Применение компонентов .NET в создании отказоустойчивого программного обеспечения

Маймистов Д.С.

Сибирский государственный аэрокосмический университет
им. академика М.Ф. Решетнева

Концепцию *мультиверсионного программирования* (МВП, multi-version programming – MVP), или *N-версионного программирования* (НВП, N-version programming – NVP) впервые представил Альгирдас Авижиенис в 1977. Основная идея МВП заключается в том, что для решения отдельных подзадач системы, используется несколько версий одного алгоритма, выполняющихся одновременно. Результаты работы этих алгоритмов анализируются, и из них выбирается один наиболее удовлетворяющий потребностям системы на данный момент времени. Выбор производится согласно внутренней логике системы. Таким образом, достигается повышение надёжности системы в целом. Различных подзадача в сложных системах, реализация, которых использует концепцию мультиверсионного программирования, может быть огромное множество. Очевидно, что для разработки таких систем необходима общая концепция и общий подход в написании алгоритмов, решающих её отдельные подзадачи. На эту роль как нельзя лучше подходит методика компонентного программирования.

Такой метод создания программного обеспечения, как компонентное программирование, появился относительно недавно. Его можно охарактеризовать как технологию создания программного обеспечения из готовых блоков. То есть программисты пытаются использовать идеи строителей, занимающихся крупнопанельным домостроением. Создание программного обеспечения из компонентов подразумевает, что компоненты будут добавляться к проекту во время разработки. При этом будет производиться их начальная настройка. Компоненты как таковые не подразумевают пользовательского интерфейса (ни для программиста, ни для конечного пользователя). В этом качестве выступают части интегрированной среды разработки и дополнительные программные дизайнеры. Первой компонентной средой был продукт, разработанный корпорацией Microsoft на заре своего существования. Впоследствии на его базе были разработаны множество других сред. Таким образом, к концу двадцатого века, компоненты стали поддерживаться почти всеми производителями интегрированных сред.

Самой развитой и совершенной компонентной моделью на сегодняшний день, является модель предложенной корпорацией Microsoft и реализованной ею в новой технологии .NET.

Определение компонента в понимании Microsoft - это объединенные в отчуждаемую форму исполняемый бинарный код и данные, которые могут использоваться для построения программных систем. Отчуждаемость подразумевает возможность использования компонента без дополнительных знаний о нем. На практике это означает, что компонент сам должен содержать сведения о себе. Компонент должен также иметь внешний (публичный) интерфейс. Интерфейс является как бы механизмом, через который можно запустить находящийся внутри компонента код. Отчуждаемость также означает, что экземпляр компонента может быть создан динамически, и что для этого не обязательно использовать всякого рода компиляторы и интерпретаторы.

По сути компонент - это класс, предоставляющий информацию о себе (метаинформацию), экземпляры которого можно создавать динамически (не имея никакой статической информации о нем).

Практически любой класс в .NET отвечает этим требованиям - метаинформация создаются для любого элемента класса (будь он трижды скрытым), экземпляр любого

класса можно динамически создать, и любой класс помещается в сборки (один или более исполнимых модулей), которые можно распространять независимо. Таким образом, получается, что любой класс в .NET может выступать как компонент. Но на самом деле это не так. И причиной тому наличие в библиотеке .NET отдельного класса Component. Любой класс, что бы иметь возможность взаимодействовать с интегрированной средой разработки должен быть унаследован от класса Component.

На основе выше приведённого описания основных концепций компонентной модели .NET, можно сделать вывод о том что .NET компоненты обладают следующими преимуществами по сравнению с компонентами, в основе которых лежат иные концепция и технология:

- Возможность интегрировать компонент в любую среду разработки, поддерживающую соответствующие стандарты Microsoft
- Возможность написания и распространения компонент сторонними разработчиками
- Возможность написания компонент в различных средах разработки и на различных языках программирования, поддерживающих соответствующие стандарты Microsoft

Таким образом, становится очевидным выбор в пользу использования компонентной технологии .NET, для разработки мультиверсионных компонент.

Список литературы:

1. Владислав Чистяков. «.Net – классы, компоненты и контролы» RSDN Magazine №3 2003г.
2. Котенок А.В. Построение среды мультиверсионного исполнения программных модулей. Вестник НИИ СУВПТ: Сб. научн. трудов; Красноярск: НИИ СУВПТ.- 2003. Вып. 14.- С. 13-21.