Т.е. в данном случае вся группа JTS может рассматриваться, как своеобразный логический JTS, состоящий в реальности из нескольких физических серверов. При этом система может быть снова дополнена любым числом новых JTS или могут быть отключены существующие.

На каждом JTS могут быть сохранены пользовательские данные, которые могут быть связаны с любыми данными других JTS. По этой причине исключение физических JTS приведет лишь к отключению хранимых на них данных, но никак не к краху всей системы.

Давайте углубимся в детали построения отдельного JTS. Любой JTS включает хранилище данных и логический уровень в виде набора Web-сервисов, которые определяют доступные над данными операции.

Все Web-сервисы работают на принципах концепции Representation State Transfer (RESTful). Данная концепция была впервые провозглашена в 2000 г. при защите докторской диссертации Роем Томасом Филдингом (Roy Thomas Fielding) - ученым из Калифорнии, ведущим архитектором протокола HTTP и основателем ряда проектов Open Source, включая Apache HTTP Server Project. Основная идея RESTful заключается в акценте не на функции, а на ресурсы. Среди ключевых постулатов:

* Каждый ресурс имеет уникальный идентификатор URI (Uniform Resource Identifier)
* Все ресурсы разделяют унифицированный интерфейс для передачи его клиентам (включает ограниченный набор унифицированных операций и базовых типов данных)
* Используется многослойный клиент-серверный протокол, не зависящий от состояния и включающий функции кэширования

По аналогии с типовыми операциями стандарта CRUD, определенных над базами данных, в RESTful определен набор базовых типов операций: POST (аналог в CRUD - CREATE), GET (READ), PUT (UPDATE) , DELETE (DELETE).

На базе принципов RESTful IBM провозгласила промышленную инициативу OSLC (Open Services for Lifecycle Collaboration), основная цель которой – унификация механизмов взаимодействия различных инструментов управления разработкой ПО. Данная инициатива описывает архитектуру и набор Web-протоколов и Web-сервисов для интеграции автоматизирующих средств.

Но вернемся снова к JTS. В JTS определены две группы Web-сервисов:

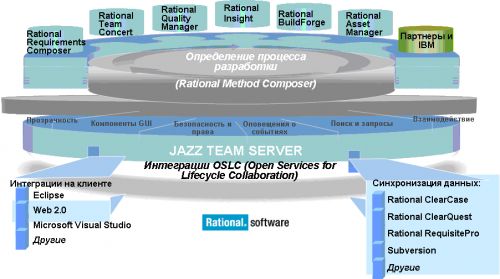
* **Общие Web-сервисы Jazz Foundation Services (JFS)**, которые входят в состав любого JTS
* **Специфические Web-сервисы JTS Extensions**, которые определяют функционал, добавляемый в систему при добавлении специфических инструментов

При этом в базовый набор JFS входят:

* **Сервис обнаружения (Discovery Service)**, который описывает, какие именно сервисы REST API определены в системе, на каких физических JTS они размещены и как к ним правильно обращаться. Это внутренние сервисы Jazz технического уровня.
* **Сервисы администрирования (Administration Services)**, которые определяют набор операций для унифицированного управления пользователями и их правами, управления проектными репозиториями, управления лицензиями и т.д.  
  Здесь надо обязательно отметить акцент на ролевые принципы предоставления доступа к функциям Jazz, когда любому пользователю определяются одна или несколько ролей (роли могут добавляться или удаляться администратором с установкой необходимых прав для любых ролей) и, в соответствии с этими ролями, пользователям предоставляются права на выполнение операций. При этом следует отметить, что в различных проектах одни и те же пользователи могут получить разные роли (например, в одном проекте можно стать руководителем проекта и обычным аналитиком в параллельном проекте).
* **Сервисы процессов (Process Services)**, которые позволяют реализовать специфическую организацию взаимодействия участников проектов, определить роли и доступные им операции, настроить проект с учетом конкретных условий, определить типы артефактов разработки, формы их представления, доступные операции над артефактами и т.д. Некоторые инструменты Jazz приходят с более жесткими предопределенными шаблонами процесса (например, в IBM Rational Quality Manager заочно определены такие типы данных, как планы тестирования, сценарии тестирования, логи и т.д.), а некоторые с более гибкими возможностями его настройки (например, в IBM Rational Team Concert присутствует очень мощный редактор процессов, с помощью которого можно выполнить все настройки "с нуля" и фактически адаптировать инструмент для любых условий разработки).
* **Сервисы управления данными (Storage Services)**, которые являются промежуточным интерфейсом для работы с хранилищами данных и позволяют унифицировать формы получения и сохранения данных независимо от типа СУБД. В частности, Jazz поддерживает разнообразные конфигурации таких СУБД, как Derby, IBM DB2, Microsoft SQL Server и Oracle.
* **Сервисы запросов (Query Services)**, которые позволяют унифицировать работу с данными и выполнять пользовательскую настройку фильтрации и поиска любых типов данных JTS по требуемым правилам. Пользователь может определить собственные запросы с указанием необходимых параметров фильтрации и, таким образом, удобно организовать собственное рабочее пространство.
* **Сервисы представления (Presentation Services)** определяют правила построения пользовательских форм GUI для клиентских средств (при этом нет разницы, “толстый” это или “тонкий” клиент и какой инструмент используется для подключения к JTS). Например, наведение мыши на некоторый элемент данных в любом средстве Jazz (клиенте Eclipse, Web-браузере), интегрированном в Jazz, позволит временно вывести типовое всплывающее окно с дополнительной информацией об этом элементе.
* **Сервисы хранилища данных (Data Warehousing Services)**, которые задействуются в большей степени для мониторинга системы, составления отчетов и организации работы с временными аналитическими данными на принципах OLAP (Online analytical processing).
* **Сервисы взаимодействия (Collaboration Services)**, обеспечивающие реализацию коммуникативных механизмов, включая функции рассылки Email, оповещений, поддержку механизма подписки на конкретные события в проекте и т.д.

**Набор инструментов**

Как мы уже упоминали, Jazz не навязывает конкретный набор инструментов, что было характерно ранее в рамках традиционных предложений. Совершенно любые не-IBM инструменты, реализующие принципы OSLC, могут быть “бесшовно” встроены в автоматизированную систему на платформе Jazz. Но при этом, разумеется, IBM предоставляет и базовый набор инструментов, из которого любые средства могут быть использованы в Ваших проектах. Давайте немного подробнее остановимся на этих инструментах.

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/c1ca0715b6ef6546b66cafba002b8f6e.gif)

**Рисунок 5. Типовая автоматизированная система на платформе Jazz**

В линейку базовых инструментов Jazz входят (по состоянию на начало 2011 г.):

* **IBM Rational Requirements Composer (RRC)**, который является средством для выявления требований различных типов (в том числе, пользовательских), создания необходимых документов (с помощью встроенного редактора или Microsoft Word), простого визуального моделирования бизнес-процессов на базе стандартов Business Processes Modeling Language (BPML), моделирования сценариев использования с помощью Unified Modeling Language (UML), использования принципов раскадровки (storyboards), разработки прототипов графического интерфейса пользователя GUI, командного согласования и утверждений результатов работы и т.д. Очень сильно автоматизировано создание словарей, т.к. добавление в них слов возможно из различных описаний артефактов и документов.  
  Аналогом данного инструмента в составе традиционной линейки IBM Rational является IBM Rational RequisitePro.
* **IBM Rational Team Concert (RTC)**, который является “первым среди первых” инструментов, предназначенным для использования всеми участниками проектной команды. Боле того, благодаря IBM Rational Team Concert к проекту могут быть подключены любые заинтересованные лица, такие как заказчики и конечные пользователи. В непосредственные функции IBM Rational Team Concert (т.е. функции, реализованные с помощью дополнительных Web-сервисов Jazz Extensions для RTC) входят управление рабочими заданиями (workitems) различных типов (например, задачами, дефектами, запросами на улучшения и пользовательскими типами данных, определенными при проектировании конкретного процесса) с возможностью настройки жизненных циклов запросов (состояний и переходов между состояниями) и интерфейсных форм GUI, планирование и управление проектом с акцентом на итерационные практики, управление версиями и конфигурациями, возможности для автоматизации проектной сборки, настройка и адаптация шаблонных процессов из поставки IBM Rational Team Concert и создание собственных шаблонов процессов.  
  В некотором роде аналог IBM Rational Team Concert в традиционных средствах - совместный дуэт продуктов IBM Rational ClearQuest и IBM Rational ClearCase.
* **IBM Rational Quality Manager (RQM)**, который предназначен для автоматизации тестирования при разработке ПО. Здесь можно создать план тестирования, определить сценарии тестирования и сгруппировать их в наборы, подключить необходимые скрипты (причем скрипты для проведения пошагового ручного тестирования можно создавать прямо в RQM), знакомиться с результатами тестирования и создать дефекты при обнаружении проблем в ходе выполняемой проверки. Поддерживается запуск скриптов автоматизированного тестирования и возвращение результатов с помощью IBM Rational Functional Tester, IBM Rational Performance Tester, IBM Rational Tester for SOA Quality, IBM Rational Robot, любых проигрывателей тестов в режиме взаимодействия через командную строку и даже средств Hewlett Packard (Mercury) - HP Load Runner, HP Quick Test Pro.  
  Аналог IBM Rational Quality Manager в традиционной линейке - IBM Rational TestManager.
* **IBM Rational Insight**, созданный на базе продуктов Cognos и предназначенный для повышения прозрачности проектов разработки и информированности руководителей о текущих состояниях проектов. В частности, грамотно развернутый инструмент IBM Rational Insight предоставляет возможности точно измерять эффективность процесса разработки и используемых практик с целью непрерывного улучшения самого процесса, получать объективную картину о текущем положении дел в проектах с целью их оперативной коррекции и снижения рисков разработки, вычисления разнообразных показателей эффективности (Например, ROI).  
  Аналог IBM Rational Insight в традиционной линейке IBM Rational - IBM Rational ProjectConsole.
* **IBM Rational BuildForge**, благодаря которому можно организовать сборку на уровне целого предприятия в мультипроектной среде, запускать процессы сборки в необходимое время, контролировать результаты каждой сборки, а затем выполнять запуск необходимых тестов с помощью IBM Rational Quality Manager по результатам сборок и т.д.  
  Аналогов данного средства в традиционной линейке IBM Rational нет.
* **IBM Rational Asset Manager (RAM)**, который является мощным решением для организации управления шаблонами, компонентами и другими наработками в ходе разработки. С помощью IBM Rational Asset Manager можно сохранить практически любые наработки, весь опыт организации и обеспечить возможности их быстрого поиска и повторного использования вместо того, чтобы заново разрабатывать уже существующие решения. В частности, существующие в организации наработки могут быть представлены в виде исходных текстов, документов, Web-сервисов и др. Из нескольких вариантов могут быть выбраны наиболее подходящие в данный момент времени решения. Сначала можно загрузить уже существующие наработки для более эффективного старта своей собственной работы, а после ее выполнения делать свои результаты доступными для других. Доступно также управление жизненными циклами наработок.  
  Аналогов данного средства в традиционной линейке также не существует.

Вне всяких сомнений, отдельного высказывания заслуживает инструмент проектирования и описания процессов **IBM Rational Method Composer**, который напрямую не входит в набор средств Jazz, но, тем не менее, играет важную роль при совместном использовании с новыми средствами. Благодаря ему можно не просто создать описание процесса разработки в виде Web-сайта, но и выполнить проектирование процесса, создать шаблон рабочих заданий и экспортировать эти задания с полными описаниями в IBM Rational Team Concert. Далее руководитель проекта может использовать эти рабочие задания для построения плана проекта и определения для них ответственных, в результате чего разработчики получат назначения с полным описанием выполнения задач, с начальными шаблонами артефактов и другими дополнительными материалами.

Вообще, следует отметить, что практически все средства традиционной линейки IBM Rational тесно интегрированы со средствами Jazz на случай, если первые уже развернуты в организации.

При развертывании автоматизированной системы на платформе Jazz можно использовать любое из перечисленных выше средств. Но при этом никто не заставляет в обязательном порядке использовать их. Уже сегодня доступно множество плагинов, связывающих базовые средства Jazz с другими инструментами. Среди таких средств, интегрированных с Jazz с помощью плагинов, можно перечислить самые разнообразные средства, такие как Microsoft Visual Studio, JIRA, iRise, CVS, SVN, Ravenflow, Maven, Hudson build system, CruiseControl build system, HP Quality Center и др.

Разумеется, интегрировать с Jazz можно и не-Jazz средства IBM. Среди них следует отметить IBM Rational Software Architect, IBM Rational Software Modeler, IBM Rational Functional Tester, IBM Rational Performance Tester, IBM Rational RequisitePro, IBM Rational ClearQuest, IBM Rational ClearCase и другие.

Кроме всего прочего, следует отметить крайне интересную разработку [**RTC Email Reader**](http://www.finecosoft.ru/rtcemailreader) - продукт отечественной компании “Финэкософт”, позволяющий настроить и автоматизировать получение Email сообщений от удаленных пользователей или систем. Напомню, что в традиционных средствах то же самое делал ClearQuest Email Reader для занесения данных в продукты IBM Rational ClearQuest и IBM Rational RequisitePro в различных случаях (при получении оповещений об ошибках в удаленном ПО, получении запросов от пользователей при отсутствии прямого доступа к системе, организации дискуссий и обсуждений и т.д.). RTC Email Reader может быть использован в основе формирования интеграционных связей с внешними инструментами, которые не поддерживают стандарты OSLC, но могут посылать Email сообщения в настраиваемом формате.

Следует отметить ориентированность средств Jazz на Интернет. IBM Rational Team Concert наряду с мощным клиентом Eclipse имеет продвинутый Web-интерфейс, а IBM Rational Requirements Composer (начиная с версий 3.x), IBM Rational Quality Manager, IBM Rational Asset Manager и IBM Rational BuildForge являются полноценными Web-приложениями, с которым взаимодействие пользователя осуществляется через “тонкого клиента”.

Несколько слов здесь обязательно стоит упомянуть о подходе **Collaborative Application Lifecycle Management (CALM)**, провозглашенном IBM в рамках технологии Jazz. CALM – это способ объединения команды на основе средств автоматизации в единый функционирующий конвейер, где все его участники четко выполняют свои обязанности, эффективно осуществляют передачу результатов своей деятельности другим исполнителям и акцентированы на быстром достижении качественного конечного результата. В целом, данный подход позволяет реализовать типовую цепочку операций в рамках отдельной итерации, как это примерно представлено на следующем рисунке (Рисунок 8).

[](https://finecosoft.ru/sites/default/files/images/ratiojazz_utopia/15a995f560bee6a60906963da89b41f1.gif)

**Рисунок 6. Collaborative Application Lifecycle Management в Jazz**

Отдельную итерацию разработки можно представить следующим образом. Исходной точкой являются начальные запросы, полученные с помощью IBM Rational Team Concert. Здесь выполняем первичную обработку запросов, расставляем их по приоритету и передаем для дальнейшего анализа в IBM Rational Requirements Composer. В последнем создаются новые и уточняются существующие требования к разрабатываемой системе и затем для реализации этих требований создаются задачи в IBM Rational Team Concert. Далее формируем новый или уточняем существующий план проекта, разносим задачи по итерациям с учетом доступных ресурсов (вполне вероятно, что какие-то менее приоритетные задачи уйдут на более поздние итерации) и распределяем эти низкоуровневые задачи между конкретными исполнителями. Архитекторы, программисты и другие разработчики в соответствии с назначенными задачами создают работоспособные решения, а затем результаты их работы вместе с определенными ранее требованиями передаются в IBM Rational Quality Manager. Тестировщики проверяют программные продукты или их отдельные компоненты на соответствие требованиям и, при необходимости, формируют запросы на устранение проблем (или описания дефектов) в IBM Rational Team Concert. Цикл замыкается и все повторяется снова.

Напоследок обязательно стоит упомянуть о том, что функционал продукта IBM Rational Team Concert абсолютно бесплатно доступен для небольших команд в размере до 10 человек. Любой желающий его может скачать с Jazz.net, развернуть в своей команде и использовать без функциональных ограничений (начиная с версий 3.x).

**Сообщество пользователей**



Jazz - это также и огромное сообщество специалистов по всему миру, занимающихся или интересующихся разработкой программного обеспечения.

Основной ресурс платформы Jazz -- [http://jazz.net](http://jazz.net/). Вы можете свободно работать с этим ресурсом, с его материалами, участвовать в формировании запросов на разработку средств автоматизации на платформе Jazz и обсуждать технологию на форумах. Максимум, что от Вас могут при этом попросить - лишь зарегистрироваться на ресурсе и не более того!

***Окончательное сравнение или что не хватает для полного счастья***

Теперь сделаем небольшую сравнительную таблицу для окончательного подведения итогов анализа и обобщения ключевых отличий Jazz от традиционного направления IBM Rational. Постараемся при этом учесть и плюсы, и минусы обоих направлений.

**Таблица 1. Таблица сравнения возможностей традиционных средств IBM Rational и Jazz**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Возможность** | **Традиционная линейка IBM Rational** | **Платформа Jazz** |
| 1 | Поддержка ключевых процессов жизненного цикла разработки | Поддерживаются бизнес моделирование, управление требованиями, управление изменениями и конфигурациями, тестирование, управление проектом. Кроме того, поддерживается анализ и проектирование. | Поддерживаются бизнес моделирование, управление требованиями, управление изменениями и конфигурациями, тестирование, управление проектом. Кроме того, поддерживаются процессы управления сборкой и наработками (assets). |
| 2 | Интеграционные возможности | Интеграции между средствами реализуются посредством разработки специфических мостов между конкретными парами инструментов. | Правила интеграции унифицированы на серверном уровне JTS. |
| 3 | Представление интерфейсов GUI | Каждое средство и каждый интеграционный мост определяют собственные правила представления интерфейсов GUI. | JTS определяет общие правила для представления интерфейсов GUI. |
| 4 | Разделение данных | Доступ к данным возможен только между двумя конкретными средствами, интегрированными с помощью мостов. | Средства, интегрированные в среду Jazz, используют единое пространство данных и могут оперировать с данными друг друга. |
| 5 | Унификация механизмов администрирования | Уникальные механизмы администрирования, специфические для каждого инструмента. | Механизмы администрирования унифицированы на серверном уровне JTS. |
| 6 | Адаптация к увеличению размеров команд | Ограниченная. | Расширяемость платформы заложена при проектировании архитектуры Jazz. |
| 7 | Взаимодействие удаленных команд | Продвинутые Web-интерфейсы и механизмы репликации Multisite. | Мощные Web-приложения. |
| 8 | Взаимодействие удаленных команд, находящихся в закрытых сетевых пространствах или сетевых пространствах с крайне ограниченным доступом | Репликации Multisite. | Не поддерживается. |
| 9 | Русскоязычная локализация | Отсутствует. | Реализована. |

Теперь сделаем выводы по результатам сравнения, приведенным в данной таблице:

* В области **поддержки ключевых процессов жизненного цикла разработки** и традиционная линейка IBM Rational, и Jazz во многом похожи. Поддерживаются практически все ключевые процессы разработки. В традиционной линейке они расширены дисциплиной анализа и проектирования за счет средства визуального моделирования IBM Rational Rose. Средства IBM Rational Software Architect и IBM Rational Software Modeler напрямую не относятся ни к одной из линеек, но, тем не менее, интегрируются с инструментальными средствами обеих. Правда надо отметить, что интеграция RSA с IBM Rational RequisitePro на порядок мощнее, чем менее гибкие операции импорта и экспорта в связке RSA - RRC.  
  Кроме того, в Jazz появляются процессы управления сборкой на базе IBM Rational BuildForge и управления наработками на базе IBM Rational Asset Manager.
* В области **интеграционных возможностей** инструменты традиционных средств могут работать только с данными друг друга. При этом обновление инструментов может потребовать необходимость выпуска обновлений для интеграционных мостов. А вот подключение к JTS любого стороннего средства, в котором соблюдены принципы OSLC, позволяет легко его интегрировать с другими средствами, уже подключенными к этому JTS. Таким образом, можно расширять инфраструктуру проекта даже за счет Open Source решений. Т.е. в данном пункте полное преимущество за Jazz.
* В области **представления интерфейсов GUI** Jazz также заметно опережает традиционную линейку IBM Rational. Любое средство, интегрированное с JTS, предоставляет полную информацию о корректном отображении собственных данных. Соответственно, любое иное средство, интегрированное с JTS, уже знает, как выводить пользователям эти данные.
* В области **разделения данных** снова полное преимущество за Jazz, благодаря более продвинутой архитектуре. Интеграция средства в среду Jazz позволяет разделять данные с уже подключенными средствами.
* В области **унификации механизмов администрирования** снова полное преимущество за Jazz. Согласитесь, что лучше один раз выполнить базовые настройки независимо от специфики инструментальных средств, чем повторять эту работу для каждого отдельного инструмента. В частности, лучше один раз завести пользователей и определить их права, чем многократно повторять эти операции в каждом инструменте, к которых к тому же значительно отличаются интерфейсы GUI!
* **Адаптация к увеличению размеров команд** в пользу Jazz. В большинстве случаев в традиционной линейке IBM Rational адаптация к росту команд возможна лишь в рамках повышения производительности аппаратных средств и СУБД. Хотя надо отметить, что IBM Rational ClearCase имеет более гибкую архитектуру в сравнении со своими собратьями по традиционной линейке. Более того, эта архитектура крайне похожа на архитектуру Jazz, когда можно "бесшовно" объединить любое число физических серверов в виде своеобразного логического сервера и при этом обычные пользователи отметят лишь повышение производительности всей системы. Кроме того, IBM Rational RequisitePro позволяет создавать проектные репозитории со связанными друг с другом межпроектными данными.  
  А вот в ключевых средствах Jazz при росте числа разработчиков можно создать большое число JTS и объединить их в рамках единого логического сервера.
* **Взаимодействие удаленных команд** реализовано в обеих линейках. Традиционные средства являются изначально Windows-приложениями, хотя для IBM Rational ClearCase и IBM Rational ClearQuest существуют продвинутые конфигурации для иных операционных систем (например, различных вариантов Unix). Для IBM Rational RequisitePro, IBM Rational ClearCase и IBM Rational ClearQuest существуют богатые Web-интерфейсы.  
  Что касается Jazz, то изначально средства данной платформы ориентированы на Web. Даже клиентские приложения, разворачиваемые локально взаимодействуют с серверами по протоколам HTTP или HTTPS в зависимости от определенных правил безопасности. В частности, если рассматривать ключевые средства Jazz, то начиная от версий 3.x все они являются Web-приложениями. Правда, надо отметить, что для IBM Rational Team Concert существует богатый Eclipse-клиент, благодаря которому его можно использовать совместно практически с любым IDE (даже при отсутствии прямых интеграционных плагинов).
* Для **взаимодействия удаленных команд с ограниченным доступом** в рамках традиционной линейки разработана технология репликации Multisite, которая позволяет синхронизировать репозитории IBM Rational ClearQuest и IBM Rational ClearCase (а вместе с ними и все остальные репозитории и артефакты) по заданному расписанию посредством посылок синхронизирующих реплик в виде прямой IP-передачи, FTP, Email, на сменных носителях и т.д.  
  В Jazz возможность объединения закрытых команд не поддерживается, т.к. требуется прямой коннект между клиентами и JTS. Выходом здесь может стать организация специфических, закрытых для остального мира информационных каналов и, чисто теоретически, - собственные решения с использованием встроенных в СУБД механизмов репликации данных.
* **Русскоязычная локализация** изначально предусмотрена в традиционной линейке не была и, соответственно, не реализована. А вот Jazz - действительно мультиязычная платформа! Российский рынок для многих компаний, включая IBM, сегодня является крайне привлекательным и, по этой причине, начиная уже с версий 2.x продуктов были доступны русскоязычные IBM Rational Team Concert и IBM Rational Quality Manager, а начиная с версии 3.x к ним присоединился IBM Rational Requirements Composer. Т.е. переход на новую платформу Jazz максимально упрощен также и для русскоязычного населения!

Также надо отметить, что дополнительно существуют механизмы импорта и синхронизации данных IBM Rational ClearCase и IBM Rational ClearQuest с IBM Rational Team Concert, импорт данных из IBM Rational TestManager в IBM Rational Quality Manager и синхронизация данных IBM Rational RequisitePro с IBM Rational Requirements Composer.

***Заключение***

Какое же заключение можно сделать по результатам проведенного сравнительного анализа. Вне всяких сомнений, Jazz - более продвинутое и в архитектурном смысле более грамотное решение. Но все же ему еще предстоит доказать на практике, что Jazz - это более эффективное решение. Как говорится "Скоро сказка сказывается, да не скоро дело делается".

Преимуществом IBM Rational является возможность автоматизировать практически все ключевые области разработки и обеспечить создание связанной проектной информации независимо от числа участников. Крайне важен здесь постоянный акцент IBM на максимальная интеграция разработчиков в единый живой организм, целью которого является непрерывный выпуск и повышения качества создаваемых программных продуктов.

С точки зрения простоты развертывания и настройки разнообразных инструментов Jazz намного легче, но это достигается лишь за счет объединения административных операций для всех используемых инструментов в рамках JTS. Тем не менее, представляется достаточно проблематичным грамотно развернуть платформу Jazz в организации либо без приглашенных профессиональных консультантов, либо без группы собственных специалистов, специально выделенных для изучения направления и последующего сопровождения среды.

Jazz гораздо терпимее относится к сторонним средствам, даже если они созданы другими вендорами, чем мы это наблюдаем в традиционной линейке. Более того, Jazz активно пропагандирует и подталкивает к применению средств Open Source. Главная цель при этом, чтобы это было непосредственно полезно самим проектам! По этой причине платформа Jazz - намного более перспективна, особенно, если в Вашей организации уже существует мультипроектная среда с большим числом используемых технологий и средств.

В числе важнейших достоинств Jazz - акцент на ролевые принципы работы. Роли (менеджер, аналитик, разработчик, тестировщик или любые другие) присваиваются участникам проекта и в соответствии с этими ролями настраиваются доступные операции над артефактами. При этом список ролей может быть гибко скорректирован, в частности, созданы новые роли, позволяющие на практике реализовать свой собственный процесс разработки ПО.

Но при всех преимуществах и перспективах Jazz, разумеется, не стоит сразу же срываться и бежать в магазин за предлагаемым инструментарием. Прежде всего, необходимо внимательно проанализировать, что уже у Вас имеется в наличии, есть ли какие-то проблемы и в чем они состоят, оценить пути улучшения ситуации при наличии проблем и просчитать финансовые выгоды или потери при модернизации технологий управления разработкой. Исходя из собственного опыта, могу дать лишь один совет - старайтесь найти опытных людей, много лет специализирующихся в области оптимизации и выстраивания процессов разработки и их последующей автоматизации. Это могут быть как специалисты консалтинговой компании, которых можно пригласить на временной основе, или собственные специалисты, нанятые для постоянной работы. Такие специалисты на основе своего опыта способны быстро оценить ситуацию, провести эффективный анализ текущей ситуации и порекомендовать наиболее грамотное решение, которое может избавить Вашу организацию от повторения уже известных ошибок и "изобретения велосипеда".

***Ссылки***

1. [Jazz Community Web Site](http://jazz.net/)
2. [A brief history of Eclipse](http://www.ibm.com/developerworks/rational/library/nov05/cernosek/index.html) (Gary Cernosek, Market Manager, IBM)
3. [The CBS Interactive Business Network](http://findarticles.com/)
4. [Kongsky's Image at Freedigitalphotos.net](http://www.freedigitalphotos.net/images/Money_g61-Stacks_Of_Coins_p30343.html)
5. [Salvatore Vuono's Image at Freedigitalphotos.net](http://www.freedigitalphotos.net/images/Music_g290-Piano_p27016.html)
6. [Xedos4's Image at Freedigitalphotos.net](http://www.freedigitalphotos.net/images/Internet_g170-Earth_p22862.html)

\*\*\*

В [Каталоге Центра IT обучения "Institutio"](http://www.institutio.ru/catalog/) представлено несколько программ обучения по направлению Jazz и отдельным его инструментальным средствам, отличающихся друг от друга включенными в них модулями:

1. PCOM0001, [Обзор технологии Jazz и основных инструментальных средств](http://www.institutio.ru/catalog/product_info.php?manufacturers_id=16&products_id=25) (общий обзор технологий Jazz и ключевых инструментальных средств), 4 дня
2. PCCM0001, [Введение в IBM Rational Team Concert](http://www.institutio.ru/catalog/product_info.php?cPath=17&products_id=26) (начальный обзор инструментального средства с целью максимально быстро научиться применять его на практике), 1 день
3. PCCM0002, [Возможности IBM Rational Team Concert](http://www.institutio.ru/catalog/product_info.php?cPath=17&products_id=29) (начальный обзор инструментального средства и его расширенные интеграционные возможности для организации командной разработки), 2 дня
4. PCCM0003, [Автоматизация командной разработки ПО с помощью IBM Rational Team Concert](http://www.institutio.ru/catalog/product_info.php?cPath=17&products_id=27) (введение в дисциплину управления версиями и конфигурациями, начальный обзор RTC и его расширенные интеграционные возможности для организации командной разработки), 3 дня
5. PREQ0001, [Введение в IBM Rational Requirements Composer](http://www.institutio.ru/catalog/product_info.php?cPath=22&products_id=23) (обзор базовых возможностей инструментального средства для быстрого старта в его использовании), 1 день
6. PREQ0002, [Возможности IBM Rational Requirements Composer](http://www.institutio.ru/catalog/product_info.php?cPath=22&products_id=24) (базовые и расширенные интеграционные возможности RRC, интегрированное использование средства с учетом подхода Collaborative Application Lifecycle Management), 2 дня
7. PREQ0003, [Управление требованиями с использованием IBM Rational Requirements Composer](http://www.institutio.ru/catalog/product_info.php?cPath=22&products_id=22) (введение в дисциплину управления требованиями, рассмотрение базовых и расширенных интеграционных возможностей RRC), 3 дня
8. PTST0001, [Введение в IBM Rational Quality Manager](http://www.institutio.ru/catalog/product_info.php?manufacturers_id=16&products_id=31) (обзор базовых возможностей инструментального средства для быстрого старта в его использовании), 1 день
9. PTST0002, [Возможности IBM Rational Quality Manager](http://www.institutio.ru/catalog/product_info.php?manufacturers_id=16&products_id=30) (базовые и расширенные интеграционные возможности RQM, интегрированное использование средства с учетом подхода Collaborative Application Lifecycle Management), 2 дня
10. PTST0003, [Интегрированное управление тестированием с использованием IBM Rational Quality Manager](http://www.institutio.ru/catalog/product_info.php?manufacturers_id=16&products_id=28) (введение в дисциплину тестирования, рассмотрение базовых и расширенных интеграционных возможностей RQM), 3 дня

* [IBM](http://www.finecosoft.ru/category/ibm)

* [Jazz](http://www.finecosoft.ru/category/jazz)

* [Rational](http://www.finecosoft.ru/category/rational)