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

ДОСТОИНСТВА И НЕДОСТАТКИ СОВРЕМЕННЫХ МЕТОДОВ СИНТЕЗА СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (часть 1)

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 1)

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.

Для синтеза специального программного обеспечения (СПО) любых автоматизированных систем управления предприятиями (АСУП), используется большой класс инструментальных средств разработки ПО: систем программирования (СП) и языков программирования (ЯП). Сам по себе синтез СПО, как процесс, есть подпроцесс в разработке АСУП. Его анализ, с точки зрения использования для синтеза СПО различных методов и методик программирования как рекомендаций по использованию различных СП, методологий, технологий или ЯП, в технической литературе встречается в виде подробного обзора некоторых отдельных аспектов их применения.

В настоящей статье автор попытался хотя бы частично обобщить и кратко описать методики создания СПО, используемые в настоящее время. Ввиду объёмности литературы по данным вопросам, в статье приводятся лишь определения методов, наиболее значимые их характеристики, отличающие их от других методов, а также их достоинства и недостатки, касающиеся разработки СПО в АСУП.

Приводимые в статье методики составляют в настоящее время основу исторически сложившейся (традиционной), но при этом не исследованной до конца методологии синтеза СПО вообще. Следует, однако, помнить, что для АСУП крупных предприятий синтез СПО даже с их помощью представляет собой чрезвычайно трудную и нетривиальную задачу [1]. Кроме того, технологии и способы синтеза СПО непрерывно развиваются.

Полный список технологий и способов реализации СПО, их критический анализ и сравнение – тема для серьезного научного исследования, размер которого превышает несколько десятков тысяч страниц. Поэтому в данной статье будут приведены некоторые важные направления и подходы к технологиям синтеза СПО. Низкоуровневые ЯП, языки программирования высокого уровня (ЯВУ) второго поколения (процедурные, декларативные и логического программирования по классификации, приведенной в [2], с. 8-15), а также языки сценариев в данном обзоре не рассматриваются. Подробнее данный анализ по перечисленным направлениям приведен в [2, 3].

Программная инженерия (ПрИ) – это совокупность инженерных методов и средств создания ПО, а также дисциплина, изучающая применение строгого количественного подхода к синтезу, эксплуатации и сопровождению ПО. С точки зрения ПрИ, проектирование ПО является формальным процессом [4].

Достоинства:

- более высокое качество проектирования СПО;
- управляемость процесса разработки;
- увеличение срока жизни СПО и возможность его доработки.

Недостатки:

- отсутствие методик эффективного управления процессами проектирования и реализации при создании СПО;
- при доработке СПО в процессе сопровождения возрастает его энтропия, что приводит к тому, что СПО становится все более хаотичным по структуре и функционированию и сильно отличается от первоначального проекта (при создании СПО АСУП с жизненным циклом 10-15 лет только первые 3-4 года тратятся на разработку, а остальные затраты времени – на сопровождение);
- проект СПО, реализуемый с помощью ПрИ, сильно зависит от предметной области и зачастую не может быть использован повторно в другой предметной области даже сразу после разработки, а тем более после нескольких лет сопровождения.

Быстрая разработка ПО (БР ПО) – основанный на 4 базовых идеях подход к созданию СПО, позволяющий вести синтез СПО довольно быстро (в сравнении с традиционными способами разработки). Эти идеи формулируют отказ от ряда правил документирования процессов общения разработчиков с пользователями, синтеза СПО и изменений в планах его разработки. За счёт этого объём работ при синтезе СПО сокращается [6].

Подход – это не технология. Поэтому его воплощение требует реализации в условиях конкретной технологии. Одним из примеров такой конкретизации стало ***экстремальное программирование*** (Extreme Programming – XP) [5]. Его отличают следующие особенности:

- одновременная работа над проектом от 3 до 10 программистов;
- разработка кода трехнедельными итерациями по частям группами по 2 программиста на одну часть СПО;
- собранный таким образом код может использоваться заказчиками;

- единицей требований к СПО на стадии анализа требований становится "история пользователя", описывающая функциональность пользователя, которая может быть разработана за одну итерацию (обозначается карточкой);

- один из программистов при этом назначается главным, он проводит короткие ежедневные утренние совещания, на которых может менять местами программистов, выполняющих задание по одной из карточек;

- благодаря возможности такой замены исходный код как всей системы, так и её частей поддерживается в максимально простом состоянии;

- заказчики принимают участие в совещаниях и пишут тесты для проверки частей и всей системы СПО.

Достоинства:

- снижение риска ошибок описания предметной области за счет постоянного общения заказчиков с разработчиками;

- высокая дисциплина при написании исходного кода и его простота, а в случаях, связанных с неспецифическими его доработками, полная информированность о них всей команды разработчиков;

- высокая скорость синтеза СПО и его внедрения.

Недостатки:

- XP предназначено для небольших команд разработчиков и применимо лишь для АСУП с числом разрабатываемых АРМ не более 20;

- сбор данных о предметной области на карточках приводит к большому разрастанию хранимого не относящегося к документации материала;

- опыт команды по разработке в интересах АСУП не может быть использован другими командами разработчиков, так как они, как творческие работники, разрабатывают свои методы построения исходного кода;

- для каждой конкретной предметной области (а их в АСУП несколько тысяч) и для каждого нового ЯВУ требуется свой подход к реализации, благодаря чему использованный код СПО для создания новых АСУП может быть использован не более, чем на 10 %.

Структурное проектирование ПО (СП ПО) – методика, по которой детально описывается функционирование системы и строится концептуальная модель её БД. Результатами проектирования при этом являются:

- модель системных процессов;

- архитектура системы;

- модели данных приложений;

- модель пользовательского интерфейса.

Методической основой данной методики является процессный подход к моделированию бизнес-процессов. В процессе работы над бизнес-моделью системы определяются сущности и их взаимодействие между собой. Не вдаваясь в подробности описанного метода синтеза СПО, перейдем сразу к анализу его достоинств и недостатков.

Достоинства:

- создание модели АРМ типа "сущность-связь";
- определение структуры БД;
- определение структуры меню приложений;
- определение распределения процессов в системе по уровням;

определение последовательности экранных форм.

Недостатки:

- сильная зависимость структуры и реализации характеристик создаваемого экземпляра СПО АРМ от предметной области;
- отсутствие поддержки программирования повторно используемых компонент;
- отрицание необходимости и, в связи с этим, невозможность повторного использования компонент данного экземпляра СПО;
- использование для создания небольших проектов СПО и низкая эффективность при реализации крупных АСУП.

Rational Unified Process (RUP) – методика синтеза СПО, целью которой является спецификация требований к СПО [4, с. 260]. Требования к СПО делят на функциональные, связанные с предметной областью, и нефункциональные, связанные со средой реализации: ОС, СУБД, ЯВУ. Для выявления требований к СПО используют:

- собеседования;
- анкетирование;
- моделирование и анализ бизнес-процессов;
- собрания специалистов, анализирующих результаты описанных выше трех действий;
- создание прототипов СПО и их демонстрация пользователям.

Достоинства:

- формирование концепции проекта, зафиксированного документально как соглашения между заказчиками и разработчиками СПО;
- получение словаря предметной области и описание в этих терминах моделей и требований;
- описание функциональных требований к СПО как вариантов его использования (use case).

Недостатки:

- сильная зависимость структуры и реализации характеристик создаваемого экземпляра СПО АРМ от предметной области;
- уникальность проектов построения СПО с помощью данных средств и, в связи с этим, отсутствие возможности повторного использования его компонент в других проектах;
- отсутствие средств моделирования изменчивости, этапа определения предметной области и различий между моделированием отдельного проекта и семейства таких проектов [7, с. 74-75];
- разработка и сопровождение крупных (свыше 20 АРМ) приложений требует значительных усилий, так как увеличивает число используемых компонент в десятки раз, по сравнению с простыми (1-2 АРМ) приложениями.

Объектно-ориентированный анализ ПО (ООА ПО) – методика синтеза СПО, целью которой является трансформация функциональных требований к СПО в предварительный системный проект и создание стабильной архитектуры системы [4, с. 291]. Методической основой данной методики является технология RUP, описанная выше. Сама методика включает в себя два вида анализа: архитектурный и вариантов использования. Архитектурный анализ необходим для определения стандартов моделирования и документирования, архитектуры системы (сущностей, связей, уровней классов) и начального представления предметной области. Анализ вариантов использования необходим для идентификации классов, исходя из представления предметной области, определения их поведения (методов классов), атрибутов и ассоциаций, а также их унификации.

Достоинства:

- удобная и простая трансформация требований к СПО в предварительный его проект;
- создание программной архитектуры СПО (уровни пользователей и программ и взаимодействие между ними): формирование списка основных абстракций предметной области и начального представления о её уровнях;
- определение списка классов СПО, их поведения и атрибутов и ассоциаций;
- унификация классов и их связывание с компонентами ЯВУ.

Недостатки:

- сильная зависимость проекта СПО от предметной области и ЯВУ;
- отсутствие поддержки программирования повторно используемых компонент;
- ограниченные возможности по повторному использованию компонент программ и классов СПО в других проектах из-за связи с предметной областью и ЯВУ.

Объектно-ориентированное проектирование ПО (ООП ПО) – методика синтеза СПО, целью которой является адаптация набора классов "анализа" архитектуры системы к среде реализации (ЯВУ) с учётом нефункциональных требований [4, с. 317]. Данная методика стала развитием ООА ПО, которое включает в себя два вида проектирования: архитектуры системы и элементов системы. Процесс создания архитектуры системы включает в себя действия:

- идентификация (определение и именование) архитектурных решений (инструментальных средств и заложенных в них реализаций), необходимых для реализации (создания проекта СПО);
- анализ методов взаимодействия классов, их иерархии и интерфейсов (в данном случае речь идет об интерфейсах классов – наборе переменных и программ, с помощью которых можно с ним работать, – в отличие от интерфейса пользователя СПО АРМ);

- формирование в зависимости от сложности предметной области (и взгляда на реализацию с помощью данного ЯВУ данной конкретной задачи программистом-разработчиком) уровней и подуровней классов, реализующих данный конкретный проект СПО;

- проектирование структуры потоков управления;

- проектирования конфигурации системы.

Процесс проектирования элементов системы включает в себя [4, с. 333]:

- уточнение описания вариантов использования;

- проектирование классов на ЯВУ;

- проектирование баз данных, зависящее от типа используемой в данном проекте СУБД.

Достоинства:

- описание СПО АРМ на стадии анализа требований в терминах объектно-ориентированных ЯП (ООЯП);

- при использовании ООЯП и объектно-ориентированных БД (ООБД) прямое преобразование данных, полученных на стадии анализа требований, в программный код.

Недостатки:

- зависимость классов СПО от структуры и вида предметной области;

- зависимость состава и структуры СПО (состава, иерархии, атрибутов и ассоциаций классов) от видения программистом предметной области и методов их реализации в ООЯП;

- отсутствие поддержки программирования повторно используемых компонент;

- библиотеки компонентов и объектов содержат миллионы элементов со всевозможными комбинациями свойств;

- большое число диаграмм (иерархии классов, ассоциаций классов и др.) для конкретного экземпляра СПО АРМ, что обуславливает большой объём документации в рамках одного проекта.

Программирование в CASE-системах – методика синтеза СПО, целью которой является использование специально созданного для описания СПО инструментального программного средства, предназначенного для поддержки процесса проектирования АСУП [8]. Наиболее известны сейчас следующие из них: ER-win, BP-win, Design/IDEF [9], SILVERRUN [10]. Кроме того широко используются средства UIMS (User Interface Management System) систем управления пользовательского интерфейса, которые также принято относить сейчас к CASE-средствам или к СУД. К ним относятся OSF/Motif, LinkWorks, Documentum`s EDMS или Enterprise Document Management System и другие продукты компании Documentum, LiveLink Intranet, Flucrum Knowledge Network, Verity Topic, а также промышленные СУД компании Excalibur Technologies Corporation, использующие в своей работе при поиске документов нечеткие логики с применением технологии адаптивного распознавания образцов Adaptive Pattern Recognition Processing (APRG). Объектно-ориентированные CASE-средства используются для создания классов объектов. Некоторые про-

изводители современных средств программирования встраивают объектно-ориентированные CASE-средства в рабочую среду своих систем, что упрощает последующий синтез СПО [8, с 124]. К таким средам можно отнести, например, продукт Simantec C++ версии 7.2 или Class Expert.

CASE-системы в основном оперируют моделями, построенными по принципу "сущность-связь" или ER-моделями [9, 10]. Модели, построенные по данному принципу, позволяют документировать проект и одновременно описывать базу данных.

Достоинства:

- распараллеливание работ по внесению изменений в информацию пользователя;
- упрощение сопровождения конкретной части ИС, связанной с отдельным пользователем;
- естественность описания информации для сопровождающего и разрабатывающего ИСШ персонала;
- объектно-ориентированные CASE-системы существенно снижают время разработки и позволяют оформлять клиентские рабочие места с помощью генераторов кода на ОО ЯП;
- набор возможностей CASE-системы помогает осуществлять достаточно быструю настройку системы на нужную информацию, причем разработчик АСУП может сосредоточиться в основном на информации предметной области и не разрабатывать СПО в традиционном понимании;
- использование в качестве средств поддержки клиентской части АСУП известных средств программирования SQLWindows, PowerBilder, Visual C++, Delphi, Optima+, JAM;
- легкое построение концептуальной схемы БД.

Недостатки:

- большинство существующих CASE-средств ориентированы, в основном, на программиста, что не позволяет пользователю принять участие в разработке АСУП;
- высокая зависимость создаваемого проекта от предметной области;
- необходимы значительные усилия групп программистов при сопровождении уже созданной АСУП, так как при изменении требований к предметной области невозможно быстрое изменение диаграмм функциональной декомпозиции и других элементов системы;
- разработка и сопровождение крупных (свыше 20 АРМ) разнородных (с разными предметными областями) приложений требует значительных усилий и значительных затрат, которые часто невозможно окупить;
- динамическое связывание, предполагающее поиск метода в классе, что приводит к общему замедлению времени поиска метода в классе по сравнению с программой на традиционном ЯП в 1,75-2,5 раза, иными словами возрастает время поиска конкретной информации;

- многочисленность методов и их вызовов, или иными словами для получения информации по (в нашем случае) факту более высокого уровня система совершит каскад вызовов для фактов, на основании которых определен данный факт;

- при работе АСУП на базе объектно-ориентированной CASE-системы на компьютерах с сегментной организацией памяти происходит замедление работы, так как каждый класс объявляется отдельным файлом, что приводит к интенсивному межсегментному обмену и увеличению времени обработки запроса в десятки и сотни раз, так как число объектов (фактов) в реальной АСУП исчисляется десятками тысяч в день;

- подавляющее большинство CASE-средств не имеют выразительных возможностей, позволяющих описать и промоделировать динамику процессов;

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

Дальнейшее развитие CASE-средств в приложении их к конкретным условиям привело к созданию новой области человеческой деятельности, связанной с автоматизацией. Она получила название реинжиниринга.

Проведение реинжиниринга СПО и предметной области – методика синтеза СПО, целью которой является проведение "обратного" анализа автоматизации (того насколько эффективно это сделано) и деятельности больших подразделений крупной компании (организации) со сложной внутренней структурой, которая позволяет пересмотреть многие аспекты построения внутренней деятельности и внутреннего документооборота, а также связей с другими организациями. В некоторых случаях это помогает существенно (в десятки и сотни раз) увеличить эффективность работы подразделений, подвергшихся данной процедуре. Основную роль в этом процессе играют INTERNET, электронный документооборот, а также средства поддержки гипертекстовых документов.

Достоинства:

- построение эффективных внешней и внутренней моделей деятельности каждого из больших подразделений и системы в целом;

- достижение существенной экономии средств заказчиков в результате построения данных моделей.

Недостатки:

- высокая стоимость средств реинжиниринга и услуг по их предоставлению;

- зависимость построенной модели от предметной области;

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

Следует заметить, что исходный код в СПО является концентрированным выражением как усилий программиста (представителя организации-разработчика), так и передаваемых ему в качестве ТЗ знаний о работе конкретного исполнителя (представителя заказчика) в той её части, в которой реализуется данный экземпляр СПО. Также следует заметить, что, несмотря на активное развитие технологий программирования и платформ синтеза АСУП, основные затраты времени и людских ресурсов (до 70 %, [4]), тратятся на отладку (доводку СПО), в какие бы части цикла разработки они ни попадали. Поэтому перечисленные недостатки можно отнести не только к методикам, но и к процессам синтеза СПО в целом.

Таким образом, обилие методик и средств разработки не гарантирует коллективам программистов и заказчикам возможности организовать синтез СПО для крупных АСУП аналогично серийному производству. Поэтому синтез СПО при помощи описанных в статье СП, СУБД и ЯВУ требует значительных трудозатрат большого числа высококвалифицированных специалистов и не позволяет впоследствии дорабатывать синтезированное ПО без особых проблем. Кроме того, использование описанных в статье средств автоматизации предприятий позволяет обеспечивать преемственность только незначительной части синтезируемого СПО.

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. ***Вендров А.М.*** Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2005.

5. ***Бек К.*** Экстремальное программирование. СПб.: Питер, 2002.

6. ***Коберн А.*** Быстрая разработка программного обеспечения, пер. с англ. М.: ЛОРИ, 2002.

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

8. ***Ражев А.В., Ксенофонтов С.Л.*** Применение CASE-технологий для разработки сложных информационных систем // Автоматизация и компьютеризация информационной техники и технологии: сборник научных трудов. Вып. 282. М. изд-во МГУЛ, 1998. С. 120-124.

9. ***The U.S. Air Force*** IDEF Methods, A structured approach to modeling and analysis, http://www.idef.com.

10. ***Методология*** структурного анализа и проектирования, пер. с англ. М.: Мир, 1993.