А. Н. Терехов, Л. А. Эрлих, А. А. Терехов

ИСТОРИЯ И АРХИТЕКТУРА ПРОЕКТА RESCUEWARE

История проекта RescueWare

Проект RescueWare берет свое начало в компании Seer Technologies, которая была основана в марте 1990 г. в Нью-Йорке как совместное предприятие фирм First Boston Corporation и IBM, а также группы учредителей-акционеров. При своем основании компания Seer Technologies опиралась на пакет передовых технологий для инструментальных средств разработки систем, основанных на модели клиент/сервер. Лидирующий продукт этой компании — Seer*HPS (High Productivity System) — был изначально создан группой сотрудников First Boston Corporation с целью способствовать быстрой и эффективной по стоимости переработке всего портфеля служебных офисных систем First Boston для развертывания его на революционной по тем временам трехуровневой распределенной архитектуре. Технология Seer*HPS была придумана и разработана под руководством Вивека Вадвы и Леонида Эрлиха, ставших затем техническими соучредителями Seer Technologies, где Вивек Вадва выступал в роли главного технолога, а Леонид Эрлих выполнял обязанности старшего вице-президента технологического подразделения. Стоит отметить, что на некоторые из изобретений, составивших ядро Seer*HPS, — такие как «Автоматизированная разработка программного обеспечения» и «Система электронного распространения программного обеспечения», — были получены патенты США.

Seer Technologies разрасталась необычайно быстро и в течение 1990–1995 гг. превратилась в международную организацию с общим доходом 118 млн. дол. и штатом в более чем 800 сотрудников. Seer вела дела в Северной и Южной Америке, Европе, а также азиатско-тихоокеанских регионах. С годами Seer не изменяла своему курсу на крупные вложения в научно-техническую деятельность и продолжала лидировать на рынке инструментальных средств разработки систем клиент/сервер; о ней высоко отзывались аналитики отрасли и пресса, включая таких создателей мнений в промышленности, как Gartner Group. Чтобы поддерживать свой быстрый рост, компания перенесла объединенную штаб-квартиру в один из офисов в Research Triangle Park в г. Кэри, шт. Северная Каролина. Находясь на пике своих корпоративных достижений, Seer Technologies успешно осуществила выпуск акций (IPO — Initial Public Offering) и стала компанией открытого типа. Последним событием для нее явилась покупка ее компанией Level 8 Inc. и слияние проводимых ею операций с деятельностью этой фирмы.

Seer Technologies ориентировалась на масштабного потребителя и крупные предприятия. Большинство клиентов Seer Technologies пользовались Seer*HPS для построения больших приложений, играющих важную роль в их бизнесе. Средняя стоимость пакета инструментальных средств и услуг по Seer*HPS составляла около 1,3 млн. дол. В процессе взаимодействия с клиентами сотрудники Seer Technologies вновь и вновь убеждались в том, что при построении новых современных архитектур необходимо прикладывать большие усилия для интеграции в создаваемую систему старых приложений. Каждый раз клиентам Seer было трудно отказаться от имеющихся у них систем, работающих на *mainframe*, и разрабатывать новые решения с нуля (так называемая «разработка с чистого листа»).

Идя навстречу нуждам потребителей, в сентябре 1992 г. Seer дала старт исследовательскому проекту по разработке пакета технологий, которые позволили бы заказчикам при создании новых, основанных на Seer*HPS архитектур использовать результаты, накопленные в старых приложениях. Начало проекту, получившему условное название RescueWare, было положено Вадвой и Эрлихом совместно с партнерами — группой российских ученых из лаборатории системного программирования математико-механического факультета Санкт-Петербургского государственного университета. В этой лаборатории разрабатывались трансляторы с языков Алгол 68, Паскаль, Модула 2, Ада, реализовывались языки, используемые в задачах искусственного интеллекта. Накопленный опыт и сложившийся коллектив, состоявший из высококвалифицированных математиков, системных программистов и инженеров-электронщиков, позволили в 1991 г. создать на базе лаборатории государственное предприятие ТЕРКОМ, основной задачей которого являлось внедрение в промышленность научных результатов лаборатории. В 1998 году было дополнительно создано ЗАО ЛАНИТ-ТЕРКОМ, входящее в состав холдинга ЛАНИТ. Бессменным лидером и генеральным директором обеих компаний был и остается Андрей Николаевич Терехов — доктор физико-математических наук,

профессор, заведующий кафедрой системного программирования Санкт-Петербургского государственного университета.

RescueWare for Seer*HPS разрабатывался в период с 1992 по 1996 г. В течение этого времени продукт прошел ряд испытаний в реальных условиях у потребителей. Первые несколько попыток применения продукта на практике оказались неудачными из-за непредвиденно высокой сложности конкретных старых систем, поэтому пришлось вносить изменения в саму методологию модернизации. Так, со временем стало совершенно очевидно, что подход по принципу «черного ящика», предполагающий полностью автоматический перенос устаревших конструкций на новые платформы, попросту не работает. Реинжиниринг старых систем требует выборочной идентификации и преобразования ценных частей старой системы, выполняемых при непосредственном участии бизнес-аналитика (так называемый принцип «белого ящика»). Потребовалось в общей сложности более четырех лет непрерывного экспериментирования с базовой технологией и методологией ее использования, чтобы сделать RescueWare коммерчески успешным.

К 1997 г. RescueWare стал серьезным пакетом многообещающих технологий модернизации старых комплексов. Одновременно с этим рынок средств разработки систем клиент/сервер иссяк, а с ним — и перспективы всей серии продуктов Seer*HPS. В административной группе Seer после ряда совещаний обозначились существенные разногласия по вопросу о стратегических направлениях деятельности компании и, в частности, об ассортименте предлагаемых ею продуктов. Вивек Вадва и Леонид Эрлих советовали перенести центр тяжести с инструментальных средств разработки систем клиент/сервер на модернизацию старых комплексов с ее более многообещающим рынком. Оказавшись не в состоянии изменить стратегический курс фирмы, они подтвердили свою веру в RescueWare тем, что покинули Seer и основали свою собственную компанию по модернизации старых комплексов, получившую имя Relativity Technologies.

Компания Relativity Technologies была основана в феврале 1997 г. в Research Triangle Park, шт. Северная Каролина. Вивек Вадва стал главным исполнительным директором компании, а Леонид Эрлих — главным техническим директором. По взаимному соглашению с Seer Technologies, Вадве и Эрлиху было полностью передано право собственности на технологию RescueWare, и основой деятельности новой компании стало продолжение партнерства с командой ЛАНИТ-ТЕРКОМа.

В течение последних трех лет Relativity Technologies Inc. продолжала быстро развиваться. В настоящее время у Relativity Technologies Inc. есть представительства в Research Triangle Park, в Нью-Йорке, Чикаго, Атланте, Далласе и Сан-Франциско, а также в Лондоне. Компания имеет штат в 50 сотрудников, работающих в разных странах, и значительно расширила свое партнерство с ЛАНИТ-ТЕРКОМ.

Relativity Technologies Inc. прочно заняла лидирующие позиции на рынке модернизации старых комплексов. Промышленные аналитики и пресса постоянно уделяют Relativity большое внимание. Компания установила ряд прямых партнерских связей с лидерами отрасли — Netscape, Rational Software, Platinum Technologies, Unisys, Ernst and Young, IMR Global и Tier Technologies. Ведущий продукт фирмы, RescueWare, пользуется большим успехом у потребителей.

Ключевым фактором успеха Relativity Technologies является расчетливая своевременность обращения к рынкам модернизации старых комплексов, вытекающая из увеличения потребности в преобразовании старых систем. Вот некоторые из основных катализаторов этой нужды в переменах:

- до сих пор не удовлетворенный спрос деловых кругов на решение проблемы 2000 г.;
- быстрое развитие систем, ориентированных на электронную коммерцию;
- популярность систем с процедурами восстановления после ошибок;
- усиление спроса на CBD (компонентно-ориентированные разработки).

Одной из первых технологических инициатив Relativity Technologies было расширение области действия RescueWare с выходом из пределов находящейся в частном владении платформы Seer*HPS на более широкий коммерческий рынок. Учитывая разнообразие потенциальных исходных и целевых платформ, было чрезвычайно важно не упускать из виду стратегическую цель, а не ограничиваться потребностями одного, пусть даже крупного заказчика. Проведя широкие исследования рынка, Relativity Technologies выбрала в качестве исходного языка Кобол, составляющий около 80% рынка старых комплексов. Было также принято решение об ориентации на два класса целевых платформ:

- платформу Microsoft, представленную системами Visual Basic, C++, HTML, ODBC, ActiveX/COM и ASP;
- семейство продуктов Java, представленное системами Java, Java Servlets, JDBC, Java Beans, Enterprise Java Beans и JSP.

Поддержка современных баз данных была расширена, с тем чтобы охватить наиболее популярные реляционные базы — Oracle, Sybase, Informix, SQL Server и DB/2. С течением времени поддержка Кобола также расширилась с IBM mainframe на множество других старых платформ — таких как AS/400, Unisys и Micro Focus. Дополнительным направлением работ в отношении старых комплексов стала поддержка PL/I и Natural ADABAS. Интегрированная среда разработки RescueWare была переведена с OS/2 на платформы Windows/NT и Windows 95/98.

Архитектура RescueWare

Первая попытка создания средства реинжиниринга

Существенную роль при разработке архитектуры RescueWare сыграл огромный опыт, накопленный сотрудниками ЛАНИТ-ТЕРКОМ в создании трансляторов, поэтому на первом этапе основные архитектурные решения принимались по аналогии с традиционной методикой написания трансляторов.

Самая первая коммерческая версия системы RescueWare создавалась как средство преобразования бизнес-приложений, написанных на Коболе, в язык Rules, являющийся основой Seer*HPS. К моменту начала работ казалось, что эта задача ничем не отличается от задачи создания обычного компилятора Кобола, только вместо объектного кода необходимо генерировать программу на языке Rules. Более того, авторы предполагали, что на вход создаваемой системе реинжиниринга будут подаваться преимущественно отлаженные программы, успешно работающие в течение нескольких лет, поэтому нет необходимости создавать истинно диалоговую систему (сообщения об ошибках, привязка к исходному тексту, встроенные текстовые редакторы и тем более отладчики). В итоге был сделан вывод, что создаваемая система может запускаться в пакетном режиме. Таким образом, пользователь мог воспринимать процесс реинжиниринга как «черный ящик», на вход которому подаются исходные программы, а на выходе генерируются эквивалентные им программы на целевых языках.

Однако во время работы над проектом «RescueWare for Seer*HPS» выяснилось, что задача реинжиниринга значительно сложнее обычной трансляции программ и требует более мощных методов. Это вызвано следующими особенностями реинжиниринга по сравнению с обычной трансляцией:

- к генерируемым программам предъявляется требование возможности сопровождения и развития (в отличие от обычной трансляции в ассемблер или машинный код);
- за долгие годы использования исходные тексты многих программ оказываются утерянными или не соответствующими выполняемым версиям, поэтому средство реинжиниринга должно быть не просто диалоговым средством трансляции, но и обладать возможностями инвентаризации приложений;
- не все конструкции исходных языков имеют точные аналоги в целевых языках (особенные трудности возникают в тех случаях, когда целевой язык беднее, чем исходный, как это было в случае с трансляцией программ на Коболе в Rules), поэтому на практике некоторые особо сложные конструкции, не имеющие аналогов в целевых языках, не могут быть оттранслированы автоматически и оставляются для ручного перевода;
- в процессе перевода может потребоваться проведение анализа и визуализации исходных текстов с целью восстановления утерянных знаний о системе (reverse engineering).

Поэтому попытки применения «RescueWare for Seer*HPS» на практике вызывали огромные трудности: перед каждым новым проектом по реинжинирингу приходилось дорабатывать сам продукт RescueWare, чтобы успешно «бороться» с теми или иными особенностями системы. Поэтому в результате нескольких случаев применения средства «RescueWare for Seer*HPS» на

практике разработчикам системы стало ясно, что невозможно проводить реинжиниринг устаревших систем без участия человека.

Новая архитектура

От «черного ящика» к функционально полной среде реинжиниринга

Накопленный опыт был учтен при проектировании новой версии RescueWare 3.0. Существующая архитектура была дополнена целым набором средств, помогающих пользователю понять структуру исходной системы и влиять на процесс реинжиниринга.

Для этого в RescueWare 3.0 появился целый класс визуальных средств, таких как HyperCode, позволяющий визуализировать программы на различных языках и осуществлять легкую навигацию по ним, Diagrammer, предназначенный для создания диаграмм отношений между различными программными компонентами, и т. п.

Одним из наиболее интересных визуальных средств в составе RescueWare 3.0 стала технология извлечения знаний. Начальным толчком для исследований в этом направлении послужило понимание того, что монолитные, одноплатформные по своей природе старые системы плохо сочетаются с современной компонентно-ориентированной, распределенной архитектурой. Поэтому перевод 10000 строк программы на Коболе в эквивалентные 10000 строк программы на Java не является жизнеспособным решением проблемы.

На этом этапе появилось понимание того, что прежде чем каким-либо образом преобразовывать старую систему, нужно предварительно разбить ее на компоненты. Задачей этого разбиения является превращение большого жестко связанного программного блока в набор отдельных и независимых компонент. Такие компоненты затем могут быть размещены в распределенной многоплатформенной среде и связаны воедино с помощью объектных брокеров, таких как COM, Java Beans или CORBA.

Один особенно интересный метод разбиения программ на компоненты основан на отделении их деловой, содержательной части от технической инфраструктуры. Этот метод, названный «Business Rules Extraction», позволял «отсеять» весь зависимый от платформы код — например, управление транзакциями, управление памятью и обмен информацией между процессами — от лежащих в его основе содержательных команд. Извлеченные таким образом «бизнес-правила» составляют основу новой системы и автоматически переносятся на новую платформу без каких-либо изменений. В то же время вся техническая инфраструктура отбрасывается и затем вновь переписывается на заданной целевой архитектуре.

Описанный процесс все время требовал активного участия в нем инженера, владеющего соответствующей предметной областью. В самом деле, преобразованию сопутствует процесс постоянного принятия решений относительно границ отдельных бизнес-правил, возможности их использования в новой системе, а также схем разбиения и интеграции приложения.

Наличие такой интерактивной среды с мощной графикой и множеством точек принятия решений существенно расширило область применения RescueWare. Теперь пользователь мог не только конвертировать приложения, но и анализировать исходную систему, восстанавливать утерянные знания об архитектуре устаревшей системы, т. е. проводить полноценное возвратное проектирование системы.

Расширяемость набора входных и выходных языков

При проектировании RescueWare 3.0 было замечено, что при написании устаревших систем использовался целый ряд разнообразных по структуре языков программирования: алгоритмические (Кобол, PL/I, Ассемблер), описания экранных форм (BMS, AS/400 screens), взаимодействия с базами данных (SQL, IMS), управления транзакциями (CICS) и т. д.

Поэтому было необходимо предусмотреть возможность расширения набора входных и выходных языков программирования. Для этого было решено транслировать все входные языки в единый универсальный промежуточный язык, из которого впоследствии производится генерация в целевые языки (впервые эта идея сформулирована в описании промежуточного языка UNCOL более тридцати лет назад). При этом подходе, если есть m входных языков и n целевых языков (или платформ), то удается уменьшить требуемое число трансляторов с mn до m+n.

Таким образом, для добавления в систему какого-либо входного языка достаточно написать синтаксический анализатор этого языка, порождающий промежуточный язык, а также

некоторые функции поддержки времени выполнения для особенностей исходного языка, не имеющих точного аналога в целевом языке. Для добавления нового выходного языка также достаточно добавить только один просмотр генерации.

На данный момент, с помощью описанного приема система RescueWare работает с языками Кобол, PL/I и Natural ADABAS в качестве входных алгоритмических языков и C++, Java и Visual Basic в качестве выходных. Кроме того, поддерживается множество вспомогательных языков и стандартов (назовем CICS, встроенный SQL, BMS, IMS, MFS и экранные формы AS/400 как наиболее важные стандарты и языки исходных платформ и HTML, Java Servlets, Java Beans, Enterprise Java Beans, ASP, JSP, ODBC, ADO и JDBC как стандарты и языки целевых платформ).

Масштабируемость

В связи с появлением большого количества новых языков программирования (а следовательно, и сущностей, с которыми приходится иметь дело в процессе реинжиниринга), возникла необходимость введения репозитория используемых объектов. Вначале были предприняты попытки воспользоваться существующими стандартными репозиториями (например, Microsoft Repository), но оказалось, что они не могут обеспечить приемлемого быстродействия на реально встречающихся объемах данных.

Поэтому RescueWare 3.0 использует в своей работе собственный специально разработанный репозиторий, основанный на расширяемой объектно-ориентированной метамодели. Классам репозитория соответствуют либо реальные объекты, либо объекты, возникающие в процессе реинжиниринга. Например, программе на Коболе соответствует класс, полями которого, в частности, являются:

- название программы;
- ссылка на исходный текст программы;
- количество строк программы;
- логический признак того, является ли данная программа синтаксически корректной.

Особое внимание было уделено проблеме масштабируемости репозитория. В частности, пришлось отказаться от традиционной нормализованной модели хранения данных, так как в этом случае даже небольшие изменения приводят к необходимости отслеживания всех связей данного объекта и, при необходимости, внесения соответствующих изменений в зависимые объекты.

Поэтому репозиторий RescueWare имеет двухуровневую структуру. Первый уровень репозитория — это полностью нормализованная таблица объектов. В качестве объектов выступают достаточно крупные самостоятельные сущности, такие как программа на Коболе, логическая модель программы HyperView и т. п. На втором же уровне каждому объекту соответствует конкретная физическая структура и более подробные данные, которые хранятся в формате BLOB (Binary Large OBjects — большие бинарные объекты)[1]. Структуры второго уровня можно целиком передвигать или удалять. Преимущество данного подхода заключается в том, что изменения объектов в большинстве случаев локализованы на каком-то одном уровне: либо меняются внешние атрибуты объекта (что вызывает изменения только на первом уровне, но не внутри данного объекта), либо изменяются внутренние атрибуты объекта (это затрагивает только второй уровень, но не другие объекты).

Таким образом, за счет использования двухуровневой структуры достигается существенный выигрыш в производительности и масштабируемости репозитория. Единственным ограничением на рост репозитория является объем свободного места на диске, так как на скорости доступа рост репозитория не сказывается. Существуют также соответствующие прикладные программные интерфейсы, входящие в RescueWare SDK (см. ниже), обеспечивающие доступ к обоим уровням метамодели и манипулирование ими.

Другим интересным архитектурным решением репозитория RescueWare являются средства логического объединения объектов в специальные контейнерные конструкции метамодели репозитория — проекты, схемы БД, словари, пакеты программ и т. д. Эти контейнерные конструкции позволяют группировать объекты репозитория в поддающиеся контролю наборы.

[1] В названии BLOB есть небольшая игра слов — помимо приведенной выше расшифровки, слово BLOB на английском языке также означает «капля» или «маленький шарик», что достаточно точно отражает суть дела: с точки зрения объектов первого уровня, BLOB действительно является «цельной каплей».

В настоящее время ведется работа по унификации подобных контейнеров с помощью создания универсального контейнера Проект.

Еще одной ключевой характеристикой репозитория RescueWare является поддержка им многопользовательского режима. На данный момент в коммерческой версии продукта такая поддержка отсутствует. Тем не менее потребность в ней хорошо осознана, и в будущих версиях такая поддержка будет внедрена.

Ослабленные синтаксические анализаторы

Одной из сложных проблем реинжиниринга является размытость описания устаревших языков программирования. Например, существуют десятки различных диалектов Кобола (включая даже диалекты, принятые на том или ином предприятии!), каждый из которых имеет свою специфику, дополнительные ключевые слова и т. п. Кроме того, многие стандарты Кобола имеют взаимно противоречащие особенности, поэтому не представляется возможным создать «универсальный» синтаксический анализатор Кобола. В качестве частичного решения этой проблемы в RescueWare реализовано несколько независимых анализаторов, рассчитанных на наиболее распространенные версии Кобола (VS COBOL II, MicroFocus COBOL и Fujitsu COBOL).

Однако в том случае, когда на вход системе подается неизвестный диалект Кобола, инженеру, производящему реинжиниринг системы, приходится вручную удалять или комментировать неподдержанные конструкции, чтобы пройти стадию синтаксического анализа. Чтобы избежать этой утомительной процедуры, был разработан так называемый «ослабленный» синтаксический анализатор (relaxed parser), который игнорирует все отличия и вариации в исходных текстах. Эти конструкции интерпретируются как подлежащие пропуску, но, тем не менее, передаются в виде комментария дальше. Естественно, в случае пропуска незнакомых конструкций не всегда удается корректно восстановить имевшуюся в виду структуру программы, поэтому «ослабленный» синтаксический анализатор используется в основном на этапе инвентаризации и начального анализа системы.

На основе сходной концепции RescueWare обучается и тому, как поступать с другими отсутствующими элементами системы — например, заголовочными файлами.

RescueWare SDK

Итак, RescueWare 3.0 показал себя мощным средством реинжиниринга с единой проработанной технической архитектурой. Поэтому многие компании оказались заинтересованными в создании своих расширений RescueWare (поддержка новых языков программирования, разработка новых графических средств и т. д.).

Для этого был создан комплект разработчика RescueWare (RescueWare SDK, Software Development Kit), представляюший собой собрание документированных прикладных программных интерфейсов. С помощью SDK стало возможным дать доступ к уже существующей архитектурной структуре RescueWare сторонним фирмам-разработчикам.

SDK позволяет использовать функции практически всех уровней RescueWare, включая рабочую оболочку, репозиторий и средства анализа. Программные интерфейсы поставляются без исходных текстов, что защищает Relativity Technologies от раскрытия принадлежащих ей алгоритмов, но тем не менее позволяет ее ключевым потребителям обрести некоторую независимость в отношении расширения технологии RescueWare. Кроме того, RescueWare SDK дисциплинирует разработчиков продукта, вынуждая их поддерживать ясную и прочную архитектуру, а также обеспечивать совместимость версий снизу вверх.

Направления развития RescueWare

RescueWare — это живой и развивающийся продукт, непрерывно учитывающий необходимость считаться с потребностями быстро меняющегося рынка. Залогом его успеха является крепкая техническая архитектура и постоянное развитие продукта, опережающее конкуренцию. Перемены — постоянная движущая сила эволюции RescueWare.

Прежде всего, это новые платформы и языки программирования, которые будут добавляться в RescueWare во все время жизни этого продукта. Это касается как исходных платформ (см. последние работы с PL/I и Natural ADABAS), так и целевых современных архитектур. Работа с платформами может принять новое направление вследствие формирующихся стратегических

соглашений с компаниями, предоставляющими платформенные решения, такие как COOL:Gen компании Sterling Software или SynerJ компании Forte.

Другим двигателем перемен служит непрекращающийся поток новых стандартов отрасли. К самым последним примерам таких стандартов относятся XML (расширяемый язык разметки), Enterprise Java Beans и COM+. Хотя с момента выпуска таких стандартов и до принятия их потребителем проходят месяцы, а иногда даже годы, лидирующему коммерческому продукту все равно очень важно опережать своих конкурентов в поддержке новых стандартов.

Необходимо также учитывать общие тенденции всей отрасли, которые, скорее всего, вызовут серьезные сдвиги в отношении технологических платформ. Так, последнее слово на рынке распределенных вычислений и электронной коммерции — это EAI (Enterprise Application Integration, интеграция приложений предприятия). EAI предлагает многослойную поддержку продуктов на уровне предприятия в целом и интеграцию «business-to-business». Эти многочисленные слои начинаются с самых фундаментальных промежуточных программных средств — таких как удаленные вызовы процедур, отправка сообщений и объектные брокеры — и продолжаются вплоть до новых сложных концепций брокеров интеграции и управления потоком деятельности.

В настоящее время Relativity Technologies активно работает над предложением ряда стратегических партнерств ведущим поставщикам EAI. Следствием этих партнерств, без сомнения, будут серьезные усилия по перекрестной интеграции продуктов.

Кроме того, следует учитывать тот факт, что устаревшие системы чаще всего были написаны не на одном языке программирования. Например, программы на Коболе чаще всего использовали монитор транзакций CICS и язык описания экранных интерфейсов BMS. При переводе на новые языки и платформы имеет смысл рассматривать программы на всех этих языках как единое целое, чтобы обеспечить целостность генерируемых результатов. Эта идея поддержана в RescueWare лишь частично, но ведутся работы по созданию так называемой «сквозной» генерации, в рамках которой программы на различных языках программирования могли бы рассматриваться одновременно.

Наконец, планируется дальнейшее развитие RescueWare как средства анализа и сопровождения устаревших программ. Изначально RescueWare было ориентировано в основном на трансформацию устаревших приложений. Однако такие средства, как HyperCode или Business Rules Extraction, существенно расширили область применения RescueWare. В будущем планируется дальнейшее развитие средств анализа и восстановления знаний, чтобы предоставить пользователю самые широкие возможности по сопровождению и реинжинирингу приложений.