Управляемая программная архитектура

Куриленко И.Е., ivan@iosys.ru, Борисов А.В. borisov@iosys.ru,
Чернышева Н.В. n.chernysheva@aamsystems.ru
ФГУП ГосНИИ «Операционных систем»

В настоящий момент значение информационных технологий для различных сфер деятельности сложно переоценить. Как известно, целый ряд областей хозяйственной и экономической деятельности являются IT критичными. Среди них особое место занимают телекоммуникационная, финансовая сферы, и области добычи и переработки природных ресурсов, энергетика. Осознание влияния информационных технологий на общую эффективность деятельности привело к широкому применению интегрированных решений и технологий, ориентированных на сложные корпоративные среды.

Описанные тенденции способствуют повышению стабильности и эффективности информационных систем и инфраструктур, однако их нельзя считать панацеей. К сожалению, в некоторых случаях при внедрении новых IT решений за высокоуровневыми стратегиями теряется прикладной момент (построение прикладного приложения, работающего в интегрированной среде). С учетом того что, высокоуровневые концепции зачастую осознанно не раскрывают подробностей реализации, а лишь указывают общие подходы, отсутствие четкого видения в вопросах модификации архитектуры существующих решений приводит к увеличению рисков при реализации проектов. Это становится особенно важным в случаях наличия в IT инфраструктуре большого количества разнородных решений, пусть и интегрированных между собой. С учетом того, что требования к IT инфраструктуре предприятия и нагрузка на ее узлы меняется, построение гибкого решения для столь неоднородной среды является особенно актуальной.

В данной работе предлагается подход к построению приложений в рамках интегрированного решения. Подход предполагает применение управляемой программной архитектуры как инструмента, адаптирующего существующие IT решения к изменяющимся требованиям. Программная архитектура – это структура программного средства, описанная в терминах среды разработки и той программной платформы, на которой оно будет развернуто [1]. Под программой с *управляемой программной архитектурой* будем понимать такую программу, которая построена из независимо разрабатываемых стандартизированных компонент и способна автоматически настраивать собственную логику работы путем изменения числа

используемых компонент и взаимосвязей между ними во время исполнения. Внесение новых компонент и дополнительных возможностей в такую программу не требует перекомпиляции всех остальных компонент, а также ручного связывания компонент между собой. Целями разработки программ по такой архитектуре являются:

- сокращение времени разработки;
- повышение качества программы;
- глубокая стандартизация компонент программного средства и получение возможности применения средств автоматизации и кодогенерации;
- автоматизация процесса построения программных средств;
- возможность автономного тестирования отдельных компонент;
- простота модификации приложения при изменении требований;
- получение возможности автоматической баллансировки нагрузки на вычислительные узлы при работе в распределенном режиме;
- автоматическое версионирование и упрощение процесса выпуска редакций программого средства (версий с разными возможностями).

Для обеспечения этих возможностей программное средство должно быть построено из компонент, реализующих некоторый полезный функционал опираясь на возможности, предоставляемые *средой исполнения программ с управляемой программной архитектурой.* Основные элементы этой среды показаны на рис. 1. В задачи среды исполнения входит:

- обеспечение исполнения приложений, построенных по управляемой программной архитектуре;
- хранение регистрационных данных компонент;
- хранение конфигурации компонент;
- обеспечение унифицированного механизма получения экземпляра компонента;
- обеспечение унифицированного механизма взаимодействия компонент между собой (в т.ч. и сетевого);
- генерация приложений;
- мониторинг нагрузки и назначение процессов узлам.

Физически, среда исполнения может быть реализована как универсальная прослойка, поставляемая в виде компонента операционной системы. В рамках данного исследования предполагается создание такой среды как службы (service) для операционной системы MS Windows.

Рассмотрим процесс генерации приложений. Под *схемой S* будем

понимать тройку S=(C,R,O), где C - множество компонент, R – множество связей между компонентами и O – множество настроек компонент. Каждый элемент множества R является тройкой (C_i,Role, C_j), где C_i и C_j являются компонентами из множества C, а Role – символическое имя роли, и обозначает, что к компоненту C_i подключен компонент C_j в роли Role. Множество O имеет вид ({id_1, <O_1,..O_N>}, …{id_k, <O_1,..O_N >}). Это множество позволяет определить для каждого экземпляра схемы, заданного уникальным идентификатором (id_1, id_2, …) настройки каждого компонента из множества C (<O_1, .. O_N >). Схемы допускают вложенность, то есть одна схема может входить как компонент в другую схему.

Ри

с.1. Основные элементы среды исполнения

На физическом уровне *приложением с управляемой архитектурой* является совокупность схемы и ее загрузчика. Схема физически хранится в базе конфигурации. Загрузчик схемы является программным модулем, обеспечивающим загрузку этого описания и создание по нему всех входящих в схему компонент и связей между ними. Таким образом, распространяя загрузчик и схему приложения, можно обеспечить запуск различной логики, а изменение логики работы программы на отдельном узле системы, свести к изменению схемы. Таким образом, в зависимости от требований из одних и тех же программных компонент может быть получено множество приложений, реализующих те или иные возможности. При этом получение схемы может выполняться как разработчиком на этапе разработки, а приложение поставляться как приложение с фиксированными возможностями и жесткой логикой работы (традиционный подход), так и на этапе эксплуатации системы на основе требований, задаваемых администратором через конфигуратор

(рекомендуемый подход).

Администратор конфигурирует приложения путем указания типа приложения и установки ролевых связей этих приложений с другими компонентами системы (например, драйверами устройств). Схемы приложений создаются автоматически генератором (рис. 2), который заносит в них все компоненты, указанные пользователем. Затем по таблице зависимостей (хранящей описание какие компоненты должны быть включены в схему вместе с компонентом типа C_i) в схему заносятся дополнительные служебные компоненты, которые необходимы для работы программы, но не конфигурируются пользователем. После чего по таблице связей осуществляется наполнение связей в схеме.

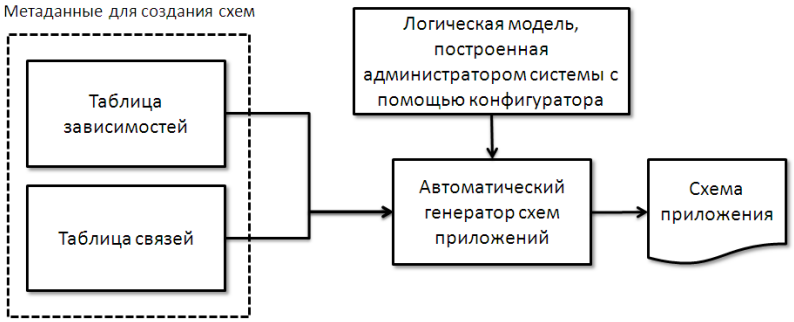

Рис. 2. Автоматический генератор схем

При этом администратор обладает знаниями в предметной области, для которой используется система, и строит ее логическую модель, располагая службы и устройства, и не вдается в подробности программной реализации. Подробнее о таком использовании можно посмотреть в работах [2,3].

Литература

1. Бахтизин В.В., Неборски С.Н. «Создание управляемой программной архитектуры» - Программные продукты и системы № 3, 2005 г, С. 2-5.
2. Борисов А.В., Куриленко И.Е., «Модель временных рассуждений в распределенной системе учета автотранспорта», Информационные технологии моделирования и управления №5 2005, - С. 786-794.
3. Борисов А.В., Куриленко И.Е., «Применение управляемой программной архитектуры при разработке систем автоматизации парковочных комплексов», Тр. междунар. науч.-техн. конф. «Информационные средства и технологии МФИ-2007». – М.: МЭИ, 2007. – Т.3. – С. 35-38.