Заметки о формировании сборок и компонентах Блэкбокс

Темиргалеев Е. Э.
(30.05.2012)

1. Общие соображения

Сборка — готовый⇨подразумевается, что может потребоваться процесс развёртывания⇦ к использованию для решения определённой задачи набор компонентов. В общем случае подходящие компоненты, из которых формируется сборка, требуют доводки.

С технической точки зрения, *компонент* — неизменяемая сущность [1, 4.1], внутренне состоящая из:

- кода, реализующего интерфейс экспорта компонента;
- ресурсов (файлы данных и пр.), используемых при работе компонента;
- *конфигурации* — состояния при загрузке. Компонент определяет конфигурацию по-умолчанию. Перманентные изменения конфигурации хранятся вне компонента ("внешние" по отношению к компоненту файлы данных и пр.; если, например, есть файл конфигурации по-умолчанию, то он считается ресурсом, частью компонента).

С прикладной точки зрения, компонент предназначен для решения некоторой задачи.

Доводка компонента под задачу сборки заключается в:

- настройке (компонент не меняется) — формирование соответствующего файла конфигурации или активация специального кода, выполняющего динамическую настройку;
- модификации компонента (например, когда решаемая им задача "далека" от потребностей сборки).

Для ясности нужно различать модификацию компонентов сборки, ориентированные именно на сборку и, возникающие в рабочем порядке прочие, такие как:

- *сопровождение* (исправление ошибок);
- общая доводка оригинального компонента, когда велика вероятность в повторном востребовании модифицированной версии в других сборках (например, доводка компонента до внутреннего стандарта организации с последующим использованием его во всех сборках);
- …

Такая классификация так или иначе будет субъективна, но суть в том, что она должна быть явно выполнена *наладчиком* сборки. И соответствующим образом оформлена. В идеале прочие изменения нужно выносить за пределы сборки.

Помимо набора компонентов сборка может содержать определённую *среду выполнения* [2, 1]. С точки зрения наладчика сборки среду, как предмет доводки, можно считать одним из компонентов. Аналогично стоит рассматритвать и сборки, включаемые в налаживаемую.

Для полного контроля с технической стороны и полноценного сопровождения сборки все её составные части должны быть доступны в неизменном виде. Соответственно, для используемых сторонних компонентов нужно заводить внутренний репозиторий, содержащий их оригинальные версии. "Рядом" же имеет смысл хранить результаты модификации этих компонентов в процессе внутреннего сопровождения и модификации общего плана (для повторного использования в разных сборках).

2. Схема

(пункты 2-5 — по необходимости)
1) Выбрать базовые компоненты и/или сборки.
2) Исключить из используемых сборок лишние компоненты. Исключение может потребовать модификацию компонентов и коррекцию настроек.
3) Выполнить ориентированную на сборку модификацию компонентов. Может потребовать коррекцию настроек.
4) Выполнить окончательную коррекцию настроек.
5) Составить инструкцию по развёртыванию. Может включать, например:
- команды компиляции, если компоненты идут только в исходных текстах;
- требования к окружению определённых компонентов или среды выполнения;
- список необходимых или рекомендуемых настроек окружения. Например, когда в качестве подразумеваемого окружения выступает Блэкбокс, это могут быть рекомендуемые команды меню, команды регистрации конвертеров в *Config*.

3. Блэкбокс как сборка

Оригинальный выпуск Блэкбокс от разработчика [3] — сборка, обеспечивающая средства разработки компонентного ПО. Включает ([2, 3], [1, 4.4]):

1) Платформенно-зависимую среду выполнения для компонентов — модулей ЯВУ Компонентный Паскаль. Модуль КП — атомарный компонент. Более "крупные" компоненты суть сборки из нескольких модулей, взаимодействующих по приватным (не предназначенным для использования клиентами) интерфейсам. Для оформления таких сборок появился определённый набор соглашений [4, 3.2] и термин — *подсистема*.

2) Компонентную библиотеку (и её платформенно-зависимую реализацию) с набором низкоуровневых средств (файлы, метапрограммирование и пр.).

3) Компонентный каркас (и его платформенно-зависимую реализацию), ориентированный на разработку пользовательских интерфейсов, — поддерживает концепцию составных документов.

4) Набор компонентов (см. Рис. 4-6. [3, 4.4]) — Text (составные текстовые документы), Form ("гуй"), Dev (средства разработки), Sql (доступ к СУБД посредством SQL).

Типовые "настройки":

1) Меню [5, 8]. Настройка среды. Для Windows:
- конфигурация по-умолчанию задаётся *HostMenus.CreateDefaultMenu*;
- текущая конфигурация распределена по файлам ресурсов подсистем */Rsrc/Menus.

2) Текстовые ресурсы [5, 9] — стандартное назначение: локализация. Доступны посредством библиотечного компонента *Dialog* из */Rsrc/Strings. При стандартном использовании — ресурсы компонентов. Обособлены только для подсистем.

3) *Config.Setup* [6] — команда настройки среды при старте. Предназначена для динамической настройки компонентов (текущая конфигурация которых не хранится перманентно).

4) *Converters.list* [7] — настройка компонента *Converters* (компонентный каркас Блэкбокс) — список конвертеров документов; по-умолчанию содержит конвертер стандартного документа. Настройка выполняется динамически; стандартный метод — регистрировать дополнительные конвертеры в *Config*.

4. Выводы и заметки к последующей работе на тему "сборок"

В процессе работы над зафиксированным выше материалом появилось другое понимание некоторых моментов:

- сборка, под которой изначально подразумевалась определённая сборка Блэкбокс, суть конкретный экземпляр компонентного ПО [2, 1];
- для обеспечения возможности оформлять компонент как неизменяемую сущность,

которую достаточно добавить к системе Блэкбокс простым копированием, при учёте сложившейся практики помещать в некоторые подсистемы технически независимые компоненты, достаточно решить задачу обособленного указания текстовых ресурсов компонента. Принимаемые ранее в контексте этой задачи во внимание меню (настройка среды), регистрация конвертеров (настройка *Converters*) — не имеют отношения к добавляемым в среду компонентам.

- текстовые ресурсы нужно использовать по назначению — как ресурсы. (У автора сложилась практика размещения там настроек вида *ключ = значение*. Смешивание в одном документе данных которые являются "ресурсами" и текущими настройками — нарушает концепцию компонента.)

Автор уже не первый раз сталкивается с такими моментами, как появление нового уровня понимания вроде бы очевидных на первый взгляд вещей, или забывание ранее понятого. Возникает ощущение нехватки общетеоретического, чёткого описания темы КПО, которое бы "вправило мозги" раз и навсегда, зафиксировало стандартную терминологию (вводимая и используемая выше (пункт 1) — приблизительная), а так же методик разработки КПО. (Задача поиска подобных материалов не решалась.)

В свете попытки хотя бы отчасти систематизировать методику формирования КПО, нужно отметить, что чёткой инженерной стандартизации не хватает и для разработки традиционного ПО (учитывая всем известное на данный момент качество среднестатистического продукта и подход производителей к ответственности за него, 28.05.2012). Предположительно, переход на более простое конструктивно КПО должен упростить этот вопрос, а местами и снять.

Например, для домашнего использования (в идеале) пользователь интуитивно подбирает нужный для его задачи набор компонентов, подстраивает их и пользуется.

Но для сфер, где требуется гарантированная надёжность, требуется и систематический подход к сборке, эксплуатации и контролю за качеством КПО, решающего поставленную задачу. Предположительно, он может выглядеть так:

- используются компоненты, выпущенные "по ГОСТу" — со стандартизированной документацией, прошедшие соответствующее тестирование на соответствие, и пр. За выполнение компонентами документированных контрактов отвечает приёмная комиссия и производитель.
- из набора доступных компонентов по стандартизированным методикам, согласно документации компонентов, осуществляется выборка необходимых для целевого программного комплекса компонентов, выполняется сборка комплекса, настройка и тестирование с последующей приёмкой. (Вопрос разработки дополнительных компонентов опускаем.)
- процесс сборки осуществляет и за результат отвечает человек соответствующей квалификации, предположительно инженерной специальности. (О термине для этой специальности была поднята тема (http://forum.oberoncore.ru/viewtopic.php?t=3979) на конференции OberonCore. Однако, из-за неточной постановки задачи, обсуждение ушло в сторону...)

Список источников

1) Пфистер К. Компонентное программное обеспечение. Перевод Ермакова И. Е. // URL: http://oberoncore.ru/library/component_soft
2) Темиргалеев Е. Э. Компонентное ПО и "текст как интерфейс". Примеры в Блэкбокс. // URL: http://oberoncore.ru/library/temir_comp-soft_text-ui_bb-ex
3) BlackBox Component Builder 1.6-rc6 (Win). // URL: http://oberon.ch/blackbox.html
4) BlackBox Component Builder 1.6-rc6 (Win). Документ "Docu/Tut-3.odc".
5) Там же. Документ "System/Docu/User-Man.odc".
6) Там же. Документ "System/Docu/Config.odc".
7) Там же. Документ "System/Docu/Converters.odc".