СОЗДАНИЕ МНОГОМОДУЛЬНЫХ ИЗМЕРИТЕЛЬНЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ НА ОСНОВЕ ТЕХНОЛОГИИ COM

О. Ю. Смольянов

Предположим, разработана линейка цифровых измерительных приборов, подключаемых к персональному компьютеру. Требуется создать виртуальный измерительный программный комплекс, который бы обеспечивал взаимодействие пользователя с устройством. Кратко сформулируем требования к современному измерительному программному комплексу:

1. Комплекс должен состоять из драйвера и клиентской программы, использующей этот драйвер. Выделение драйвера в отдельный модуль необходимо для того, чтобы не переписывать объемную часть кода по взаимодействию с аппаратной частью. Это обусловлено тем, что клиентских программ может быть и несколько. Драйвер предоставляет базовые функции для непосредственной работы с измерительным устройством (старт/стоп устройства, чтение измерительных данных). Клиентская программа визуализирует процесс работы с прибором (математическая обработка измерительных данных, отображение, и.т.д.).
2. Клиентская программа должна обнаруживать все приборы данного типа, установленные на ПК. Требуется предоставить возможность пользователю одновременно работать со всеми возможными измерительными каналами всех подключенных приборов серии.
3. Часть приборов из производимой линейки уже разработана и выпущена с драйверами устройств. Другая часть приборов разработана, но функциональность драйвера претерпела изменения. На будущее, возможно, запланирована разработка новых приборов данной серии, с еще более широкими возможностями драйвера. Клиентская программа не должна зависеть от каждого драйвера серии, в противном случае для каждого нового драйвера придется переписывать клиентскую программу. Таким образом, расширение серии устройств не должно повлечь за собой изменения в клиентской программе.

Самое простое возможное решение - написание двухуровневой программной системы, состоящей из драйверов (уровень 1) и клиентских программ (уровень 2). При этомвозникает два уровня взаимодействий: исполняемого EXE-модуля (программа пользователя) и динамической библиотеки (драйвер), а также драйвера и измерительного прибора.

Пример двухуровневой структуры приведен на рис. 1. Такой структурой достигается относительная гибкость - нет необходимости в пе-

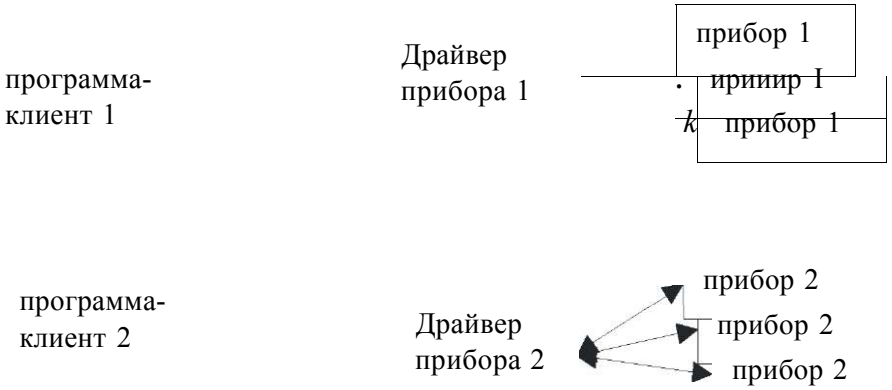

Рис 1. Двухуровневая структура многомодульного измерительного программного комплекса

переписывании заново низкоуровневой части кода. Возникает два комплекта поставки: полный, включающий в себя все компоненты (прибор, драйвер и программа), и базовый (прибор и его драйвер) - на случай, если пользователь захочет самостоятельно написать свою собственную клиентскую программу.

Однако, не смотря на кажущуюся гибкость, такая система не удовлетворяет всем вышеперечисленным требованиям к современной системе, поскольку в этом случае приходится изменять клиентскую программу при каждом видоизменении драйвера устройства. Такие системы неприменимы для большого числа вариаций драйверов внутри одной серии приборов. Приходится переписывать код программы, меняя интерфейс взаимодействия с драйвером.

Для выхода из этой ситуации необходимо повысить гибкость системы, увеличив уровень абстракции. Необходимо, чтобы программа взаимодействовала не с драйвером конкретного прибора, а с каким-то объектом, который был бы универсальным для всей серии приборов. Для того, чтобы программа не зависела от интерфейса драйвера, необходимо скрыть спецификацию интерфейса от программы. Это должен быть объект, который является посредником между программой и драйвером, и который переадресует вызовы заранее оговоренного интерфейса программа-объект в интерфейс драйвера. Таким образом, для каждого драйвера нужно написать свой объект, который, в свою очередь, предоставлял бы привычный интерфейс для клиентской программы. В этом случае, при создании новой версии прибора не требуется изменять и переписывать программу, нужно лишь создать новый объект для измененного драйвера. В этом и состоит суть нового, трехуровнего подхода к решению задач на построение измерительной системы.

В качестве промежуточного уровня берем некий объект, к которому предъявляются следующие требования:

• полное использование возможностей драйвера (внутренний интерфейс драйвер-объект), то есть умение объекта использовать драйвер;

• представление универсального внешнего интерфейса (для клиентских программ);

• внедрение в качестве plug-in в клиентские программы;

• возможность расширения внешнего интерфейса при сохранении поддержки старых версий;

• независимый от языка внешний интерфейс.

Оптимальным вариантом реализации такого объекта является технология COM, предназначенная, в первую очередь, для повышения надежности и гибкости взаимодействия программных продуктов между собой. COM-технология не определяет структуру программного продукта, язык программирования и прочие детали реализации. Она определяет механизм взаимодействия COM-объектов между собой и относится к так называемым двоичным стандартам, обеспечивающим независимость от языка. Возможности расширения интерфейсов и их гибкого взаимодействия лежат в основе самой концепции COM: доступ к интерфейсам обеспечивается через уникальные идентификаторы интерфейсов *GUID* (Global Unique Interface Identifier), уникальность которых гарантирует операционная система. COM-объекты могут быть прозрачно друг для друга модифицироваться, поскольку доступ к объектам обеспечивается через GUID. Кроме того, в технологии COM предусматривается возможность регистрации COM-объектов в системном реестре. При регистрации в него записывается путь к модулю, содержащему объект и его глобальный идентификатор CLSID (Class Identifier). COM-объект можно причислить к одной или нескольким категориям, каждая из которых - это уникальный GUID, имеющий строковое название. Тем самым объекты одного типа можно организовать в одну категорию, что обеспечивает поддержку встраиваемости. Программе-клиенту достаточно найти зарегистрированные в реестре объекты определенной категории и подключить их. Модуль, содержащий COM объект, может размещаться в любом месте на диске, или же на другом компьютере, главное, чтобы в реестре был записан полный путь к нему.

Реализованные COM-объекты, удовлетворяющие вышеуказанным требованиям, являются промежуточным (вторым) уровнем в трехуровневой структуре программного комплекса, которая показана на рис. 2. Схема изображает взаимодействие объектов трехуровневой структуры с использованием COM-объектов.

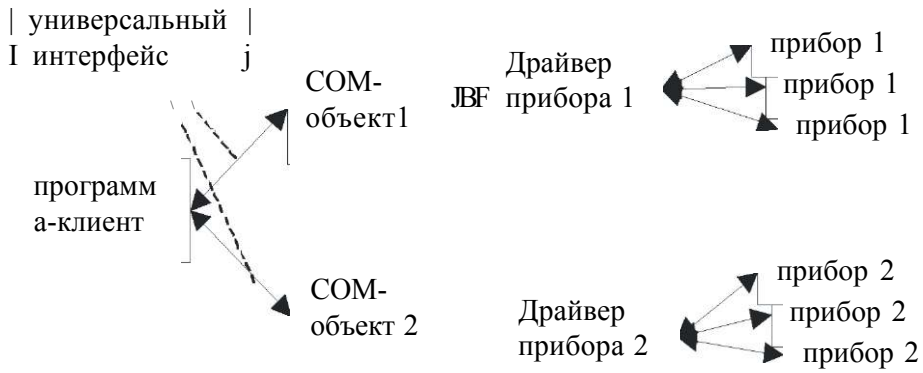

Рис 2. Трехуровневая структура многомодульного измерительного программного комплекса

Первый уровень - это драйверы устройств. Они предоставляют набор функций для работы с приборами. В пределах одной линейки приборов драйверы могут отличаться интерфейсом, но построены на основе одной концепции. Это позволяет создать объекты второго уровня, поддерживающие один общий внешний интерфейс.

Второй уровень - это COM-объекты, объединенные одной категорией и регистрируемые в реестре. Основная задача для программиста на этом уровне - разработать максимально универсальный внешний интерфейс для COM-объектов.

Третий уровень - клиентская программа, которая при инициализации осуществляет поиск COM-объектов в реестре по нужной категории. Она должна уметь работать с универсальным интерфейсом, предоставляемым COM-объектом.

Описанный подход был успешно реализован при создании программного обеспечения цифрового осциллографа. Были написаны две клиентские программы: первая - для работы с серией приборов Bordo, вторая - для работы не с конкретным прибором, а с абстрактными цифровыми сигналами, для математической обработки сигналов, поступающих из подключаемых COM-объектов. При этом было обеспечено сопряжение двух клиентских программ на основе общего COM-объекта.

Литература

1. *Трельсен Э.* Модель COM и применение ATL 3.0 Пер. с англ. - СПб.Ж BHV - Санкт-Петербург, 2000. - 928 с.