УДК 004.413:338.5

РЕИНЖЕНЕРИЯ ПРОЕКТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Н.А. Сидоров, Е.А. Авраменко, В.А. Хоменко

Национальный авиационный университет
03058, Киев, проспект космонавта Комарова, 1,
тел.: 406 7396, факс: (044) 497 8106; e-mail: sna@nau.edu.ua

Рассматриваются вопросы восстановления и переработки документации проектов программного обеспечения, в том числе с целью обеспечения соответствия объектно-ориентированной парадигме и универсальному процессу проектирования RUP. Приводится практический пример проведения таких работ для одного из проектов программного обеспечения.

The questions of the software projects documentation restoring and conversion including object-oriented paradigm and rational universal process requirements support are presented. The practical case study of the similar work is giver.

Введение

Проекты программного обеспечения (ПО) требуют создания и ведения проектной документации [1]. Особенно это касается больших и долгоживущих проектов, полное и качественное документирование которых является необходимым условием их управляемости на протяжении всего жизненного цикла ПО. Однако на практике зачастую создается только та проектная документация, которую требует заказчик ПО. Причина этого понятна – дополнительное документирование требует дополнительных затрат. Увеличение числа крупных проектов и систематически сопровождаемого ПО определяет повышение интереса отечественных разработчиков к задаче обеспечения качества документирования проектов. Способом решения указанной задачи является реинженерия ПО [2]. Статья состоит из трех частей. В первой части рассматриваются вопросы документирования проектов ПО; во второй – подходы к реинженерии проектов, обеспечивающей восстановление и переработку документации; в третьей – пример реализации реинженерии.

1. Документирование проектов ПО

К причинам, побуждающим разработчиков документировать проекты, можно отнести следующие:

- стремление обеспечить эффективное взаимодействие исполнителей проекта;
- желание детальное специфицировать разрабатываемое ПО большой сложности;
- требование снизить расходы на сопровождение ПО;
- желание обеспечить максимальную взаимозаменяемость исполнителей проекта;
- требования заказчика к объему и качеству документации;

стремление сертифицировать процессы разработки ПО на соответствие определенным стандартам (например, SEI/CMMI) [3].

Спецификации ПО можно разделить на три вида (в соответствии с этапами жизненного цикла) [4] (рис.1): требований; проектные (архитектурные и детальные); реализации (исходные коды программ и структуры данных).

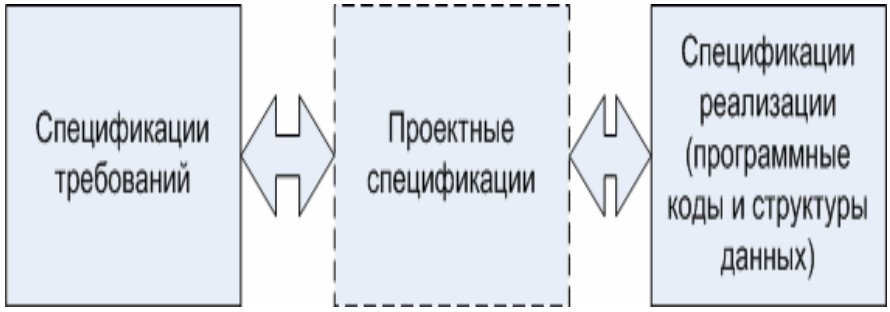

Рис. 1. Виды спецификаций ПО

Условия создания ПО вынуждают разработчиков обязательно реализовывать спецификации требований и реализации. Первые необходимы для работы с заказчиком, а вторые являются результатом генерации собственно ПО. Проектными спецификациями, зачастую, пренебрегают, принося их в жертву снижению трудозатрат затрат и времени на реализацию проекта. При этом в документации исчезает звено, которое необходимо для дальнейшей эффективной работы с артефактами проекта и продуктивного взаимодействия его

участников (рис.2). Отсутствие указанного звена также повышает расходы на сопровождение и повторное использование компонентов ПО.

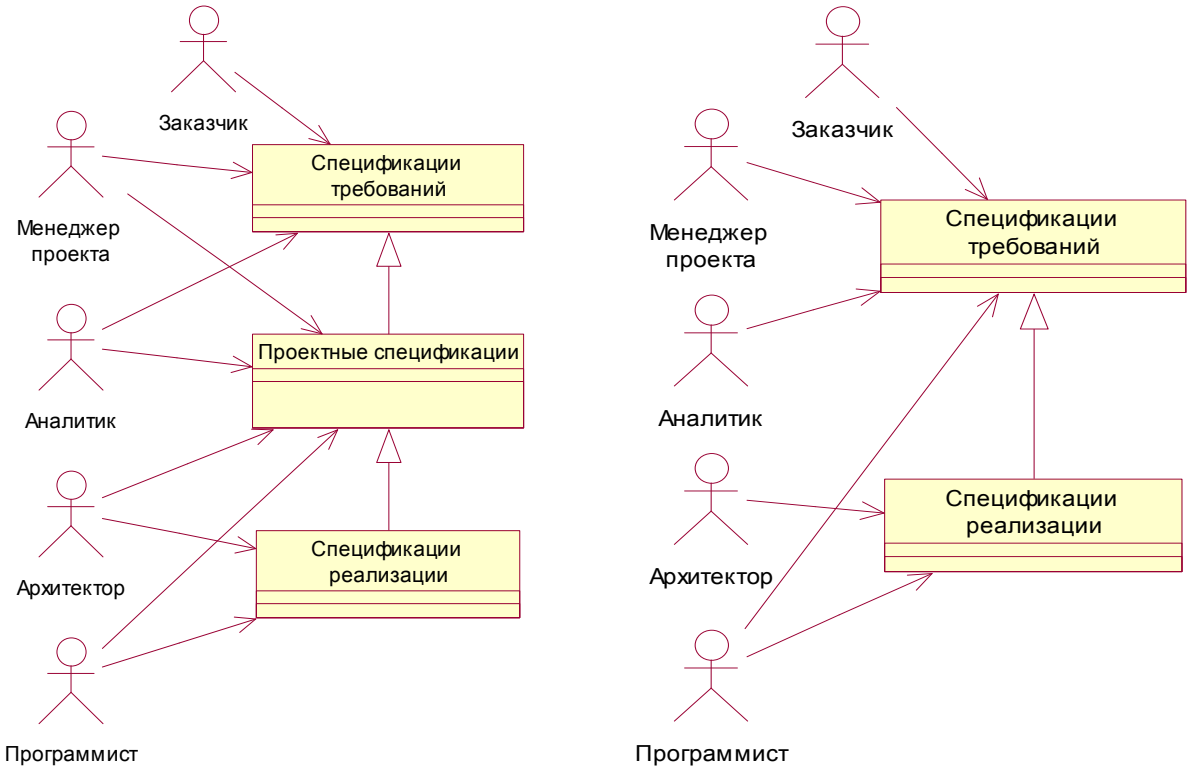

Рис. 2. Взаимодействие участников проекта: а – при наличии проектных спецификаций; б – при отсутствии проектных спецификаций

Негативные последствия такого отношения к документации начинают сказываться в процессе разработки и особенно остро проявляются на этапах сдачи ПО и его сопровождения. В результате проект приобретает следующие нежелательные свойства:

– ошибочность проектирования и реализации из-за отсутствия единого видения менеджерами и исполнителями проекта архитектуры и деталей ПО;

– затрудненность контроля выполнения детальных требований к проекту и ПО из-за их отсутствия или размытости;

– сложность интеграции эклектичных компонентов в единую систему из-за разнобоя в средствах, методах и стилях реализации ПО, применяемых программистами;

– слабая взаимозаменяемость исполнителей из-за выбывающих или перемещаемых на другой участок работы программистов, уносящих с собой общее видение и детали своей работы «в голове» и не оставляющих после себя документальных артефактов. Новый исполнитель, при отсутствии высокоуровневых спецификаций, должен тратить значительное время на изучение и анализ кодов, прежде чем начать активную работу;

– сложность сопровождения из-за необходимости даже при незначительных доработках ПО через достаточно большой промежуток времени затрачивать время и ресурсы на анализ и восстановление структуры ПО.

Очевидно, что убытки от «плохого» документирования проектов имеют тенденцию к росту при увеличении числа проектов, осуществляемых организацией, а также при увеличении их сложности и времени жизни.

Вышеперечисленные свойства, приобретаемые ПО, рано или поздно приводят руководителей к необходимости внедрения современной системы автоматизированной разработки ПО, которая, как правило, предусматривает полное документирования проектов. Внедрение такой системы требует внесения изменений в процесс разработки, роли и обязанности участников проектов. При наличии у разработчика сопровождаемых им проектов возникает отдельная проблема эффективного выполнения процессов сопровождения. При этом необходимым является восстановление недостающих спецификаций и других документов, предусмотренных внедряемыми стандартами документирования. Обычно, восстановления требуют в первую очередь «выпавшие» из процесса разработки проектные спецификации (рис. 1). Восстановление осуществляется с помощью обратной инженерии [2]. Это требует специальных работ по каждому сопровождаемому проекту, для эффективного проведения которых необходимо определить соответствующую методологию и средства.

2. Процессы восстановления и переработки спецификаций проекта

Изменение документации, в зависимости от исходного состояния проекта и новых требований к документированию, может затрагивать разные виды спецификаций проекта. На рис. 3 показана модель восстановления и переработки спецификаций проекта, в которой можно выделить процессы двух основных видов: редокументирование и обратная инженерия [2]. Редокументирование реализует изменения формы представления спецификаций определенного этапа разработки, без использования спецификаций других этапов. Обратная инженерия реализует восстановление или изменение (путем реализации процессов прямой инженерии) спецификаций определенного этапа разработки на основе спецификаций других этапов.

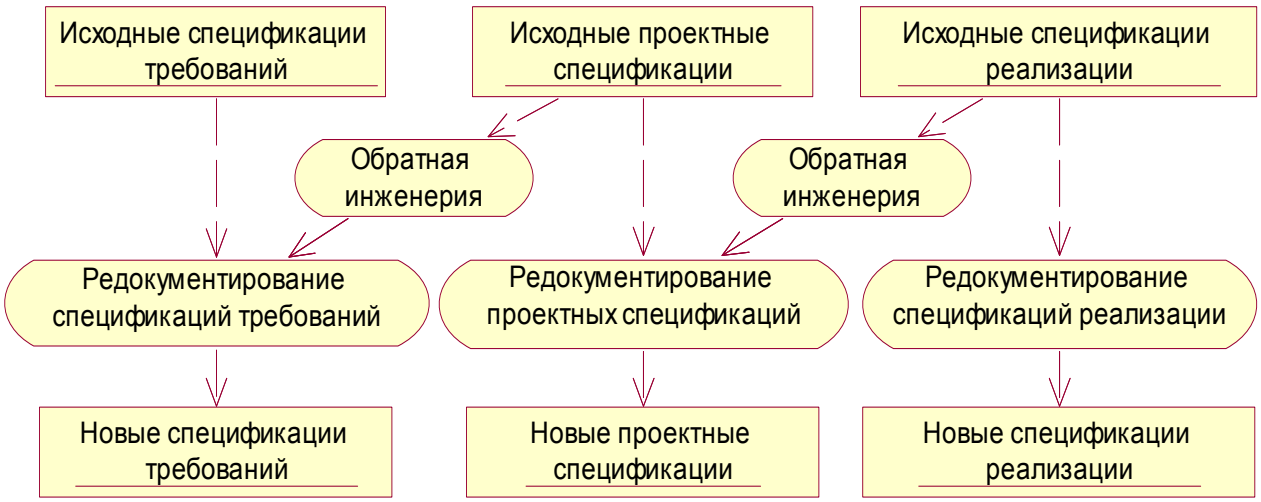

Рис. 3. Процессы восстановления и переработки документации

Ориентация на базовый процесс разработки

Артефакты, получаемые в результате проектирования ПО, создаются в рамках определенной системы (модели) проектирования. В ней можно выделить следующие основные элементы: парадигму проектирования, языки специфицирования, инструментальные средства проектирования и документирования.

Сейчас одним из наиболее совершенных и перспективных процессов разработки ПО-унифицированный процесс разработки (Rational Unified Process - RUP) [5]. В RUP проектирование основано на объектно-ориентированной парадигме, а для специфицирования проекта и его результатов используется язык Unified Modeling Language (UML) [6]. В настоящее время RUP поддерживается семейством продуктов IBM Rational [7].

Исходная документация

Состав и содержание исходной документации проекта определяется стандартами и требованиями, на которые ориентировался разработчик ПО. В Украине, традиционно, разработчик руководствуется национальными государственными и отраслевыми стандартами, при этом чаще всего состав документации соответствует стандарту, который предусматривает следующие виды документов [8]: спецификация; ведомость держателей подлинников; текст программы; описание программы; ведомость эксплуатационных документов; формуляр; описание применения; руководство системного программиста; руководство программиста; руководство оператора; описание языка; руководство по техническому обслуживанию; программа и методика испытаний; пояснительная записка.

Стандарт требует наличия двух обязательных документов – спецификации (для комплекса программ) и текста программы (для программного компонента) [8]. Перечень остальных документов определяется на этапе разработки и утверждения технического задания (ТЗ).

Документация RUP

В рамках RUP, при выполнении рабочих процессов разрабатывается ряд моделей, которые представляются с использованием языка UML. К основным относят следующие модели [5, 7]: требований; анализа, проектирования, развертывания, реализации, тестирования.

Основываясь на анализе фаз и рабочих процессов RUP можно распределить эти модели по видам спецификаций (рис.1) следующим образом [5]:

- требований – модель требований;
- проектные – модели анализа и проектирования;
- реализации – модели развертывания, реализации и тестирования.

Очевидно, что для эффективной переработки исходной документации проекта на соответствие моделям RUP необходимо провести предварительное сопоставление их содержания. В таблице приведено сопоставление моделей и артефактов RUP [5, 9] с видами исходных документов проектов ПО, определяемых стандартом [8]. Сопоставление проведено на основе анализа их назначения, семантики и содержания. Кроме того, в таблице приведены типы процессов, обеспечивающие переработку содержимого исходных документов, и исполнители, в качестве которых могут выступать эксперты или соответствующие CASE-инструменты [10].

Сопоставление моделей и артефактов RUP документам стандарта [8] Таблица

Модель рабочих процессов RUP	Артефакт RUP	Исходные документы [8]	Тип процесса	Исполнитель
Модель вариантов использования	Актанты	ТЗ, формуляр, описание применения, руководство программиста	Редокументирование	Эксперт
	Описание архитектуры (представление модели вариантов использования)			
	Глоссарий			
	Прототипы интерфейса пользователя	Руководство оператора, руководство программиста, текст программы	Редокументирование Обратная инженерия	Эксперт, CASE-инструмент
Модель анализа	Классы анализа	Руководство программиста, описание программы	Обратная инженерия	Эксперт
	Анализ реализации варианта использования	Руководство оператора	Редокументирование	Эксперт
	Пакеты анализа	Спецификация, описание программы	Обратная инженерия	Эксперт
	Описание архитектуры (представление модели анализа)	Спецификация, описание программы, руководство программиста	Обратная инженерия	Эксперт
Модель проектирования	Классы проектирования	Текст программы	Обратная инженерия	CASE-инструмент
	Проекты реализации вариантов использования	Руководство программ-миста, тексты програм-мы	Обратная инженерия	Эксперт
	Подсистемы проектирования	Описание программы, текст программы	Редокументирование Обратная инженерия	Эксперт, CASE-инструмент
	Описание архитектуры (представление модели проектирования)	Описание программы, текст программы	Редокументирование Обратная инженерия	Эксперт, CASE-инструмент
Модель развертывания	Описание архитектуры (представление модели развертывания)	Руководство истем-ного программиста, руководство по техническому обслуживанию	Обратная инженерия	Эксперт

Модель рабочих процессов RUP	Артефакт RUP	Исходные документы [8]	Тип процесса	Исполнитель
Модель реализации	Компонент	Описание программы, текст программы	Редокументирование	Эксперт
	Подсистемы реализации		Редокументирование	
	Интерфейсы			
	Описание архитектуры (представление модели реализации)	Исполняемый код		CASE-инструмент
	План сборки	Руководство системного программиста	Обратная инженерия	Эксперт
Модель тестирования	Тестовый пример	Методика испытаний, программа испытаний	Редокументирование	CASE-инструмент
	Тестовый компонент			
	Процедура тестирования			
	Процедура тестирования			
	Дефект			
	Оценка теста			

3. Пример реинженерии проекта ПО

Вышеизложенные положения были использованы при реализации задачи переработки проектных решений документации одной из подсистем ПО, входящего в состав автоматизированной системы изготовления и учета документов государственного образца. ПО проекта включает около восьмидесяти программных модулей, треть из которых содержат оригинальные исходные коды на языке Object Pascal общим объемом около 15000 строк. Кроме того, ПО включает базу данных (БД) в формате SQL-сервера Firebird, содержащую более 60 сущностей. Решение задачи было направлено на адаптацию документации и процессов сопровождения ПО к условиям RUP. Необходимость решения этой задачи была вызвана желанием снизить риски и расходы на сопровождение проекта, а также необходимостью привлечения иностранных инвесторов и сторонних разработчиков к сотрудничеству. Для работы были предоставлены следующие исходные материалы: ТЗ, постановка задачи, формуляр, руководство пользователя, руководство программиста, руководство системного программиста, руководство по техническому обслуживанию, физическая база данных, исходные коды программ. Предварительный анализ предоставленных документов выявил три принципиальные особенности проекта:

– отсутствие проектных спецификаций (проблема «потерянного звена»);

– ориентация документации проекта на требования стандарта [8], которые не согласуются с требованиями RUP;

– отсутствие объектно-ориентированной парадигмы в документации проекта.

Первая особенность устранялась путем применения обратной инженерии с использованием следующих CASE-инструментов:

– IBM Rational Rose – средство визуального моделирования;

– Ensemble Rose Delphi Link 3 – дополнительный программный модуль, предназначенный для интеграции моделей IBM Rational Rose со средой разработки Delphi;

– IBDataWorks – средство фирмы SoftMosis для моделирования БД под SQL-серверами InterBase и Firebird.

Устранение второй особенности производилось путем экспертного сопоставления требуемых артефактов моделей RUP с содержанием исходной документации и построением соответствующих моделей с применением инструмента IBM Rational Rose.

Устранение третьей особенности было частично реализовано с помощью средств визуального моделирования, работающих в контексте объектно-ориентированной парадигмы. При этом, было установлено, что предоставленное ПО создано с нарушением объектно-ориентированной парадигмы, в частности, уровень логики приложений был реализован на основе структурного подхода с нарушением принципов абстрагирования и инкапсуляции. Выявленные противоречия сделали невозможным создание на UML части

описаний реализации вариантов использования. Поскольку такая ситуация потенциально может отразиться на качестве процессов сопровождения и внесения изменений в ПО, а также эффективности повторного использования программных компонентов, заказчику было рекомендовано переработать код приложений для приведения его в соответствие с объектно-ориентированной парадигмой.

Далее приведены три примера реализации процессов восстановления и переработки спецификаций проекта.

Восстановление модели БД

На рис. 4 схема процесса восстановления модели БД основе исходной физической БД. В процессе реинженирии использовались два CASE-инструмента: IBDataWorks – для получения описания БД на языке Data Description Language (DDL) на основе предоставленной физической БД; Data Modeler (компонент IBM Rational Rose) – для получения на основе скриптов DDL объектного представления модели БД. Такой подход позволил в несколько раз сократить время восстановления модели БД и обеспечить ее интеграцию в общую модель ПО в формате IBM Rational Rose.

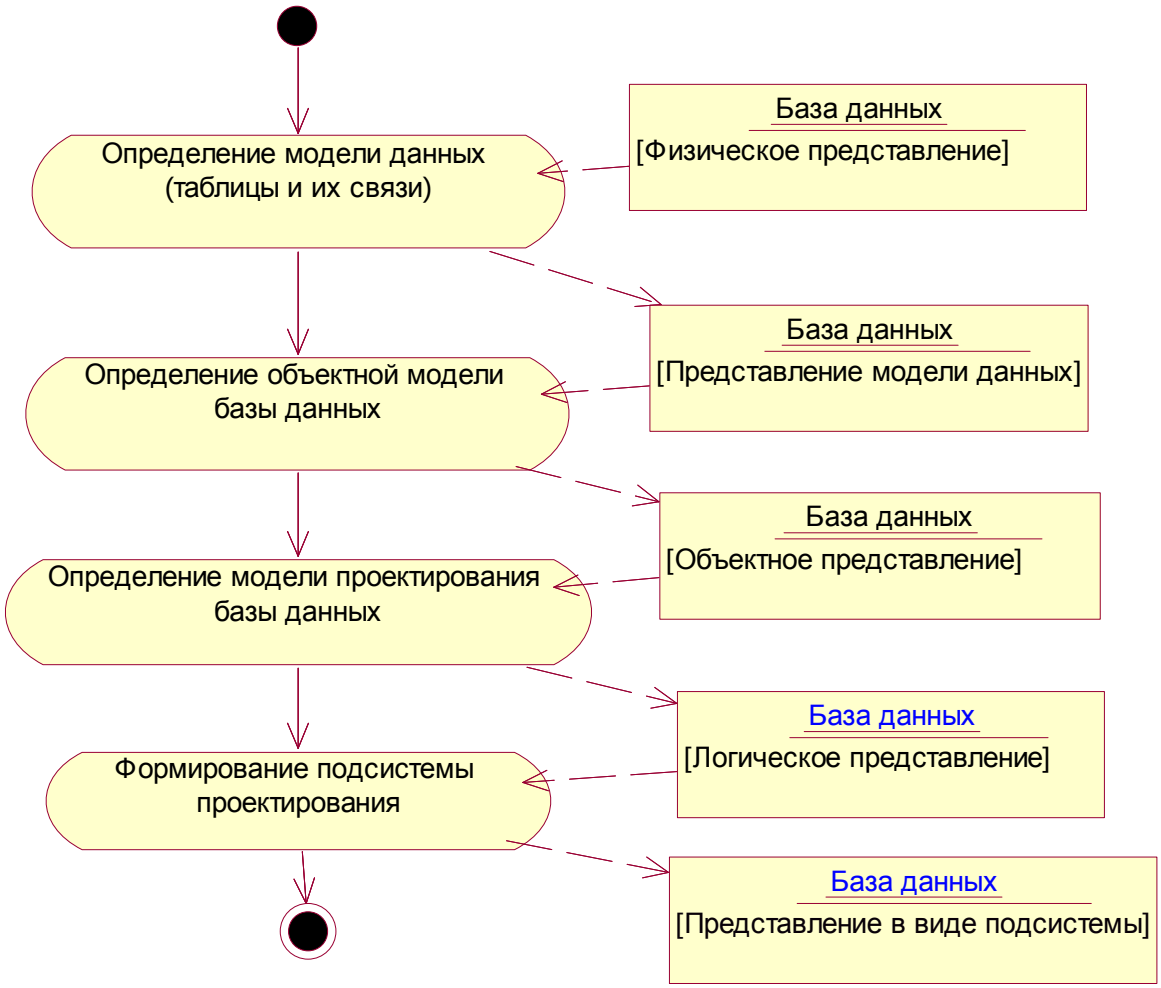

Рис. 4. Процесс восстановления моделей БД

Восстановление классов проектирования для реализации пользовательского интерфейса

Для восстановления спецификаций классов пользовательского интерфейса была применена обратная инженерия на основе исходных кодов с применением дополнительного программного модуля Ensemble Rose Delphi Link 3 для IBM Rational Rose. В результате, практически автоматически были получены и интегрированы в общую модель диаграммы классов всех представлений пользовательского интерфейса ПО. Внешний вид одной из форм интерфейса и соответствующая диаграмма классов, полученная после

восстановления, показаны на рис. 5 и 6 (часть атрибутов и операций библиотечных классов на рис. 6 не отражены).

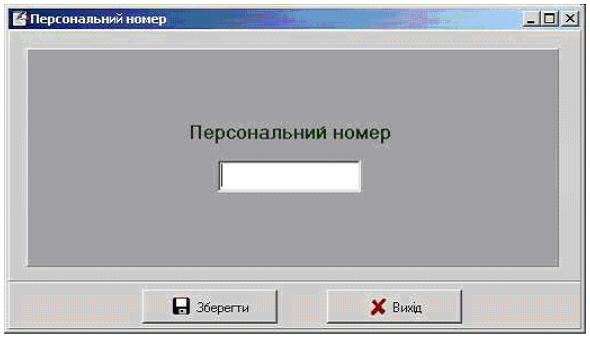

Рис. 5. Вид формы пользовательского интерфейса

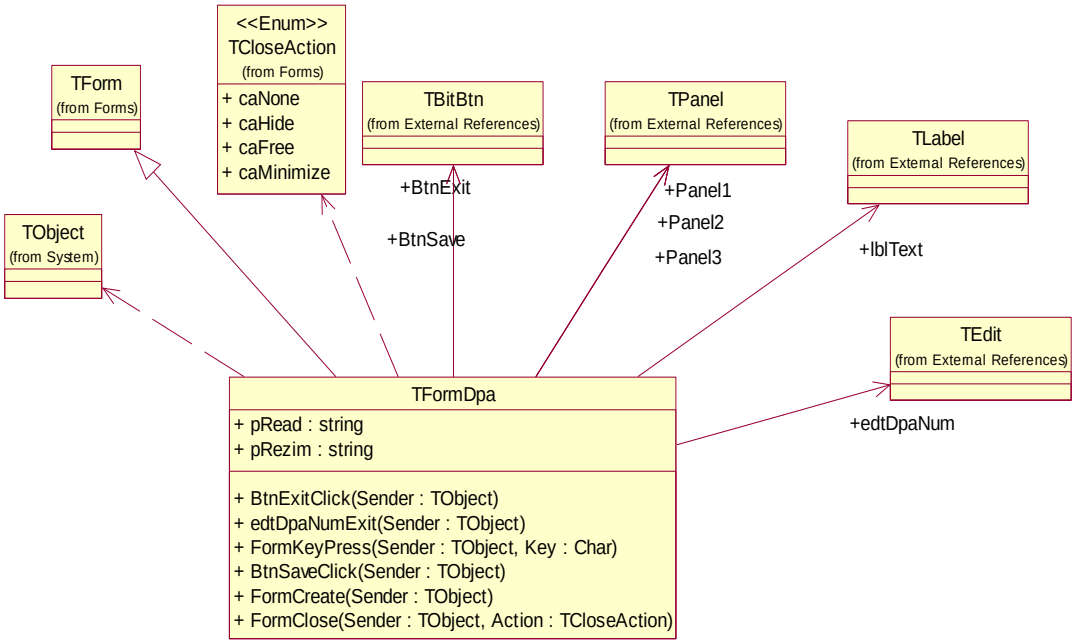

Рис. 6. Диаграмма классов формы интерфейса, полученная после обратной инженерии

Восстановление модели проектирования на основе исходного кода

Восстановление модели проектирования производилось также с применением средств обратной инженерии, имеющихся в IBM Rational Rose. Схема процесса обратной инженерии и типы полученных артефактов показаны на рис. 7.

Заключение

Современные условия разработки ПО требуют совершенствования соответствующих процессов, например, путем выбора действующих стандартов и внедрения их положений в деятельность разработчиков ПО. Важнейшими показателями совершенства процессов инженерии ПО полнота и качество документирования проектов [3]. Одна из проблем, возникающая при изменении стандартов качества документации или переходе на новые стандарты документирования – необходимость переработки спецификаций сопровождаемого и наследуемого ПО. Решение этой проблемы требует применения реинженерии. Эффективность процесса переработки документации может быть повышена путем определения исходных и требуемых артефактов спецификаций, соответствующих процессов и CASE-средств для их выполнения. Пример, реализации такого подхода для реинженерии одного из проектов ПО, рассмотренный в работе показал его практическую применимость и эффективность.

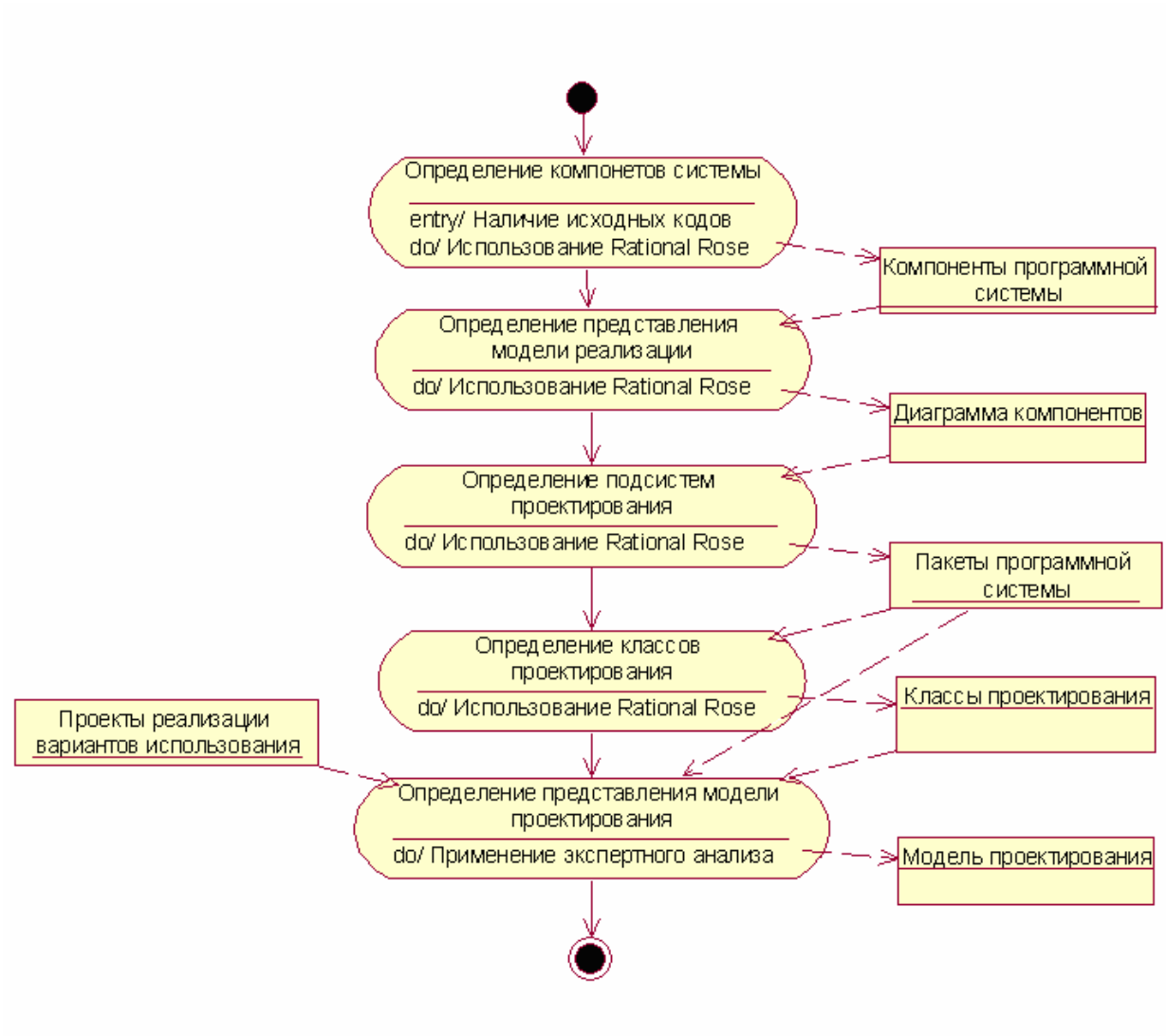

Рис. 7. Схема процессов восстановления модели проектирования

1. *Вигерс К.* Разработка требований к программному обеспечению / Пер. с англ. – М.: «Русская редакция», 2004. – 576 с.
2. *Сидоров Н.А.* Восстановление, повторное использование и переработка программного обеспечения // УсиМ. –-1998. – № 3. – С. 79–83.
3. *Ахен Д.М., Клауз А., Тернер Р.* CMMI: Комплексный подход к совершенствованию процессов. – М:.-МФК, 2005. – 330 с.
4. *Соммервил И.* Инженерия программного обеспечения. – М.: Издательский дом «Вильямс», 2002. – 624 с.
5. *Якобсон А, Буч Г., Рамбо Дж.* Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496 с.
6. *Боггс У., Боггс М.* UML и Rational Rose® 2002: Пер. с англ. – М.: Издательство «Лори», 2004. – 509 с.
7. Программное обеспечение IBM Rational. Методология и инструментальные средства разработки программных систем. Обзор продуктов и решений. – IBM Corporation, 2005. – 96 с. (http://www.ibm.com/software/rational/).
8. ГОСТ 19.101–77 Единая система программной документации.
9. ISO/IEC 19501:2005. Unified Modeling Language Specification. Version 1.4.2.
10. *Torii K., Matsumoto K.* Ginder2: An Environment for Computer-Aided Empirical Software Engineering // IEEE Transactions on Software Eng. – 1999. – 25, N 4. – P. 474–478.