УДК 681.3.06

О ТЕХНОЛОГИИ МНОГОУРОВНЕВОЙ РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ

© 2002 г. С. Д. Махортов

Воронежский государственный университет

Общеизвестно, что разработка любого более или менее серьезного программного комплекса немыслима без применения какой-либо технологии программирования. Это обусловлено все возрастающими сложностью и объемами решаемых задач. При этом важнейшее место среди технологий занимают концептуальные средства, определяющие стиль и методы проектирования и разработки ПО. Существенной особенностью нашего времени явилось возникновение таких областей применения программных систем, для которых сопровождение ПО соизмеримо с его разработкой, т. е. никогда не наступает момент, когда “программа окончательно готова”. К такого рода областям относится, например, автоматизация финансово-хозяйственной деятельности предприятий, где непрерывные и существенные изменения в законодательстве постоянно держат эксплуатируемое ПО в стадии частичной разработки. Для подобных программных систем приобретает особенное значение не только определение последовательности создания отдельных его частей, но и вопросы о том, разработчики какого уровня и на каких этапах занимаются их созданием.

На взгляд автора, несмотря на эволюцию парадигм проектирования и написания кода, приведшую нас от модульного к объектно-ориентированному программированию (ООП) [1], по-прежнему актуальным является представление о двумерности структуры развивающейся программы [2, 3], согласно которому каждое расширение функциональности программы можно рассматривать как добавление определенного компонента (вертикального слоя), относящегося к некоторым уже сформированным образованиям (горизонтальным слоям). При этом изъятие вертикального слоя не приводит к катастрофе программы, а лишь обедняет ее функциональность. Каждый же горизонтальный слой программы является ее неотъемлемой частью. Даже если допустить, что программу удалось спроектировать в виде единственного объекта некоторого класса, то и в этом случае его публичные свойства и методы, непосредственно экспортирующие функциональность объекта, можно считать вертикальными слоями, а скрытые свойства и методы, опосредованно реализующие функциональность, — составляющими горизонтальные слои. Различие точек зрения на это представление состоит лишь в том, какие слои являются первичными — горизонтальные или вертикальные.

Однако математик, привыкший к абстрактному обобщению, задался бы вопросом: “почему только два измерения?”. В настоящей статье делается попытка ввести третье измерение в модель структуры расширяемой программной системы.

В программировании давно существует понятие уровня разработки. Так, множество существующих языков программирования, всегда разделялось по их уровню. Если представить в самом низу машинный язык, а в самом верху — разговорный человеческий, то какой-либо язык программирования относится к низкому или высокому уровню соответственно его относительному расположению в этой иерархии (ассемблер — язык низкого уровня, паскаль — высокого). При этом повышение уровня как правило облегчает разработку, но сужает ее возможности. Немного обобщим это понятие. Будем считать, что вся программная система разрабатывается сразу на нескольких уровнях, каждый из которых занимает свое место в вышеуказанной иерархии. Можно сразу же выделить 3 таких уровня (в порядке повышения): системный, проблемный, пользовательский.

Системный уровень разработки программного комплекса (не путать с операционной

системой, расположенной еще ниже, однако в иерархии это самый близкий к ОС уровень) — это базовый уровень, который содержит средства поддержки технологии разработки, и создается наиболее квалифицированными программистами, возможно, незнакомыми с конечной областью применения всего программного комплекса. Программные объекты этого уровня могут использоваться для создания не только одной, но и целого класса систем, основанных на данной технологии.

Проблемный уровень занимает позицию выше и содержит в основном реализацию решения конкретных задач на уровне программиста. Специалисты, создающие этот уровень, могут быть менее квалифицированными в программировании, однако должны разбираться в предметной области, для которой создается программная система.

Пользовательский уровень — самый высокий в смысле приближения к уровню мышления обычного пользователя. Разработку на этом уровне часто называют настройкой, однако ее сложность и объемы порой не уступают разработке, особенно это касается упомянутых выше систем автоматизации. Специалисты этого уровня могут быть экспертами в предметной области, но иметь лишь приближенное представление о программировании.

Приведенная классификация является наиболее общей, идеализированной и может уточняться по мере развития и реализации технологии. В частности, пользовательский уровень может состоять из нескольких подуровней, характеризующих степень образованности пользователя в программировании. В дальнейшем также оказывается, что уровни могут пересекаться. Кроме того, наша классификация предполагает, что исходный язык программирования является компилируемым по крайней мере в близкий к машинному код. Конечно, режим интерпретации программ имеет свои существенные преимущества в плане возможностей отладки [4], сопровождения и пр. Однако для нас эффективность по крайней мере базовой части и наиболее тонких в смысле сложности алгоритмов мест является определяющей.

Описываемая технология, как и почти любая другая технология программирования, нуждается в программных же средствах поддержки. Рассмотрим их реализациию автором для операционной системы Windows с использованием среды разработки Delphi 5 и сервера баз данных Interbase 6. При этом следует иметь ввиду, что выбранные средства не являются единственно возможными в этом плане. По-видимому, аналогичные построения можно реализовать в C# / .Net или еще какой-либо среде, поддерживающей компонентную модель программирования. Выбор возможных серверов БД еще более широк. В дальнейшем под термином “компонент” мы подразумеваем компонент в смысле Delphi.

Итак, первоначально на вышеупомянутом системном уровне разработки создается основа — базовое приложение, содержащее минимальный интерфейс пользователя и средства его формирования, а также механизм добавления/исключения компонентов. Этот механизм умеет загружать пакеты компонентов, регистрировать компоненты, настраивать их свойства, вызывать методы. Таким образом, уже на этом этапе программа без дальнейшей корректировки ее базового кода становится неограниченно расширяемой, ведь подключить к ней можно любой пакет компонентов Delphi, разработкой которых занимаются сотни фирм и тысячи программистов всего мира. На следующем этапе разрабатывается и добавляется к базовому приложению визуальный дизайнер форм, позволяющий создавать новые формы из загруженных компонентов. Одним из возможных путей здесь, по которому пошел и автор, является моделирование в базовом приложении исходной среды разработки (IDE Delphi), в которой компоненты легко переводятся в режим дизайна. Основной метод здесь – изучение полудокументированных и недокументируемых интерфейсов среды и их реализация в специальном классе. Полезными источниками информации в этом плане явились книги [5, 6]. При загрузке базовым приложением пакетов Delphi кроме регистрации самих компонентов регистрируются также связанные с ними редакторы, что позволяет в режиме дизайна форм достичь полной визуальности, во многом не уступающей Delphi. Разрабатываемые пользователем формы можно сохранять как на клиентском компьютере (“локальная настройка”), так и в общей базе данных (“гло-

бальная настройка"). В базовое приложение (на системном уровне) включаются также несколько классов наиболее часто используемых в данной области применения стандартных форм (форма-дерево, форма-таблица, форма-дерево/таблица), которые будут служить основой для других уровней разработки. На начальном этапе реализуется также как базовый стандартный набор операций с БД (соединение, сохранение/восстановление, выполнение SQL-команд). Вообще в базовое приложение рекомендуется включить возможности, которые могут потребоваться в дальнейшем в любой разрабатываемой системе проектируемого класса.

Следующим шагом является создание и распространение с базовым приложением **проблемных компонентов**. Среди них может содержаться, например, компонент "экспортер бухгалтерской проводки" или "расчет налога". Теперь, когда компоненты могут подключаться без программирования, пользователь сам сможет собрать нужную конфигурацию программы (например, для конкретного рабочего места бухгалтерии) и в дизайнере форм разработать, например, новую форму первичного бухгалтерского документа не с нуля (хотя и это возможно в нашей системе). Тираж распространения компонента будет соизмерим с тиражом разрабатываемой программы, причем компонент будет использоваться для разработки на пользовательском уровне. Создание же самих проблемных компонентов осуществляется по нашей классификации программистами на проблемном уровне. Проблемные компоненты можно создавать более адаптированными к пользователю, чем обычные компоненты (назовем их системными). Разработанный дизайнер форм способен распознавать проблемные компоненты и показывать названия их свойств и методов в пользовательской интерпретации (в том числе на русском языке). Заметим, что системы пользовательского программирования путем сборки из заготовок глубоко разработаны Е. А. Жоголевым и его учениками [7] и в их исследованиях называются **системами обосновательного гиперпрограммирования.**

Далее выясняется, что помимо установки свойств и возможности вызова методов компонентов требуется обрабатывать связанные с ними события. Чтобы это было возможным без изменения исходных текстов базового приложения, разрабатывается (другой возможный вариант — используется готовый) скриптовый язык (например, подмножество Pascal), на котором можно писать обработчики событий и другие программные части, сохраняемые в разрабатываемых формах. Конструкции языка в целях адаптирования к пользователю должны допускать возможность переименовывания (в том числе по-русски). Реализуется двусторонняя связь опубликованных свойств и методов компонентов с переменными скриптового языка. Из соображений эффективности важно, чтобы этот язык допускал расширения в виде пользовательских функций (UDF), написанных на обычном языке программирования (например, Delphi) и подключаемых в виде DLL. Реализуется также редактор скриптовых программ с возможностью подсветки синтаксиса (с выбором языка — Pascal или SQL).

На данном этапе пользователь уже имеет неограниченные возможности для дальнейшей разработки программ без изменения базового кода. Единственное, что этому может препятствовать — он должен быть в определенной мере программистом. Ему необходимо иметь представление о свойствах и методах используемых компонентов, знать скриптовый язык. В качестве дальнейшего повышения пользовательского уровня разработки можно предложить использование проблемно-ориентированной транслирующей экспертной системы.

Важным моментом данной реализации является обеспечение начиная с некоторого момента возможности параллельной разработки системы на всех уровнях, возможности работы на более низком уровне для повышения эффективности разрабатываемой системы.

Описанная технология успешно применяется при разработке системы автоматизации учета кадров и зарплаты крупного завода г. Воронежа.

Почему же изложенную схему разделения на уровни можно считать третьим измерением в модели структуры разработки программ? Во-первых, потому, что на каждом из рассмотренных уровней может применяться упомянутое выше представление о слоях. Во вторых, при должной технологической поддержке являются взаимозависимыми вдоль

уровней (по крайней мере, в сторону повышения), соответствующие горизонтальные и вертикальные слои. Однако более глубокое исследование этого вопроса не является предметом данной статьи.

Наконец, уточним, что приведенное разделение уровней основано на личном опыте автора и его субъективном видении процессов разработки, поэтому возможны другие классификации, что не противоречит самому факту существования третьего измерения.

ЛИТЕРАТУРА

1. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Пер. с англ. — М.: Бином, 2000. — 560 с.

2. *Фуксман А. Л.* Технологические аспекты создания программных систем. — М.: Статистика, 1979. — 184 с.

3. *Горбунов-Посадов М. М.* Расширяемые программы. — М.: Полиптих, 1999. — 336 с.

4. *Калечиц В. Е., Лободин Н. И., Махортов С. Д.* Система интерактивной и пакетной отладки в режиме эмуляции // В сб.: Математическое моделирование и программное обеспечение в САПР. — Горький: ГГУ, 1984. — С. 42—47.

5. *Лишнер Р.* Секреты Delphi 2 / Пер. с англ. — К.: НИПФ “Диасофт Лтд”, 1996. — 800 с.

6. *Орлик С. В.* Секреты Delphi на примерах. – М.: Бином, 1996. — 316 с.

7. *Жоголев Е. А.* Объектная организация систем гиперпрограммирования // Программирование, 1997, № 5. — С. 24—32.