КОНФЕРЕНЦИИ СЕМИНАРЫ

.NET: ДОРОГА К ВСЕПРОНИКАЮЩЕЙ ИНТЕГРАЦИИ

Microsoft Research Academic Days in St. Petersburg
Зеленогорск, 21-23 апреля 2004 г.

Rafal Lukawiecki, Project Botticelli

От редакции:

В апреле 2004 года в Зеленогорске прошли «Академические Дни Microsoft Research в Санкт-Петербурге», которые собрали более сотни занимающихся информатикой ученых из разных вузов России.

В серии часовых докладов сотрудники MS Research изложили основные направления и результаты исследований, проводимые в области создания программного обеспечения.

В этом номере журнала публикуется аналитическая статья участника конференции проф. В.О. Сафонова, посвященная технологии .Net («дот нет»).

Внимание слушателей привлек очень живой доклад, который сделал на эту тему Rafal Lukawiecki. С разрешения Microsoft Research мы помещаем на диск полную видеозапись этого доклада. Доклад «.Net: Road to Pervasive Integration», который был сделан на английском языке, запомнился слушателям, благодаря популярному изложению, ясному языку, четкому произношению, обилию метафор и несомненному педагогическому мастерству докладчика.

Сокращенное начало этого доклада приведено ниже.

... Давайте поговорим о всеобщей интеграции. Повсеместной! Такая интеграция происходит автоматически, для этого не надо прикладывать усилий. Имеем ли мы такую возможность на сегодняшний день? Нет! Ситуация сегодня такова, что все хотят соединяться со всеми. Но каждое соединение стоит нам времени и денег, и в итоге приходится принимать решение в пользу конкретной компании или конкретного приложения... Много ли имеется таких соединений? Конечно, их количество растет экспоненциально, в результате чего бизнесмены, правительство и ученые говорят, что это самая большая и дорогая проблема, имеющаяся сейчас в области информационных технологий. Люди хотят объединяться. Но это стоит слишком дорого и требует слишком больших усилий. Так как же можно решить эту проблему? Давайте поговорим о будущем. Там, в самом отдаленном будущем, любые два компьютера будут соединены. Любые два приложения тоже интегрированы. И Вам не надо для этого ничего делать. Вы просто поставите рядом два компьютера, и они сами будут знать, для чего им надо объединиться. Хочу

привести пример. У меня есть очень хорошая метафора. Есть одна научно–популярная телевизионная передача: она американская, поэтому не уверен, что она есть в России. Она очень старая, ей порядка 35 лет. Называется «Star treck». В России она есть? И да и нет. Наверное, это зависит от того места, где вы живете. «Star treck» – обычная научно-популярная передача. В ней есть космический корабль, который летит в космос и находит новые цивилизации. Это называется «первый контакт». Если вы смотрели эту передачу, то могли заметить нечто невероятное. Всякий раз, когда корабль впервые сталкивается с новой цивилизацией (то есть они никогда-никогда не встречались до того, как встретились в первый раз), компьютеры космического корабля полностью интегрируются с компьютерами другой цивилизации. Как такое возможно? Microsoft утверждает, что используется .Net ☺. Я не уверен, что так оно и есть, но это говорит о том, что осуществление подобной мечты возможно.

Рассмотрим набор простых протоколов, которые позволяют сети взаимодействовать. SMTP (Simple Mail Transfer Protocol) – плохой протокол. Но у нас есть электронная почта, которая работает во всем мире. (И спам. ☺). Что я хочу создать – это как раз такой набор очень простых протоколов, не обязательно оптимальных по всем критериям, но которые смогут предоставить необходимую всеобщую интеграцию.

Вы могли подумать: «Да, .Net и есть решение!» Не могу сказать, что это так. В «древние» шестидесятые годы у компьютеров, на которых я программировал, была возможность взаимодействия. Она называлась «Милая девушка, которая берет магнитные накопители из одного компьютера и ставит их в другой». С помощью .Net мы избавили программистов от необходимости создавать специальные решения для интеграции. Внутри .Net все программное обеспечение замечательно интегрировано друг с другом. Можно ли это считать конечным решением? Нет! Для того, чтобы объяснить, почему нет и чтобы опровергнуть мнение многих из вас, я хочу представить новую концепцию, может быть, не новую для вас, но новую для нашей лекции, и я представлю ее неформально. Я хочу сопоставить интеграцию на различных уровнях абстракции. Давайте посмотрим на тот процесс, который может быть четко описан на естественном языке. Например, оформление счета, получение денег, выяснение, нет ли у меня аллергии на пенициллин, путем проверки данных в моей медицинской карте. Я говорю сейчас о том уровне, на котором не нужны программисты. Уже сейчас нам понятно, что .Net бесспорно полезен для интеграции на тех низких уровнях абстракции, когда у нас есть достаточный формализм. Однако можно заметить, что .Net не подходит для интеграции на высоком уровне, где есть неясность, много различных вариантов.

Что мы можем сделать на высоком уровне? Довольно много! Это как раз тот случай, где технология Web сервиса предоставляет не очень точный, но полезный ответ. Web сервис часто определяют как программируемый компонент приложения, получить доступ к которому можно, используя стандартные открытые Web-протоколы. Что я подразумеваю под стандартными открытыми Web-протоколами? Для начала я хочу показать цель: «нирвана» взаимодействий! Это когда Oracle и Sun, IBM, приложения .Net и решения LINUX работают вместе и уважают друг друга. Примерно так, как нам хотелось этого в 60-х годах. Такая «нирвана» в Web-сервисе возможна, если

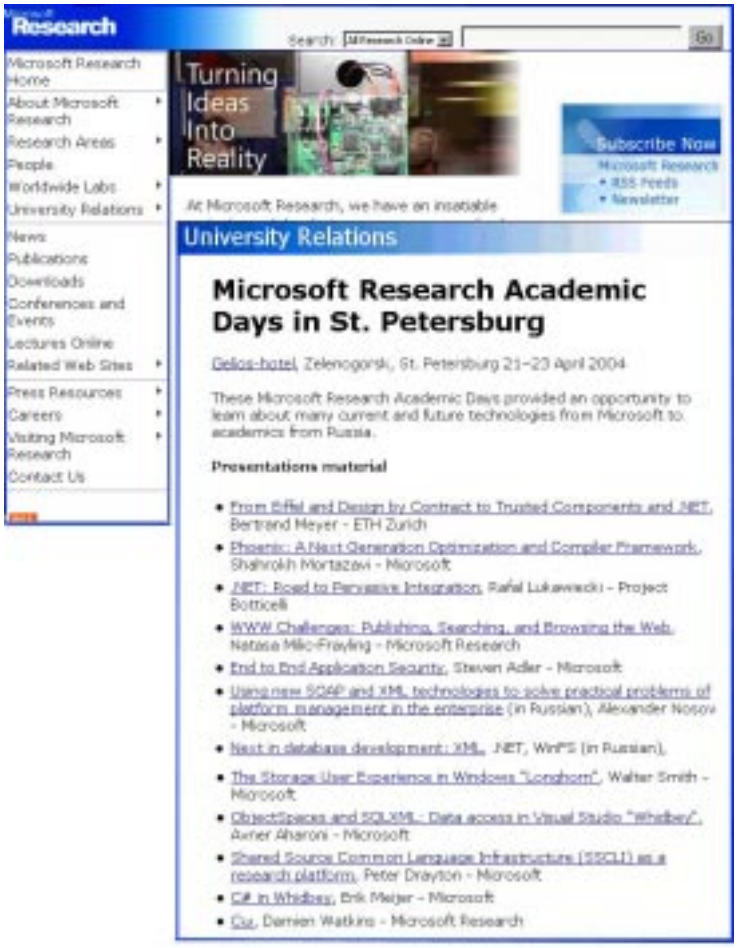
есть языковая независимость, платформенная независимость, независимость оборудования. И если существует возможность иметь независимость внутри архитектуры, то она необходима и снаружи – в интерфейсе. Это и есть наша цель. Как же ее достигнуть? Надо определить стандарты. Перед тем, как я расскажу вам о стандартах, я должен представить одну из наиболее важных концепций, как требование для выполнения этой «нирваны». Конечно, это позднее связывание. Мир деловых людей, правительственные разработчики ПО тоже впервые осознали важность позднего связывания, потому что они вдруг обнаружили, что проведение обычных транзакций на этой новой платформе невозможно, если использовать старый тип транзакций. Вдруг обнаружилось, что такая, казалось бы, простая операция, как резервирование места online для проведения отпуска, требует некоторого количества действий. Часть из них могут быть необратимыми. Например, я звоню в турагенство, желая поехать на какой–нибудь красивый, но очень маленький остров в Средиземном море. Я поручаю моему агенту купить мне самый дешевый билет на самолет и заказать комнату в самом дешевом отеле. «Хорошо, говорит агент, я зарезервирую Вам билет на самолет. Вы полетите самыми дешевыми авиалиниями в Европе. У них осталось всего одно место. Не беспокойтесь! Я куплю его для Вас!» Турагент покупает мне этот билет и начинает искать отель, но остров очень маленький, и свободных комнат нет ни в одном отеле. Агент приходит ко мне и говорит: «Рафал, к сожалению, в отелях нет свободный комнат, твоя поездка не удастся. Пожалуйста, заплати мне 100 евро за невостребованный билет, которым ты не воспользуешься, но который я уже зарезервировал». Так? Да, это и есть проблема сегодняшнего дня. Резервирование билета – необратимая операция. И что делать? Как можно представить семантику подобной транзакции? Конечно, пользуясь новой концепцией, которая называется компенсирующие транзакции. Этот не совсем транзакции, но мы называем их транзакциями, и в них заложена отличная идея того, что даже при отсутствии возможности отката операций баланс системы на более высоком уровне остается стабильным. В нашем примере мы компенсируем деньги клиента за счет авиакомпании или за счет специального фонда, который существует у нас и находится в некоем сейфе. Понятно, что подходы с поздним связыванием – это необходимое требование при разработке стандартов взаимодействия.

Без какого-либо ущерба мир движется в сторону XML. Я не буду об этом много говорить. Пойдем в сторону Web-сервисов сетевого взаимодействия, в сторону простых стандартов. Нам нужен язык, своего рода эсперанто. К примеру, SOAP – простой соединительный протокол для связи между поставщиком услуг и потребителем, примитивный, и он не решает пока больших проблем, связанных со смыслом данных. Поэтому мы настаиваем на протоколе под названием WSDL (Web service description language), для того чтобы описать API (application programming interface) синтаксиса Web-сервиса. WSDL делает много вещей, которые должен делать и много того, что вовсе не должен делать. Это плохой протокол, но, как и SMTP, он выполняет свою функцию, поэтому мы им пользуемся. Однако, WSDL не отвечает на важный воп-

рос: как найти Web-сервис, если возникнет такая необходимость. Поэтому на арену выходит UDDI, который сам по себе не менее плохой протокол, он не готов до конца, одним словом, он ужасен. Но он работает! UDDI не самый совершенный способ для этого, но он принят во всем мире. Что он делает? Он позволяет Вам создать массу проблем ☺ и Web-сервисов, которые решают эти проблемы. Он очень неточен и требует, возможно, еще большой доработки, но в настоящее время мы им пользуемся, потому что, безусловно, это лучший способ понять, что надо исправить.

Вам все это известно, поэтому приношу свои извинения за то, что говорю об известных вещах. Я обещаю, что начиная с этого момента, буду говорить о тех вещах, которые по статистике знает не более, чем 20% этой аудитории...

Полная видеозапись доклада и его текст на русском языке находятся на диске к журналу.



Стенограмму доклада подготовили и перевели на русский язык

Нозик Анна Александровна, студентка 4 курса математико-механического факультета СПбГУ,

Мамаева Светлана Олеговна, аспирант кафедры ИИСТ СПбГУ.