УДК 629.78.018

Б.Б. МИХНИЧ, С.В. ОЛЕЙНИК, Е.В. СОКОЛОВА

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

ЯЗЫКОВО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ СТЕНДОВЫХ ИСПЫТАНИЙ ПОДСИСТЕМЫ ДАННЫХ ПЛАТФОРМЫ СПУТНИКА МС-2-8

Рассматривается программное обеспечение для автоматизации стендовых испытаний подсистемы данных платформы спутника МС-2-8. Анализируются принципы языково-ориентированного подхода при создании программного обеспечения. Показано, что такой подход исключает промежуточное звено – программиста, позволяет технологу непосредственно описать ход вычислительного процесса, что повышает отказоустойчивость и гарантоспособность системы.

языково-ориентированное проектирование, предметно-ориентированный язык программирования, автоматизация испытаний, спутник

Введение

В настоящее время зарождаются дискуссии об актуальности применения языковоориентированного проектирования (ЯОП) при разработке программного обеспечения [1, 2].

Значительная часть усилий при разработке классическим методом уходит на уточнение постановки задачи, что предполагает активное взаимодействие разработчика с пользователем непрограммистом. Более того, пользователь зачастую и сам не имеет четкого представления о своих нуждах. Попытка выбрать для общения с ним формальный язык или специализированную компьютерную терминологию обречена на неудачу в силу языковой нестыковки. Нужна возможность работать в терминах концепций и понятий решаемой проблемы, вместо того чтобы переводить мысли в нотацию языка программирования общего назначения (классы, методы, циклы, ветвления и т.п.). Для этого нужен язык, специфичный для предметной области.

Конечный пользователь практически никогда не обдумывает проблему в наборах инструкций, он выражает её в словах, понятиях, концепциях, мыслях. Эта модель решения содержится в голове в виде взаимосвязанных концепций, специфичных для области, в которой работает пользователь и её можно считать решением и объяснить другому программисту с достаточной степенью детализации, чтобы он мог сесть и написать программу (скажем, на Java), решающую проблему. И не нужно выражать решение в терминах языка программирования – оно может быть в любой форме. Другими словами, должен быть способ использовать это представление как готовую программу, а не только как способ общения с другими программистами.

Целью этой статьи является освещение практической реализации программного обеспечения на основе данного метода. А также обзор уже существующих разработок и сред, использующих языково-ориентированный подход, оценка их надёжности и гарантоспособности.

Языково-ориентированное проектирование в наши дни

Сравним классический подход и языково-ориентированный [2]. На рис. 1 изображён классический подход к разработке программного обеспечения, например ООП. На рис. 2 представлено распределение трудозатрат при языково-ориентированном проектировании. Как в классическом подходе, который изображён на рис. 1, так

и в языково-ориентированном (рис. 2) сначала требуется формулировка решаемой задачи в виде модели решения.

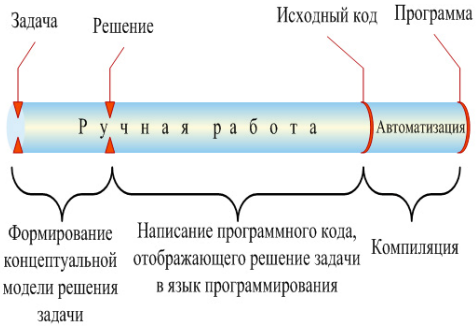

Рис. 1. Классический подход

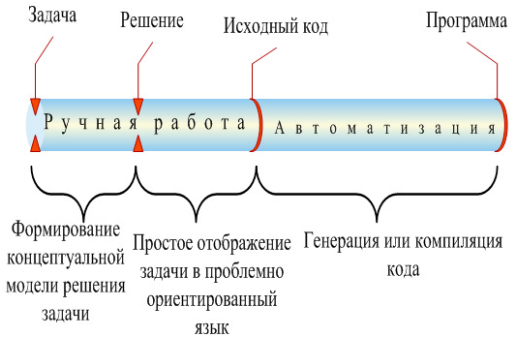

Рис. 2. Языково-ориентированное проектирование

На этом общее у них заканчивается вплоть до получения готового программного продукта. Преимущество языково-ориентированного подхода очевидно: отсутствует звено кодирования, вместо него используется работа в концепциях и понятиях предметной области, для которой создаётся данный программный продукт. Язык, специфичный для предметной области разрабатывается заранее, возможно привлечение более опытных экспертов данной отрасли и программистов, что повышает надёжность и гарантоспособность системы.

Уже достаточно давно существует такой стиль программирования, при котором мы стараемся описывать программные приложения с помощью языков, специфичных для соответствующей предметной области. Возьмём, к примеру, MatLab – это довольно сложный DSL (предметно-ориентированный язык программирования – англ. domain-specific programming language, DSL [3]), и работает он именно потому, что жестко сфокусирован на своей предметной области. Также можно упомянуть "малые языки" Unix, которые генерируют программный код посредством lex и yacc; конфигурационные XML файлы; активные модели данных – их еще называют "таблицы с метаданными", а программы - "программы, управляемые таблицами", в самой программе эти данные определенным образом интерпретируются и определяют поведение системы; Lisp – внутри которого разрабатываются языки, чаще всего используя его макросы; SQL [1].

Уже стали доступны инструменты, реализующие в себе языково-ориентированный подход:

- Software Factories от Microsoft ® Studio Visual Studio и IBM WebSphere – этот проект в настоящий момент представляет собой набор инструментов DSL Tools, доступных для скачивания в составе SDK к Visual Studio 2005 [4, 5];
- MPS – Meta Programming System, среда разработки, в которой можно описывать язык, ориентированный на решение конкретной проблемы и использовать его в выбранной предметной области. В этой среде существует базовый язык – для простейшей предметной области (арифметика, условные выражения, циклы), язык коллекций и язык пользовательского интерфейса [6].

Постановка задачи

Метод разработки программного обеспечения на основе ЯОП применялся при создании ПО имитации работы подсистемы данных платформы (ПДП) спутника МС-2-8. Конструкция спутника состоит из множества подсистем, которые взаимодействуют между собой. Перед выводом спутника на орбиту и началом его взаимодействия с другими наземными и космическими системами, требуется моделирование процесса полёта и работы бортового компьютера на Земле. ПДП спутника является координирующим звеном между другими системами, она принимает команды от одной системы спутника и передаёт её той системе, которой адресована данная команда.

Подсистема имеет широкий набор возможных команд, на базе которых и конструируются разнообразные технологические цепочки, реализующие сложные вычислительные процессы. Итак, ПО имитации работы ПДП относится к сложным проблемно-ориентированным комплексам и требует привлечения специалистов для повышения надежности и гарантоспособности.

Особенности программной реализации

Для представления данных была применена модель траекторий вычислительного процесса, в которой каждая прикладная задача представляется как функциональное преобразование множества допустимых исходных наборов данных, во множество результирующих наборов данных [7]. Среда компоновки сценариев даёт возможность технологу сформировать управляющие цепочки для тестовых проверок – на рис. 2 этот этап соответствует фазе простого отображения задачи в проблемно ориентированный язык; а также генерировать из них бинарные файлы, которые являются своего рода программой для спутника – этап компиляции.

Для создания нового сценария или выбора на редактирование существующего, используется форма, изображённая на рис. 3.

На рис. 4 представлена форма формирования сценария из базовых команд ПДП. Каждая из команд помимо уникального номера и типа обладает

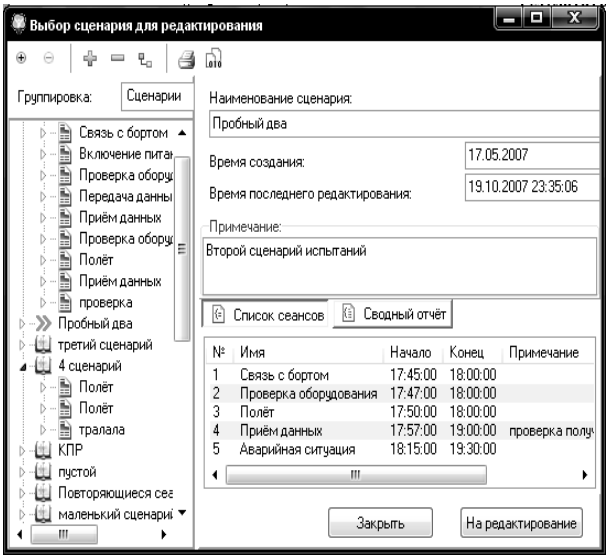

Рис. 3. Главная форма приложения в режиме менеджера сценариев

дополнительными характеристиками, которые задаются во время проектирования и влияют на поведение системы во время испытаний спутника.

Вынесение траекторий вычислительного процесса в базу данных позволило достичь одновременно удобного машинного представления и представления, понятного человеку.

Было предпринято разбиение базы данных на множество допустимых команд, выдаваемых ПДП, набор типовых сеансов, которые, в свою очередь, составляют сценарии поведения системы, для проведения испытаний. Эффективность данного метода также заключается в предложенном объединении команд в типовые последовательности и формировании из них сценариев.

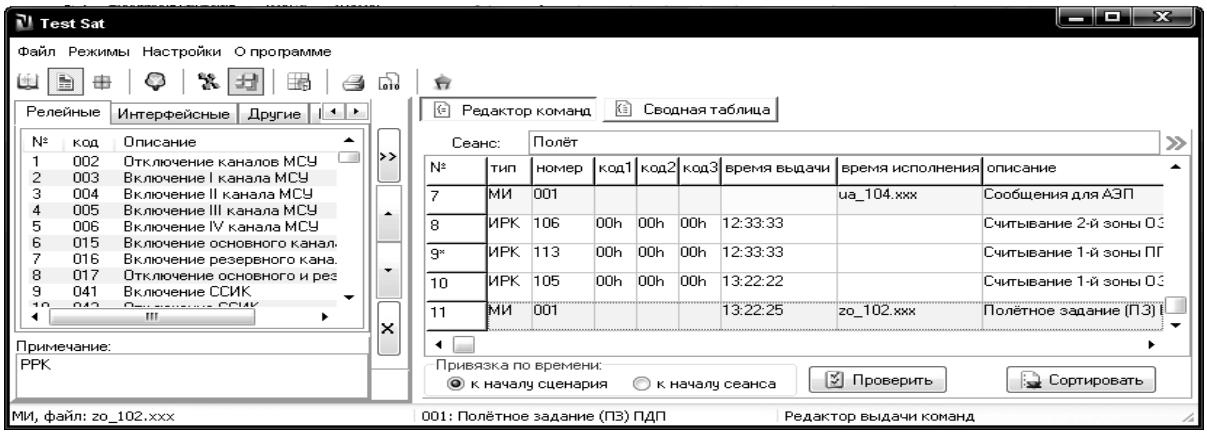

Рис. 4. Главная форма приложения в режиме организатора сеансов

Это реализует принцип повторного использования кода и снижает трудозатраты на описание схожих траекторий моделируемых вычислительных процессов.

Отображение базы данных в память осуществляется при помощи объектов классов. Базовые свойства и методы объектов реализованы в родительских классах и, при необходимости, переопределяются и расширяются в классах наследниках. Например, алгоритм работы со списками объектов типа TNameId был реализован один раз в классе TListNameId, и унаследован в классах, описывающих списки команд, сеансов и сценариев, которые являются наследниками типа TNameId.

Контроль технолога средой разработки сценариев на момент составления программы испытаний и автоматизация процесса генерации бинарных файлов позволяют обеспечить надёжность получаемых результатов.

Выводы

В результате исследований был на практике проверен языково-ориентированный подход в проектировании сложных проблемно-ориентированных комплексов, который позволил полностью перенести описания выполняемых действий, необходимых для реализации сложного проблемно-ориентированного комплекса из программной реализации во внешний источник информации, носителем которого является DSL.

Языково-ориентированное программирование даст возможность непрофессионалам самим писать код. Всевозможные эксперты в различных предметных областях смогут программировать, используя соответствующие DSL.

Данная разработка доказала, что при использовании языка, ориентированного на конкретную проблему трудозатраты на составление кода значительно уменьшаются, повышается качество конечного продукта и не требуется особой квалификации программиста от разработчика, он может быть экспертом той отрасли, для которой создаётся программное обеспечение.

Литература

1. Фаулер М., Языковой инструментарий: новая жизнь языков предметной области [Электронный ресурс]. – Режим доступа: http://www.maxkir.com/sd/languageWorkbenches.html.

2. Дмитриев С. Языково-ориентированное программирование: следующая парадигма [Электронный ресурс]. – Режим доступа: http://www.rsdn.ru/article/philosophy/LOP.xml,

3. Кириллов Д. Ориентация на язык // Компьютера. – 2006. – № 10. – С. 33-38.

4. Greenfield J., Short K., Software Factories. Assembling Applications with Patterns, Models, Frameworks, and Tools. – John Wiley & Sons, 2004.

5. Microsoft Enterprise Framework & Tools Group – Domain Specific Languages Toolkit [Электронный ресурс]. – Режим доступа: http://lab.msdn.microsoft.com/teamsystem/workshop/ dsltools/ default.aspx.

6. Сайт JetBrains MPS [Электронный ресурс]. – Режим доступа: http://www.jetbrains.com/mps.

7. Соколова Е.В. Проектирование сложных проблемно-ориентированных комплексов на основе модели траекторий вычислительных процессов // Радиоэлектронные и компьютерные системы. – 2007. – № 7(27). – С. 41-44.

Поступила в редакцию 18.01.2008

Рецензент: д-р техн. наук, проф. И.Б. Туркин, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.