Суворов Р.Е., Петров А.В. Применение инструмента Enterprise Core Objects при разработке прикладных программ. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей VIII Всерос. научно-техн. конф. – Пенза: ПДЗ, 2008. – С. 93-95.

ПРИМЕНЕНИЕ ИНСТРУМЕНТА ENTERPRISE CORE OBJECTS ПРИ РАЗРАБОТКЕ ПРИКЛАДНЫХ ПРОГРАММ

Р.Е. Суворов, А.В. Петров

Рыбинская государственная авиационная технологическая академия им. П.А. Соловьева,
г. Рыбинск

Цель данной статьи – ознакомить читателя с опытом использования инструмента UML-моделирования и кодогенерации Enterprise Core Objects (http://www.CapableObjects.com, далее – ECO) при разработке учебной программно-аппаратной системы управления оборудованием на базе технологии Microsoft .NET Framework 3.5 и местом данного инструмента в процессе разработки ПО.

Enterprise Core Objects (ECO) – продукт шведской фирмы Capable Objects AB, предназначенный для разработки прикладного ПО с использованием подхода «Domain Driven Design». Основу ECO составляют:

1) инструменты UML-моделирования, полностью поддерживающие UML 1.1 (11 диаграмм);

2) транслятор UML диаграмм в C#\Delphi .NET код;

3) расширяемый набор сервисов, включая сервис объектно-реляционного отображения (ORM), сервисы управления состоянием объектов (сервис персистентности, сервис транзакций, сервис кэширования, сервис версионинга объектов, реализующий, в частности, функциональность «Undo\Redo»), сервис взаимодействия с ASP.NET для создания Web-приложений и т.д.

По умолчанию поддерживаются следующие серверы баз данных: MS SQL, InterBase, MySql, Oracle, DB2. Кроме того, ECO имеет возможность интеграции с такими распространенными IDE, как CodeGear RAD Studio и Microsoft Visual Studio. Что касается кодогенерации, то отметим особо, что ECO осуществляет слияние исходных текстов (код, написанный вручную, не удаляется), что позволяет удобно совмещать код, написанный программистом, и код, сгенерированный ECO. В текущей версии ECO возможна трансляция следующих UML-объектов**:**

- диаграммы классов, которая, будучи используемой вместе с диаграммой состояний объекта (State Chart), является основным инструментом разработки ПО с помощью ECO;
- языка объектных ограничений (OCL);
- диаграммы последовательности, раскрывающей логику взаимодействия и порядок обмена сообщениями между объектами.

Отдельно хотелось бы упомянуть механизм подписки, отслеживающий изменение атрибутов объектов и осуществляющий автоматический пересчёт вычисляемых полей классов. Особенностью ECO, которую некоторые относят к её недостаткам, является жесткая привязка данного инструмента к платформе Microsoft .NET Framework, что, однако, является довольно распространенным явлением в области UML-кодогенерации.

Можно выделить следующие недостатки ECO: отсутствие поддержки автоматической генерации диаграмм по написанному коду; уменьшение скорости работы программы в некоторых случаях (например, при использовании интерпретируемого языка OCL для обработки больших объёмов данных).

В качестве аналогов ECO можно рассматривать такие CASE-средства, как IBM Rational Rose, AllFusion Process Modeler 7 (ранее – BPwin), NHibernate, Visual Paradigm SDE. Наш выбор определили такие черты ECO, как хорошая интеграция с платформой .NET, стабильность данного продукта и ориентированность ECO не только на работу с данными, но и на разработку приложения в целом.

ECO используется нами для разработки учебной программно-аппаратной системы управления оборудованием. Система предназначена для обучения студента созданию параллельных программ управления цикловым и позиционным оборудованием на текстовом и графических предметно-ориентированных языках программирования. Известно, что основу эффективной организации разработки составляет та или иная методология процесса разработки ПО. На данное время существует значительное количество методологий разработки, однако, по нашему мнению, на практике необходимо применять принцип, выраженный Алистэром Коуберном – «Каждому проекту своя методология». В нашем проекте был применен итеративный подход к разработке, выделены определенные стадии и четко описаны результаты, которые должны быть достигнуты в результате каждой стадии при поддержке ECO (таблица). Основным критерием выбора стадий и ожидаемых результатов было четкое разделение уровней абстракции каждой стадии и степени её детализации.

Процесс разработки ПО и его поддержка UML (ECO)

<table>
<tr><th colspan="2" rowspan="2"></th><th rowspan="2">Стадии</th><th colspan="2">UML диаграммы и степень их детализации</th></tr>
<tr><th>Степень детализации</th><th>Документы</th></tr>
<tr><td rowspan="3">Уровни абстракции</td><td rowspan="2">Концепции</td><td>Анализ требований</td><td rowspan="2">Система представляется как черный ящик. Детализируется внешний по отношению к системе мир и его взаимодействия с ней.</td><td>Варианты использования (текст \ диаграммы)
Концептуальная диаграмма последовательности</td></tr>
<tr><td>Анализ предметной области</td><td>Концептуальная диаграмма классов</td></tr>
<tr><td>Спецификации</td><td>Спецификация межкомпонентных взаимодействий

Спецификация внутрикомпонентных взаимодействий</td><td>Система представляется как множество компонентов. Компонент = модуль (подмодули) + интерфейсы. Модули представляются как черные ящики.</td><td>Интерфейсная диаграмма классов
Описание ответственности (текст \ интерфейсные</td></tr>
</table>

			Детализируются ответственности модулей и подмодулей.	*диаграммы взаимодействия)*
	Реализации	*Реализация*	*Система представляется как множество классов. Детально прописываются все отношения между ними, все атрибуты классов, области видимости.* *Методы классов создаются, исходя из тех сообщений, которыми классы обмениваются.* *При необходимости используются шаблоны проектирования.*	*Диаграмма классов.* *Диаграмма последовательности* *Диаграмма состояний* *OCL*