Удаленное сопровождение корпоративных информационных систем

Проф.А.Н.Терехов, А.А.Терехов
Холдинг "ЛАНИТ", Санкт-Петербургский Государственный Университет
Опубликовано в журнале "Data Communications Russian Edition", №5-6, 1998

Постановка задачи

Сопровождение всегда признавалось одним из основных этапов жизненного цикла программного обеспечения (ПО). Уже к середине 70-х годов было признано, что сопровождение – это этап, занимающий более 50% затрат на разработку и внедрение программного комплекса [1].

В современных корпоративных информационных системах этот вопрос приобретает первоочередное значение. Обычно корпорации имеют распределенную структуру с большим количеством территориально удаленных отделений или филиалов (например, банки и их филиалы, дистрибьютерские компании с представительствами в различных регионах и городах, государственные учреждения с региональным подчинением и т.д.).

Смена версий ПО ведет для таких организаций к ощутимым сложностям в связи с необходимостью их массового тиражирования и внесения изменений во все используемые в организации копии данной программы, или базы данных, или экранные формы и т.п. Поддержка целостности используемых версий становится дорогостоящей операцией, причем ее стоимость резко возрастает с ростом размера корпорации и увеличением количества филиалов.

В дальнейшем мы будем называть главное отделение корпорации просто Центром, а подчиненные ему отделения, расположенные в других городах и регионах – филиалами. Обычно новые версии ПО создаются в Центре и лишь затем внесенные изменения доводятся до филиалов. Мы будем рассматривать только такую схему.

Основная трудность системы поддержки версий состоит в том, что на стороне филиала требуется наличие достаточно квалифицированного оператора, который мог бы произвести установку ПО от начала до конца и при этом мог бы справиться с возможными ошибками. Более того, от оператора может потребоваться также понимание особенностей конфигурации имеющегося в филиале программно-аппаратного комплекса. В нашей практике встречались, например, ситуации, когда ПО переставало функционировать только из-за различий в типах используемых в филиалах модемов. Оператор должен уметь самостоятельно диагностировать проблемы такого рода.

К сожалению, в реальной жизни дело обстоит по-другому. Далеко не везде имеются квалифицированные программисты, разбирающиеся в работе сопровождаемой программы. Зачастую в филиалах находятся только конечные пользователи программы. Обычно такая ситуация вызвана объективными причинами – например, тем, что содержать в каждом филиале программиста слишком накладно.

В связи с этим, для успешного функционирования ПО Центру приходится содержать большой штат технической поддержки. Хорошо при этом, если проблему, возникшую в филиале, удается диагностировать и решить по телефону или e-mail. Но что делать, если этих средств недостаточно?

Многие фирмы вынуждены в таких случаях посылать на место “аварии” квалифицированного программиста из Центра. Это, конечно же, приводит к несоразмерно большим затратам на сопровождение.

Более удачным подходом является использование средств удаленного доступа (таких как NetWare Connect 2.0 компании Novell или Remote Access Service для Windows NT Server корпорации Microsoft), позволяющих работать на территориально удаленном компьютере как на локальном. Наиболее частый случай использования удаленного доступа – это взаимодействие с Центром командированных сотрудников. Тем не менее, это не всегда позволяет решить проблемы сопровождения. Во-первых, использование подобных средств в большой корпорации зачастую влечет большие расходы. Во-вторых, резко возрастает риск несанкционированного доступа [2]. Наконец, даже при таком подходе для исправления проблемы в филиале требуется вмешательство специалиста из Центра и поэтому с ростом числа филиалов подобное решение становится все менее и менее привлекательным.

В данной статье мы попытаемся показать, как можно существенно сократить расходы на сопровождение с помощью использования "технологии удаленного сопровождения". Основное требование к такой системе состоит в том, что модификация системы в филиалах должна осуществляться Центром автоматически.

Разумеется, первоначальная установка ПО в филиалах все равно должна производиться квалифицированным системным программистом, но такая установка происходит всего один раз. Во всех последующих случаях модификация должна требовать минимального вмешательства человека.

Мы будем рассматривать вопрос *создания* новой системы, одним из основных требований к которой является сокращение расходов на ее сопровождение и модификацию. Задача уменьшения стоимости сопровождения *уже существующей* системы является значительно более сложной и должна решаться с учетом конкретных особенностей данной системы, так как, конечно, не имеет общего решения.

Различные подходы к задаче тиражирования

Прежде чем перейти к основным вопросам статьи, отметим, что существует несколько вариантов решения проблемы тиражирования. Наиболее очевидный вариант заключается в создании инсталяционного пакета для каждой новой версии ПО с дальнейшей рассылкой по филиалам. Это решение, конечно, обладает достоинством универсальности, так как в дистрибутив можно включить любые изменения. Существует множество пакетов для создания дистрибутивов (наиболее распространенным является InstallShield). Однако при таком подходе ради изменения одного бизнес-правила или одного атрибута поля базы данных потребуется замена *всего* существующего комплекса (или, как минимум, компоненты, в которой произошли изменения). Чем больше используемая система, тем выше накладные расходы для данного подхода.

Другой вариант заключается в создании специального инструментального средства, отслеживающего все изменения и затрагивающего минимальное число компонент системы при их внесении. При таком подходе можно пересылать только необходимые изменения (среди разработчиков распространен термин “дельта”) между предыдущей версией системы и нынешней. Такие инструментальные средства позволяют многократно снизить накладные расходы и сетевой трафик, но, к сожалению, редко имеются в наличии, поэтому необходимо подумать об их создании еще до начала проектирования системы в целом. Для реализации этого варианта требуется ориентироваться на средства разработки, с помощью которых было создано приложение. Например, в разработанном в ЛАНИТе средстве поддержки программных комплексов, созданных с использованием PowerBuilder и Oracle, поддерживается именно описанный "дельтовидный" режим.

Наконец, при реализации информационных систем с помощью технологии Интранет [3] можно снизить затраты на сопровождение за счет использования языков сценариев (scripts), так как изменения сценария "прозрачны" для пользователя. К сожалению, область применения данного метода очень ограничена: можно производить таким образом начальную обработку данных, полученных из форм, но никому не придет в голову писать бизнес-логику на скриптах.

Пример реализации системы, использующей удаленное сопровождение

Покажем процесс проектирования системы, использующей технологию удаленного сопровождения, на примере создания системы бухгалтерского учета “Касса”, выполненном ЗАО“ЛАНИТ-ТЕРКОМ” для районного узла связи города Кировска (Ленинградская область).

Задача изначально носила распределенный характер. Каждый из узлов связи, подчиненных районному, оказывает разнообразные услуги населению, такие как прием и посылка телеграмм, получение абонентской платы за

телефон, оплата междугородных переговоров и т.д. Полученные за услуги деньги должны быть учтены и переданы в центральный узел связи вместе с информацией об услугах, за которые они были получены. Тем самым, помимо основного требования легкости в сопровождении, к разрабатываемому комплексу предъявлялись следующие требования:

- учёт всех оказанных услуг в локальной базе данных;
- печать чека для выдачи покупателю (для этого использовался кассовый аппарат “Электроника” с возможностью связи с компьютером);
- передача всех данных из местных узлов связи в Центр;
- синхронизация и репликация всех имеющихся данных.

Кроме этих основных задач существовали также дополнительные условия на работу программы:

- требовалась устойчивая работа с приемлемым временем реакции на запросы в следующей конфигурации: в центральном узле связи – Pentium с Windows NT, в прочих – 486 с Windows 95 и модемной связью по выделенной линии с ориентировочной скоростью модемов 21,600 бод.
- требовалась возможность продолжения работы в случае прерывания модемной связи.

К моменту начала проектирования уже существовало программное обеспечение, решавшее эти задачи, но локально для каждого отделения связи. Результаты работы за день пересылались в Центр на дискетах, где их сводили в единую базу данных.

Нами был разработан программный комплекс, работающий по технологии Интранет, для которого выполнялось основное требование заказчика – легкость и дешевизна сопровождения. В процессе создания данной информационной системы было решено множество интересных задач (от налаживания надежной модемной связи до управления печатью кассового аппарата), но сейчас хотелось бы ограничиться рассмотрением только тех аспектов созданной системы, которые связаны с сопровождением.

Для решения поставленных задач нами были выбраны следующие ОС и средства разработки: Windows NT на сервере, Windows 95 на клиентах, HTML документы со внедренными ActiveX-компонентами с использованием Microsoft Internet Explorer в качестве броузера. В качестве СУБД использовался SQL Server. Программное обеспечение писалось на Visual Basic и Visual C++.

Опишем обычную схему работы системы.

В начале каждого рабочего дня на рабочих местах в филиалах запускается Windows 95, в которой сразу же запускается Internet Explorer (версии 3.0 или выше), который по умолчанию загружает HTML-страницу Центра. На этой странице находится всего одна кнопка, нажатие на которую

вызывает загрузку ActiveX-компоненты, являющейся, собственно исполняемой программой. Реально это происходит следующим образом: вначале сравниваются версии ActiveX-компонент в Центре и в филиале. Если версии не совпадают, то компонента копируется на машину филиала и потом запускается; если же версии совпадают, то запускается локальная копия филиала. Операции сравнения и подкачки версий целиком поддерживаются штатными средствами Internet Explorer'a.

Для уменьшения сетевого трафика желательно разбиение крупных систем на большое количество ActiveX-компонент. Однако с увеличением числа различных подсистем растут и накладные расходы на их оформление в виде ActiveX. Поэтому представляется целесообразным остановить дробление на уровне отдельных библиотек.

Параллельно с основным, рабочим, процессом, на клиенте живет процесс, отвечающий за поддержание связи. Он проверяет наличие ping'а Центра и, в случае его отсутствия, разрывает все текущие модемные соединения и производит автодозвон.

При занесении новых данных филиал постоянно работает с локальной базой данных, но при этом пытается внести изменения и в глобальную базу данных – в Центре. Если связь по каким-либо причинам прервалась, то в специальной таблице Log (Журнал) делается запись о том, что данные удалось занести только в локальную БД. При восстановлении связи все записи, упомянутые в журнале, переписываются в глобальную БД.

В созданной системе для большинства запросов достаточно было обратиться к локальной базе данных, так как большинство филиалов ведут вполне самостоятельную работу. Однако иногда приходится читать данные из глобальной БД (с копированием в локальную). Система использует ночное время для синхронизации всех локальных баз данных с центральной.

Таким образом, локальные базы данных служат только буферами между локальными приложениями и центральной базой данных. В случае прерывания модемной связи между филиалом и Центром приложение продолжает работу, но уже только с локальной базой данных. Однако, в конечном итоге, все данные попадают в центральную базу данных. По сути, реализован простейший случай *асинхронного тиражирования данных* [4].

Трудности реализации, связанные с внешними ограничениями на систему

В связи с ограничениями на быстродействие целевых машин и стоимость создаваемого ПО использование разнообразных средств, позволяющих организовывать удаленный доступ к базам данных и мониторинг транзакций, не представлялось возможным. Поэтому такие средства пришлось написать самим, что, впрочем, не было чересчур сложным из-за небольших требований к их

возможностям. В результате был использован следующий механизм для доступа к удаленной базе данных.

В начале работы программное обеспечение филиала создает на сервере Центра объект, имеющий методы типа “Добавить в базу данных”, и в дальнейшем вызывает методы данного объекта с помощью Remote Automation Manager (подробное описание этой технологии приведено в [5]). Тем самым удалось существенно снизить сетевой трафик по сравнению с возможными альтернативами, например, использованием таких надстроек над ODBC как Data Access Objects (DAO) или Remote Data Objects (RDO). Кроме того, удалось добиться существенной разгрузки клиента, так как большинство запросов выполняется непосредственно на сервере и по сети передаются лишь результаты запросов (в отличие, скажем, отRDO, в котором попросту нет выполняющегося на сервере процесса). Используемая схема работы позволит и в будущем легко управлять распределением нагрузки между клиентом и сервером.

Почему ActiveX?

Остановимся на соображениях, по которым для реализации данной системы была выбрана технология ActiveX.

ActiveX, как и ее обобщение DCOM (Distributed Component Object Model), является логическим продолжением идей более ранней технологии OLE (Object Linking and Embedding), позволявшей динамически переиспользовать независимые компоненты в разных приложениях на одном компьютере. ActiveX перенесла эти возможности в World Wide Web [6]. К ее достоинствам можно отнести:

- возможность создания клиент-серверных приложений на основе WWW;
- мощность предоставляемых средств работы в сети (компоненты ActiveX выполняются на компьютере клиента и обладают всей полнотой доступа к его ресурсам);
- легкость повторного использования написанных компонент;
- поддержка стандарта OLE и, как следствие, большое количество уже разработанных компонент, с помощью которых легко создавать новые приложения.

Среди основных недостатков ActiveX – отсутствие многоплатформенности и недостаточно надежная модель безопасности. К сожалению, ActiveX до сих пор удобно использовать только в сетях на основе Windows. В нашем случае этот недостаток был несущественен, так как использование Unix-серверов не планировалось. Однако в более крупных сетях это становится существенной трудностью.

Модель безопасности ActiveX тоже является поводом для беспокойства. Безопасность загружаемых ActiveX-компонент частично может быть проверена

с помощью цифровых сертификатов, но конечное решение должен принимать пользователь. Таким образом, загружая интересующую его компоненту из сети, пользователь подвергает свою систему риску несанкционированного доступа. К счастью, в сетях Интранет этот вопрос успешно решается, так как они закрыты для доступа извне.

Наконец, заметим, что идея использования ActiveX как платформы для обновления приложений появилась одновременно у программистов во всем мире. Об этом свидетельствуют и цифры, приведенные в статье [7], согласно которым наибольшее количество случаев использования технологии ActiveX (24%) приходится именно на обновление приложений.

Обобщенный подход к удаленному сопровождению

Выше мы показали использование технологии удаленного сопровождения на примере проектирования простой *гомогенной* двухуровневой информационной системы с централизованной базой данных.

Те же самые идеи можно применять и для создания многоуровневой *гетерогенной* информационной системы. В этом случае потребуется использование более мощных средств для реализации удаленного сопровождения. Мы считаем наиболее удачным вариантом использование архитектуры CORBA [8].

Стандарт CORBA был разработан и опубликован группой OMG (Object Management Group) еще в 1990 году. Основными достоинствами стандарта CORBA являются:

- независимость от языка программирования и платформы;
- поддержка объектно-ориентированных и распределенных приложений;
- возможность переиспользования созданных компонент.

С другой стороны, у этой технологии существуют и недостатки. Прежде всего, хотелось бы отметить некоторую тяжеловесность CORBA – для разработки систем с такой архитектурой приходится затрачивать очень большие усилия. Кроме того, существуют некоторые трудности с перераспределением нагрузки между клиентом и сервером, то есть достаточно тяжело регулировать "толщину" клиента.

Наконец, хотелось бы обратить внимание читателя на еще одно существенное различие между CORBA и ActiveX/DCOM. Все технологии фирмы Microsoft бесплатны для пользователей Windows 95/NT. В то же время, для разработки системы с использованием стандарта CORBA необходимо тратить дополнительные средства.

Заключение

В данной статье мы попытались очертить круг проблем, связанных с сопровождением корпоративных информационных систем. Нам кажется, что при проектировании любой крупной системы одним из основных требований должна быть минимизация стоимости сопровождения. В связи с этим, имеет смысл заранее предусмотреть механизм распространения новых версий системы.

Большое количество проблем, связанных с темой статьи, осталось "за кадром", например:

- сопровождение системы с гетерогенной архитектурой;
- изменения версий ПО, вызванные изменениями баз данных;
- минимизация данных, требуемых для обновления версии и так далее.

Надеемся, что со временем мы сможем более подробно осветить и эти вопросы.

Литература

[1] **Brooks, Jr., F.P.** *"The Mythical Man-Month: Essays on Software Engineering"* // Reading, MA: Addison-Wesley, 1975
[2] **Брет Глас** *"Выработка стратегии удаленного доступа"* // LAN Magazine/RE, №4, 1997
[3] **П.Храмцов** *"Intranet – мифы и реальность"* // Открытые системы, №4, 1997
[4] **Б.О.Калиниченко** *"Асинхронное тиражирование данных в гетерогенных средах"* // СУБД, №3, 1996
[5] **Paul Bonner** *"Remote OLE Automation"* // PC Magazine, February 6, 1996 (русская версия данной статьи: http://pcmag.newman.ru/PAPERS/P961.htm
[6] **Дэвид Чеппел** *"Технологии ActiveX и OLE"* // Microsoft Press, Русская редакция, 1997
[7] **Тод Уоллак** *"Знакомство с ActiveX"* // ComputerWorld Россия, №2, 1998
[8] **Object Management Group** *"The Common Object Request Broker: Architecture and Specification"* // Revision 2.0, July 1995