Типы компонентных структур

Компонентная модель – отражает многочисленные проектные решения по композиции компонентов, определяет типы паттернов компонентов и допустимые между ними взаимодействия, а также снижает время развертывания программной системы в среде функционирования.

Каркас – представляет собой высокоуровневую абстракцию проекта программной системы, в которой отделены функции компонентов от задач управления ими. То есть, бизнес–логика – это функции компонентов, а каркас задает правильное и надежное управление ими. Каркас объединяет множество взаимодействующих между собою объектов в некоторую интегрированную среду для решения заданной конечной цели. В зависимости от специализации каркас называют “белым или черным ящиком”.

Каркас типа “белый ящик” включает абстрактные классы для представления цели объекта и его интерфейс. При реализации эти классы наследуются в конкретные классы с указанием соответствующих методов реализации. Использование такого типа каркаса более характерно для ООП.

Для каркаса типа «черный ящик» в его видимую часть выносятся точки, разрешающие изменять входы и выходы.

Композиция компонентов включает четыре возможных класса:

– композиция компонент–компонент обеспечивает непосредственное взаимодействие компонентов через интерфейс на уровне приложения;

– композиция каркас–компонент обеспечивает взаимодействие каркаса с компонентами, при котором каркас управляет ресурсами компонентов и их интерфейсами на системном уровне;

– композиция компонент–каркас обеспечивает взаимодействие компонента с каркасом по типу «черного ящика», в видимой части которого находится спецификация для развертывания и выполнения определенной сервисной функции на сервисном уровне;

– композиция каркас–каркас обеспечивает взаимодействие каркасов, каждый из которых может разворачиваться в гетерогенной среде и разрешать компонентам, входящим в каркас, взаимодействовать через их интерфейс на сетевом уровне.

Компоненты и их композиции, как правило, запоминаются в репозитории компонентов, а их интерфейсы к репозитарии интерфейсов.

Повторное использование в компонентном программирование в общем

случае представляет собой любые порции формализованных знаний, добытые во время реализации программных систем и используемых в новых разработках.

Повторно используемые компоненты (ПИК) – элементы знаний о минувшем опыте разработки систем, которые могут использовать не только их разработчиками, но и путём адаптации к новым условиям. ПИК упрощает и сокращает сроки разработки новых программных систем. Высокий уровень стандартизации и распространение электронных коммуникаций (сети Интернет) обеспечивает довольно простое получение и широкое использование готовых компонентов в разных проектах за счет:

– отражения фундаментальных понятий приложения;

– скрытия способа представления и предоставления операций обновления и получения доступа;

– обработки исключительных операций в приложении.

При создании компонентов, предназначенных для повторного использования, общий интерфейс должен содержать операции, которые обеспечивают разные способы использования компонентов. Возможность повторного использования приводит к усложнению компонента, а значит к уменьшению понятности. Поэтому требуется некоторый компромисс между ними.

Главным преимуществом создания программных систем из компонентов является уменьшение затрат на разработку за счет:

– выбора готовых компонентов с подобными функциями, пригодными для практического применения;

– настраивания готовых компонентов на новые условия, которые связаны с меньшими усилиями, чем разработка новых компонентов.

Поиск готовых компонентов основывается на их классификации и каталогизации.

Метод классификации предназначен для представления информации о компонентах с целью быстрого поиска и отбора. Метод каталогизации обеспечивает физическое размещение компонентов в репозиториях для непосредственного доступа к ним в процессе интеграции.

Методология компонентной разработки систем

Эта методология включает в себя две основные фазы процесса разработки.

1. Разработка отдельных компонентов исходя из следующих требований:

– формализованное определение спецификаций интерфейсов, поведения и функциональности компонента;

– использование системы классификации для поиска и отбора необходимых

компонентов, а также для их физического размещения;

– обеспечение принципа повторности.

2. Интеграция (композиция) компонентов в более сложные программные образования:

– разработка требований (Requirements) к программной системе;

– анализ поведения (Behavioral Analysis) программной системы;

– спецификация интерфейсов и взаимодействия компонентов (Interface and Interaction Specification);

– интеграция разработанных компонентов и компонентов повторного использования (Application Assembly and Component Reuse) в единую среду;

– тестирование компонентов и среды;

– развертывание (System Deployment) программной системы у пользователя;

– поддержка и сопровождение программной системы (System Support and Maintenance).