А.В. Крючков
(Академия ГПС МЧС России; e-mail: hook66@list.ru)

ДОСТОИНСТВА И НЕДОСТАТКИ СОВРЕМЕННЫХ МЕТОДОВ СИНТЕЗА СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (часть 2)

Проведён анализ используемых в современной практике методов синтеза специального программного обеспечения.
Ключевые слова: программное обеспечение, методика разработки приложений.

A.V. Kruchkov

ADVANTAGES AND DISADVANTAGES OF MODERN SYNTHESIS METHODS OF SPECIAL SOFTWARE (part 2)

The analysis of usually used synthesis methods of special software is carried out.
Key words: software, method of applications development.

Статья поступила в редакцию Интернет-журнала 31 марта 2015 г.

Задачи синтеза специального программного обеспечения (СПО), составной части программного обеспечения (ПО) в автоматизированных системах управления производством (АСУП), требуют применения разных инструментальных средств разработки ПО: систем программирования (СП) и языков программирования (ЯП).

Данная статья является продолжением предыдущей обобщающей статьи с тем же названием, где анализируются достоинства и недостатки применяемых в настоящее время методик создания СПО.

Порождающее программирование (Generic programming – GP) – методика синтеза СПО, целью которой является "разработка и реализация порождающей доменной модели семейства систем" [4, с. 143]. Доменная модель – это модель выявленной на основе определенных методик предметной области, созданная в терминах компонент ООЯП. Порождающая доменная модель – это ориентированная на группу предметных областей модель данной предметной области в терминах компонент ООЯП, позволяющая выполнять генерацию кода для конкретных АРМ. GP "направлено на достижение самого высокого уровня автоматизации – автоматической сборки" [4, с. 144]. Необходимость в нем возникла тогда, когда стало очевидно, что синтез СПО при помощи CASE-систем имеет свои довольно жесткие ограничения, связанные с необходимостью хранения поколений настроек конфигураций. Его реализациями являются:

- родовое программирование;
- параметризованное программирование;
- компонентно-ориентированное программирование;
- метапрограммирование на основе шаблонов;
- аспектно-ориентированное программирование;
- ментальное программирование.

Для определения некоторых из данных методик потребуется определить другие методики синтеза СПО, на основании которых они были построены.

Родовое программирование – методика синтеза СПО, целью которой является создание шаблонов компонент ПО, из многочисленных сочетаний которых составляются эффективные программы [4, с. 174].

Достоинства:

- возможность повторного использования кода;
- представление предметных областей в виде коллекций абстрактных компонент;
- ориентация на создание СПО для семейств предметных областей.

Недостатки:

- универсализация пакетов компонент влечёт за собой увеличение количества компонент и их параметров, что в свою очередь приводит к чрезвычайному усложнению процессов на стадии реализации СПО;
- ограниченность знаний о представлении семейств предметных областей в виде коллекций объектов из-за чего, с одной стороны, некоторые значимые её характеристики не попадают в её описание и не реализуются в виде компонент ПО, а с другой – многие другие попадают, но не используются и создают чрезмерную сложность в их использовании;
- отсутствие чётких правил создания компонент СПО, их определения и формулировки;
- зависимость компонент от видения программистом семейств предметных областей и методов их реализации в библиотеках компонент ООЯП;
- необходимость повторного создания групп компонент СПО для разных семейств предметных областей.

Параметризованное программирование – методика синтеза СПО, целью которой является создание библиотек параметризованных компонент ПО [4, с. 196]. Параметризованный компонент – компонент ПО, обладающий определёнными параметрами, в зависимости от значения которых он частично изменяет свою функциональность. Методика создавалась Дж. Гогеном, специалистом по языку Ada, в котором широко применялись параметризованные модули. Ada стал той основой, на которой выстраивалась идея данной методики. Применено в STL (Standard Template Library) ЯП C++. Используется в Java в качестве библиотек параметризованных классов.

Достоинства:

- возможность повторного использования кода;
- явное представление спецификаций интерфейсов абстрактных компонент (в дальнейшем использовано в J2EE);
- получение параметризованных типов данных (компонент ПО), которые обеспечивают создание СПО для разных предметных областей.

Недостатки:

- универсализация пакетов компонент влечет за собой увеличение количества компонент и их параметров, что в свою очередь приводит к чрезвычайному усложнению процессов на стадии реализации СПО;

- отсутствие чётких правил создания компонент СПО, определения количества и типов параметров, которые необходимо использовать;

- зависимость компонент от видения программистом семейств предметных областей и методов их реализации в библиотеках компонент ООЯП;

- необходимость повторного создания групп компонент СПО для разных семейств предметных областей.

Программирование компонент – методика синтеза СПО, целью которой является создание повторно используемых компонент ПО. Для определённого семейства смежных предметных областей создаются компоненты СПО с целью их повторного использования [4, с. 24]. Легла в основу спецификации технологии COM (Common Object Model), разрабатывавшейся изначально для поддержки составных документов (текст, графика, электронные таблицы). Развитием данной методики (технологии) стала распределённая модель DCOM (Distributed COM) корпорации Microsoft. Целью её было создание технологии динамически активизирующихся компонент, содержащихся в DLL. Основная идея распределённых компонент ПО заключалась в том, что объекты создавались кем угодно и где угодно. Публикация состояла в необходимости указания самого интерфейса, с помощью которого к данному объекту обращались с удалённой персональной ЭВМ (ПЭВМ), а также места его нахождения в сети. При обращении при помощи RPC (Remote Procedure Call) – удалённого вызова процедуры – код процедуры (метода объекта) не загружался на ПЭВМ вызова, а исполнялся на ПЭВМ, содержащей объект, возвращая только результат выполнения. Обзор данной технологии приведён в [5, с. 572], а также в [6, с. 529]. В последствии на базе технологии COM были созданы технологии ODBC (Object Database Connectivity), OLE (Object Linking and Embeding), ADO (Active Data Objects), предназначенные для работы с БД, списками объектов и для связи с объектно-ориентированными ЯВУ. Помимо этого развитием технологии DCOM стала разработанная Microsoft технология ".*NET*".

Достоинства:

- проявление меньшей зависимости СПО от предметной области по сравнению с методами ООА ПО и ООП ПО;

- возможность повторного использования кода;

- быстрая адаптация к ряду требований семейства предметных областей;

- возможность построения семейств программ.

Недостатки:

- отсутствие поддержки в популярных ООЯП и, как следствие, необходимость создания отдельных библиотек компонентов для каждого из них;

- отсутствие четких правил создания компонент СПО, их определения и формулировки;

- различие реализации одинаковых компонент для различных групп предметных областей, ООЯП и программистов.

Метапрограммирование на основе шаблонов – методика синтеза СПО, аналогичная GP, целью которой является представление типов данных в виде шаблонов. Она заключается во вставке генераторов кода в компилируемые библиотеки ЯП [4, с. 386]. В этом типе программирования статический код управляет динамическим. Статический код записан в исходных файлах СПО. Динамический – формируется в процессе работы программы. Для этого и нужны генераторы кода в компилируемых библиотеках. Реализуется при помощи метафункций и представлений метаинформации.

Достоинства:

- более гибкая структура программы и более оптимизированный код;
- меньший объём исходного кода СПО;
- возможность повторного использования некоторых частей кода.

Недостатки:

- отсутствие четких правил создания метафункций и представлений метаинформации для СПО;
- как следствие предыдущего недостатка – менее понятный исходный код программы, структура которого изменяется от программиста к программисту;
- высокая сложность сопровождения СПО;
- зависимость генерируемого кода от видения программистом предметной области и методов их реализации;
- необходимость повторного создания метафункций и представлений метаинформации для разных предметных областей.

Аспектно-ориентированное программирование (АОП) – методика синтеза СПО, аналогичная GP, целью которой является разбиение задач группы предметных областей на ряд функциональных компонентов и аспектов (частей функциональных компонентов), из которых затем будет собираться конкретный проект СПО [4, с. 245]. Понятие "аспект" пришло в программирование из методов инженерии предметной области, которые понадобятся к рассмотрению ниже. Аспектом принято считать проекцию некоторого понятия, которую необходимо составлять с использованием других понятий, относящихся к данной же предметной области. Некоторые участки одних моделей пересекаются с участками других моделей [4, с. 135].

Достоинства:

- меньшая зависимость от предметной области, чем у других способов реализации;
- разработка СПО данным образом позволяет разрабатывать обобщённые повторно используемые элементы дополнительных библиотек, с помощью которых будут собираться продукты СПО АРМ ИС;
- возможность повторного использования кода;
- хорошая приспособленность к ООЯП;
- сокращение спутывания кода при реализации.

Недостатки:

- отсутствие чётких правил формирования оптимального набора задач автоматизации, для которого будет формироваться набор аспектов;
- избыточность в реализации, которая увеличивается с ростом числа аспектов, и вместе с тем, поощрение как можно большего числа аспектов при разработке данного конкретного программного продукта;
- связанные с этим большой объём и насыщенность информации о частях кода, из которых предполагается строить СПО семейства прикладных областей;
- необходимость доработки существующих ООЯП до аспектных языков.

Субъектно-ориентированное программирование – методика синтеза СПО, классифицируемая как одна из методик аспектно-ориентированной декомпозиции, целью которой является построение индивидуальных пользовательских проекций моделируемых объектов семейства предметных областей [4, с. 245]. Пользовательская проекция на компонент ПО в конкретном проекте СПО – это коллекция классов или их фрагментов (субъект), которая однозначно идентифицирует данный экземпляр СПО для данного АРМ. Субъектом может быть полная или частичная объектная модель какой-то части или всей предметной области.

Достоинства:

- возможность повторного использования кода;
- хорошая приспособленность к ООЯП;
- полное или частичное соответствие поведения реализованного СПО поведению использующего его субъекта предметной области;
- введение правил сочетания, позволяющих группировать субъекты в семейства и определять их поведение и свойства;
- введение правил соответствия, позволяющих единообразно описывать свойства и поведение различных классов субъектов со сходной функциональностью.

Недостатки:

- отсутствие чётких правил формирования оптимального набора задач автоматизации, для которого будет формироваться набор субъектов;
- избыточность в реализации кода;
- невозможность реализации динамической изменчивости субъектов;
- хорошая реализация только простых приложений, так как увеличение сложности поведения объектов и субъектов ведет к необходимости их переопределения, что в технической литературе принято называть проблемой разделения исходно-единичного объекта или объектной шизофренией [7].

Предметно-ориентированные языки программирования (ПОЯП) – [4, с. 146] – методика синтеза СПО, аналогичная порождающему программированию, целью которой является решение задач определённой предметной области. В какой-то степени этому определению удовлетворяет приведённое ниже описание языка и системы программирования и создания АРМ на базе продуктов компании 1С. Такие языки делят на несколько типов по типу представления информации (текстовые и графические) и технологии реализации (фиксирован-

ные автономные, встроенные и модульные). Примером текстового встроенного или фиксированного автономного ПОЯП является SQL. Другим примером ПОЯП является RISLA [8] – язык описания финансовых продуктов.

Достоинства:

- простота в наращивании характеристик;
- понятное описание предметной области и удобство манипулирования им в рамках конкретного АРМ (задачи автоматизации);
- возможность многократного использования некоторых частей кода (в модульных языках) в связанных предметных областях.

Недостатки:

- формирование "изолированных технологических участков", неспособных к взаимодействию с другими технологиями;
- трудности взаимодействия реализаций разных версий этих языков с другими ПОЯП;
- необходимость наличия общей платформы с инфраструктурой, обеспечивающей взаимодействие модульных ПОЯП (реализацией такой платформы по замыслу Microsoft должно стать IP).

Ментальное программирование (Intentional programming – IP) – методика синтеза СПО ("новаторская расширяемая среда программирования и метапрограммирования на основе активного кода"), которая позволяет прикладным программистам с помощью разработанных ими библиотек внедрять новые предметно-ориентированные расширения ЯП. [4, с. 459]. Intention – намерение, языковая абстракция, которая может быть как текстовым оператором, так и графическим объектом. Намерения (вместо операторов) образуют исходный код в IP. Библиотеки намерений выступают в качестве основных расширений в ООЯП. Методика разрабатывается с 1991 года и пока до конца не завершена и не опробована, но в настоящее время ей прочат большое будущее. По замыслу авторов-разработчиков IP должна создавать платформу взаимодействия различных ПОЯП. Планируется также, что данная методика заменит разработку с помощью большинства существующих и использующихся в настоящее время ЯП.

Достоинства:

- активный исходный код – расширяемая графическая структура, абстрактно-синтаксическое дерево – характеризующийся собственным поведением во время написания программ;
- работа с графом исходного кода в диалоговом режиме (режиме студии);
- возможность повторного использования кода и организации его в эффективные библиотеки;
- предметно-ориентированные нотации могут реализовывать собственные методы оптимизации, зачастую более эффективны, чем предыдущие попытки реализации данных частей кода в ООЯП;
- исчезновение синтаксического разбора исходного текста программ в связи с появлением графической нотации;
- упрощение рефакторинга – проверки соответствия заданным начальным характеристикам – в процессе написания исходного кода.

Недостатки:

- резкое увеличение числа библиотек и намерений при разработке крупных (более 20 АРМ) АСУП и как следствие затруднение разработки;
- данная методика пока недостаточно исследована при одновременной разработке нескольких АСУП одной организацией-разработчиком и её доработка продолжается;
- существует зависимость части намерений от предметной области, а при возрастании их числа число намерений растет в геометрической прогрессии, что усложняет код и сводит на нет все его преимущества;
- более высокая стоимость СПО по сравнению с другими методиками программирования.

Расширяемый язык разметки XML (eXtensible Markup Language) – [4, с. 146], [9, с. 620, 642] – методика синтеза СПО, целью которой является создание стандартизированного гибкого способа описания содержимого документов, размещаемых в Интернет. Позволяет описывать "представления данных" в БД и работать с ними. Возник как продукт развития класса языков разметки. Так как XML разрабатывался как серия стандартов, на нем основаны многие подходы к реализации, сформулированные как стандарты. Вот некоторые из них:

- простой протокол доступа к объектам (SOAP – simple object access protocol);
- XML Schema (задающий требования к структуре документов язык);
- XLST (программа применения таблицы стилей к XML-документу);
- DOM (Document Object Model – объектная модель документов – интерфейс прикладных программ, представляющий XML-документ в виде дерева);
- XPath (подъязык XLST, используемый для идентификации частей XML-документа, подлежащих преобразованию);
- XQuery (стандарт для представления запросов к БД XML-документов).

Помимо этого для ряда отраслей промышленности разрабатываются свои уникальные стандарты обмена XML-документами. В [9, с. 620] их указано около 30.

Достоинства:

- "универсальный язык эпохи Интернета" (Билл Гейтс);
- простая расширяемая спецификация;
- возможность создания динамических веб-страниц;
- возможность загрузки в и выгрузки из БД данных, описания БД сложной (нереляционной) структуры;
- возможность преобразования в другие форматы;
- возможность создания запросов с несколькими многозначными маршрутами.

Недостатки:

- большое количество стандартов разных отраслей;

- обозначение одинаковыми названиями разных понятий в разных вариантах стандартов;

- зависимость XML-документа и стандартов от предметной области;

- разработка и сопровождение крупных (свыше 20 АРМ) разнородных (с разными предметными областями) приложений требует значительных усилий;

- уникальность проектов построения СПО с помощью данных средств для каждой конкретной предметной области, и, в связи с этим, отсутствие возможности повторного использования его компонент в других проектах.

Несмотря на активное развитие технологий программирования и платформ синтеза АСУП, основные затраты времени и людских ресурсов (до 70 %, [4]), тратятся на отладку (доводку СПО). Поэтому недостатки, сформулированные для каждого из перечисленных направлений прикладного программирования можно отнести к процессам синтеза СПО в целом.

Обилие методик и средств разработки не гарантирует коллективам программистов и заказчикам возможности организовать синтез СПО для крупных АСУП аналогично серийному производству. Поэтому синтез СПО описанными традиционными методами требует значительных трудозатрат большого числа высококвалифицированных специалистов и не позволяет в последствии дорабатывать синтезированное ПО без особых проблем. В том числе и это обстоятельство порождает проблему "больших проектов", сутью которой является отсутствие возможности сдачи проекта по синтезу СПО для крупной АСУП в срок даже при наличии значительного финансирования и большого числа выполняющих работу программистов. Решение данной проблемы при помощи рассмотренных в статье методик программирования представляет собой актуальную научно-техническую задачу, решение которой будет иметь большое государственное значение.

Литература

1. ***Басс Л., Клементс П., Кацман Р.*** Архитектура программного обеспечения на практике, пер. с англ. СПб.: Питер, 2006.

2. ***Зыков С.В.*** Основы современного программирования. Разработка гетерогенных систем в Интернет-ориентированной среде (на основании курса лекций по информационным системам на факультете информационной безопасности МИФИ): учеб пос для вузов. М.: Горячая линия – Телеком, 2006.

3. ***Себеста Р.У.*** Основные концепции языков программирования, пер. с англ. М.: изд. дом "Вильямс", 2001.

4. ***Чарнецки К., Айзенкер У.*** Порождающее программирование. Методы, инструменты, применение, пер. с англ. СПб.: Питер, 2005.

5. ***Таненбаум Э., М. ван Стен.*** Распределённые системы, принципы и парадигмы // Сер. "Классика компьютерной науки", пер. с англ. СПб.: Питер, 2003.

6. ***Избачков Ю., Петров В.*** Информационные системы: учеб. для вузов. СПб.: Питер, 2005.

7. ***Subject-oriented*** programming and design patterns Draft. IBM Thomas J. Watson Research Center, Yorktown Heights, NY. http://www.research.ibm.com/sop/sopcpats.htm.

8. ***A. van Deursen, Klint P.*** Little Languages: Little Maintenance? // In Journal of Software Maintenance. No. 10. 1998. Pp. 75-92. http://www.cwi.nl/~arie.

9. ***Крёнке Д.*** Теория и практика построения баз данных, пер. с англ., СПб, Питер, 2005.