Опыт использования UML при создании технической документации

Д. В. Кознов
dim@dk12687.spb.edu

А. Ф. Перегудов
peregudov@dip.ru

К. Ю. Романовский
kostet@tercom.ru

А. А. Кашин
archie_cashin@slawdog.com

А. Е. Тимофеев*
archie.cashin@gmail.com

В работе описывается опыт использования UML при создании документации для ПО комплекса автоматизации вещания. UML применялся для извлечения знаний — то есть как инструмент возвратной инженерии (*reverse engineering*) — и для составления иллюстраций в итоговом документе. При извлечении знаний использовался цикл «читатель/рецензент» из методологии SADT, который был изменен согласно специфике данной задачи. Применялись диаграммы компонент и кооперации. Описываемый в данной статье способ применения UML эффективен в случае, когда люди, изучающие систему, не пользуются ее программными кодами (например, они является техническими писателями, менеджерами и т. д.) и/или они не собираются участвовать в программировании этой системы и поэтому не нуждаются в детальной информации.

Введение

UML (*Unified Modeling Language*) — это язык для проектирования и визуализации программного обеспечения (ПО) [1]. На на-

* Санкт-Петербургский государственный университет кино и телевидения. 191126, ул. Правды, 13 Санкт-Петербург, Россия

стоящий момент он является общепринятым стандартом. Он широко используется в различных бизнес-компаниях по производству ПО, поддержан специальными программными инструментами[1]. Регулярно выходят новые версии стандарта, вокруг UML продолжают разворачиваться и развиваться многочисленные исследования. Однако много ожиданий, связанных с UML и визуальным моделированием вообще, так и не оправдалось. Чертежи в программировании не стали основой проектирования, подобно тому, как они являются стержнем процессов создания различных инженерных объектов — зданий, самолетов, автомобилей, пароходов, космических кораблей и т. д. UML не составил языка программирования следующего поколения, еще более, вслед за ассемблером и алгоритмическими языками, приблизившегося бы к человеческому восприятию и не потерявшего при этом машинной семантики. Чертежи, выполненные с помощью UML, например на этапе проектирования, нуждаются в дальнейшем сопровождении, так как само проектирование, как правило, невозможно единожды завершить[2]. Затратность процесса создания и сопровождения UML-чертежей во многих случая превышает выгоды от их использования. Зачастую применение UML оказывается дополнительной нагрузкой на процесс, поскольку UML используется механически или потому, что «так надо». В такой ситуации особую ценность приобретают описания успешного опыта практического использования UML. Это дает возможность профессионалам соотнести свои достижения непосредственно, минуя обобщения опыта в каких-либо технологиях с последующей расшифровкой и извлечением рациональных зерен при использовании этих технологий. В данной работе описывается случай использования UML в рамках процесса извлечения знаний о существующей системе [3] при создании технической документации для ПО комплекса автоматизации телевизионного вещания Санкт-Петербургской компании ЗАО «Фирма «ДИП»». При создании документации исполь-

[1] Так называемыми CASE-системами, самые известные из которых — IBM Rational Rose, Borland Together.

[2] Процесс разработки ПО является не водопадным, как в других инженерных областях, а итеративно-инкрементальным. То есть в проект постоянно вносятся изменения как в силу изменчивости ПО, так и из-за его сложности (изменчивость и сложность — два из четырех базовых свойств ПО, сформулированных Ф. Бруксом в [2].) В частности, ПО невозможно спроектировать сразу, перед началом процесса кодирования — часто происходят возвраты и уточнения проекта системы, а также поиск иных решений, и все это — уже в тот момент, когда идет реализация.

зовались диаграммы компонент (*component diagrams*) и кооперации (*cooperation diagrams*) UML. На диаграммах компонент отображалась статическая структура ПО, с помощью диаграмм кооперации создавались сценарии отдельных ее фрагментов и извлекалась информация для уточнения ее статической структуры. UML использовался для организации максимально эффективного взаимодействия технического писателя и экспертов. В данном проекте мы использовали методику «читатель/рецензент», предложенную в рамках метода структурного анализа SADT [4], модифицировав ее в соответствии с потребностями нашей задачи. Знания о системе, извлеченные с помощью UML, фиксировались как с помощью UML, так и в виде текста. Конечной целью было создание документа, включающего в себя иллюстрации, выполненные с помощью UML, а не просто создание некоторого набора UML-чертежей. При разработке документации использовалась XML-технология DocLine [5], созданная на кафедре системного программирования СПбГУ на основе известного продукта DocBook[3]. Данная технология позволила упростить процесс повторного использования отдельных фрагментов документации — это было оправдано, так как документация должна была иметь справочный характер, то есть содержать большое количество повторов. В рамках данного проекта технология DocLine была проинтегрирована с пакетом Microsoft Visio для удобства сопровождения UML-чертежей.

1. Комплекс автоматизации телевизионного вещания

Данный комплекс предназначен для формирования и выпуска в эфир телевизионных программ (*ТВ-программ*) в автоматизированном режиме с минимальным участием обслуживающего персонала. Сама ТВ-программа представляет собой непрерывную во времени цепь передач, каждая из которых включает в себя прямые включения с внешних линий, передачи из студии, а также сюжеты, воспроизводимые с видеомагнитофонов и видеосерверов. Для коммутации

[3] DocBook — это технология разработки технической документации на основе XML, обеспечивающая получение конечных документов различного формата (PDF, HTMLHelp и др.) на основе единого исходного текста [6]. На сегодняшний день DocBook — стандарт де-факто для документирования свободно распространяемого ПО (Open Source-проектов), в том числе Linux. Кроме того, данный продукт широко используется в рамках других проектов.

в эфир сигналов с различных устройств воспроизведения используется микшерно-коммутационное оборудование. Видеомагнитофоны, видеосерверы, коммутационное оборудование имеют дистанционное управление, что дает возможность автоматического формирования выходной ТВ-программы под управлением ПО автоматизации эфира в соответствии с загруженным расписанием передач. Использование подобных систем особенно актуально при выпуске новостных и спортивных ТВ-программ [7]. Появление подобного рода систем стало в настоящее время одной из областей продуктивного применения информационных технологий. Программные компоненты автоматизированных ТВ-систем функционируют на стандартных компьютерных платформах. Аудиовизуальный контент ТВ-передач производится и обрабатывается в файловой форме — форме медиаданных. Для сохранения медиаданных используются централизованные или распределенные файловые хранилища. Рабочие места пользователей реализуются в виде клиентских мест на базе PC, объединенных сетевой инфраструктурой управления, и передачи данных [8]. Рассматриваемый в данной работе автоматизированный ТВ-комплекс предназначен для решения следующих задач:

1. Формирования эфирной программы из различных передач, воспроизводимых с видеосерверов, видеомагнитофонов и внешних линий в соответствии с расписанием эфира или непосредственно по командам оператора.

2. Вставки межпрограммных и рекламных блоков.

3. Трансляции спортивных мероприятий в «хоккейном» (задержанном) режиме с возможностью удаления отдельных фрагментов транслируемого события или замены их на рекламные блоки.

4. Воспроизведения сюжетов для новостей или иных программ прямого эфира по команде выпускающего режиссера.

5. Синхронного многоканального воспроизведения сюжетов на устройства отображения (видеомониторы, проекционные системы и т. д.), расположенные в декорациях съемочного павильона.

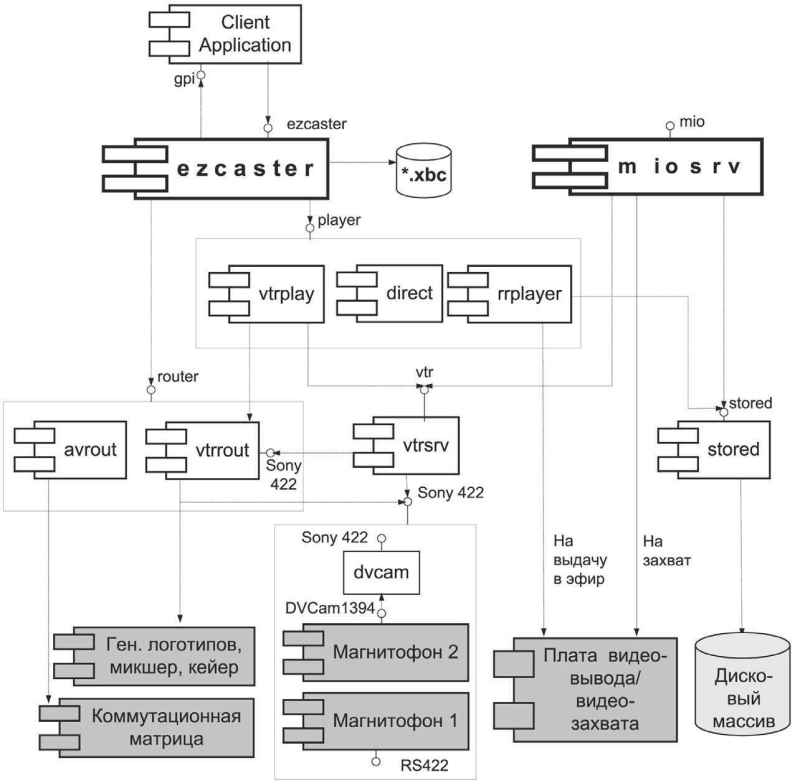

Рис. 1. Комплекс автоматизации вещания

Комплекс автоматизации вещания состоит из управляемого оборудования и управляющего программного обеспечения. ПО комплекса разделяется на серверное и клиентское. На рис. 1 представлена структура ПО серверной части и управляемое оборудование. Традиционные UML-компоненты обозначают здесь программные компоненты. Компоненты, окрашенные в серый цвет, представляют оборудование: видеомагнитофоны, платы видеовывода/видеозахвата, генератор логотипов и т. д. Программные компоненты, для удобства изображения, сгруппированы в более крупные компоненты, которые предоставляют одинаковый интерфейс, но изображаются в виде обычных прямоугольников. Например,

компоненты «vtrplay», «direct», «rrplay» управляют проигрыванием видеоматериалов на разных носителях, соответственно: на ленточных магнитофонах, внешних линиях, файловых носителях через платы видеовывода/видеозахвата. Эти компоненты реализуют одинаковый интерфейс «play», который скрывает различия типов материалов и способов их воспроизведения, унифицируя интерфейс управления воспроизведением. «Бочонками» изображены хранилища данных. Не закрашенные «бочонки» представляют логические хранилища данных — СУБД и конфигурационные файлы, а серые используются для изображения аппаратных хранилищ (дисковых массивов). Все программные компоненты могут быть распределены в Интернете. Они поддерживают XML/RPC интерфейсы для взаимодействия. Серверная часть комплекса может функционировать на платформах Windows и Linux.

2. Постановка задачи

После ряда успешных продаж системы оказалось, что ее дальнейшая эволюция затруднена в виду отсутствия хорошей документации. Существовало много описаний аппаратуры системы, а также ее функциональности. Но эти документы либо предназначались специалистам в области телевидения (инженерам, редакторам, телеоператорам и т. д.), либо являлись рекламными материалами. Почти полностью отсутствовала документация о системе, ориентированная на программистов и описывающая именно ПО комплекса. Были лишь скупые описания интерфейсов программных компонент, служившие, скорее, справочной информацией для самих авторов, плохо подходившие для обучения новых сотрудников и совсем не подходившие для инженеров, участвующих в проекте, менеджеров проекта, руководства компании. Необходимо отметить, что менеджеры проекта и руководители компании были по образованию телеинженерами, и в целом в компании для них был затруднен доступ к программистской информации. То есть существовал часто встречающийся на практике информационный барьер между программистами и специалистами других специальностей[4]. Итак, было решено создать пакет документов, описывающих ПО серверной части комплекса автоматизации телевещания, ориентирующийся на описание общей структуры ПО и всех интерфейсов

[4]Похожая ситуация, возникшая в одном крупном телекоммуникационном проекте, описана в [9].

его компонент. Не требовалось создать полный и исчерпывающий подстрочник кода. Документация должна была решить следущие задачи:

1) упростить процесс сопровождения и развития системы, в частности обучение новых специалистов;

2) упростить интеграцию комплекса с ПО заказчика;

3) улучшить понимание программистской части системы менеджментом проекта, а также инженерами и руководством компании[5].

Для этих целей было решено организовать отдельный специальный проект сроком на 4 месяца. При этом руководство компании поставило необходимым условием, чтобы ведущие специалисты-разработчики комплекса минимально отвлекались на проведение этой работы, а кроме самих документов был бы также создан процесс их дальнейшего сопровождения. Для этой цели был привлечен технический писатель, который являлся программистом по специальности и имел предварительные знания об информационных системах в телевидении. Предполагалось, что он будет работать на 1/4 рабочего дня. Кроме того, к работе был привлечен программист для создания документации в XML-формате с использованием технологии DocLine, а также для доработки этой технологии для нужд проекта. Программист имел высокую квалификацию и должен был работать также 1/4 рабочего дня[6]. Со стороны авторов системы к проекту было привлечено 3 эксперта: главный архитектор системы, который мог позволить себе не более 1 встречи в неделю на 2 часа, а также ответы на короткие вопросы по телефону, главный инженер системы, с которым состоялось несколько встреч в конце проекта, топ-менеджер компании, принимавший отчеты по текущей работе один раз в месяц.

[5] Мы столь подробно обсуждаем мотивы, контекст и особенности процесса создания документации, поскольку это трудоемкая и непростая задача, вроде бы понятная и нужная, но на практике чрезвычайно трудно выполнимая. Ф. Брукс еще в конце 60-х годов в своей знаменитой книге «Мифический человеко-месяц» писал, что «многие пытались на всю жизнь привить молодым программистам уважение к документам, преодолевающее лень и пресс графика работ. В целом нам этого не удалось.» Ситуация с тех пор радикально не изменилась.

[6] Фактически, вместо одного программиста над проектом работало трое. Однако, чтобы не затруднять дальнейшее изложение, мы оценили и сложили суммарное время их работы и обозначили единственную ставку.

3. Как все происходило

Технический писатель ознакомился со всеми документами, которые описывали комплекс автоматизации вещания. Однако эти документы были написаны с очень разных позиций и не давали целостного представления о ПО системы. Возникла необходимость в дополнительных источниках информации. Технический писатель отказался от изучения программного кода системы, поскольку это значительно увеличило бы трудозатраты на создание документации. Он обратился за помощью к экспертам. Однако выяснилось, что они способны рассказывать о системе очень много, долго и с воодушевлением, но технический писатель очень быстро терял нить их рассказа. Возникла необходимость в специальных средствах для управления общением с экспертами. Причем стало очевидно, что сами эксперты не способны предложить такую систему общения.

Как средство коммуникации технический писатель решил использовать UML. Были выбраны диаграммы компонент для изображения общей структуры ПО системы. Первая диаграмма выглядела, как показано на рис. 2. Технический писатель смог извлечь из документации только названия интерфейсов компонент и в общих чертах (весьма приблизительно!) уловить их назначение. Следующей задачей была идентификация самих компонент. Вместе с экспертом, имея чертеж, представленный на рис. 2, технический писатель дорисовал компоненты. Кроме программных компонент были идентифицированы также элементы оборудования, с которыми работает комплекс. Первая итерация идентификации компонент системы произошла очень быстро. Однако технический писатель осознавал, что имеющаяся у него картина поверхностна и содержит неточности. Он решил проводить дальнейшую идентификацию посредством изучения основных алгоритмов, по которым должна работать система. Тут он столкнулся с недоумением эксперта, который, будучи главным архитекторам системы, создавал ее для работы по бесконечному количеству различных алгоритмов и не мог обозначить главные алгоритмы.

Тогда технический писатель решил оттолкнуться от бизнес-функциональности, но ему потребовалось время, чтобы осознать, какие из бизнес-функций, обозначенных в проектной документации, базисные, какие — второстепенные, а какие — рекламные. Делал он это постепенно, вначале «нащупав» первую функцию, окончательное описание которой изображено на рис. 3. В итоге оказа-

Рис. 2. Первый вариант диаграммы компонент

лось, что комплекс автоматизации телевещания реализует всего две базисные бизнес-функции. При изучении базисных функций системы использовались диаграммы кооперaций UML. Было решено не строить полное описание всех возможных алгоритмов, но подробно описать лишь главные ветки, которые задействуют основные компоненты системы. В описаниях допускались лишь небольшие вариации этих веток. Главным UML-чертежом был тот, который изображен на рис. 1. Все остальные UML-диаграммы его так или иначе дополняли, комментировали и т. д. При создании и обсуждении чер-

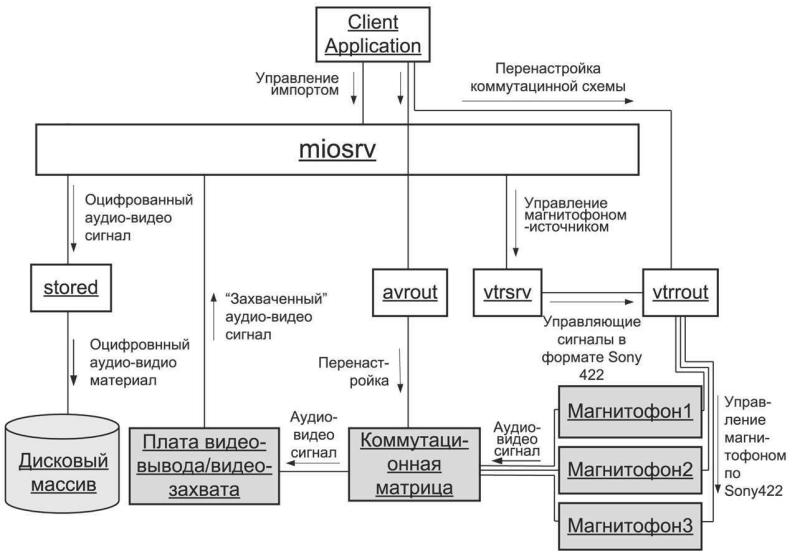

Рис. 3. Пример диаграмы кооперации

тежа, изображенного на рис. 3, и других, подобных ему, чертеж с рис. 1 существенно дополнился и уточнился. Главным образом уточнялись связи между компонентами, реже появлялись новые компоненты[7]. Постепенно сложился следующий режим работы над документацией. На каждую встречу с экспертом технический писатель приносил UML-чертежи, на которые он предварительно внес информацию, полученную им в прошлый раз. Часто бывало, что при подготовке этих чертежей у технического писателя возникали дополнительны вопросы. Если они были несложные, то они обсуждались с экспертом по телефону. Эксперт проверял UML-чертежи, указывал на ошибки, при обсуждении которых возникала новая (для технического писателя) информация о системе. Технический писатель ориентировался на чувство информационного насыщения и сбалансированности деталей, целостности, также обращая внимание на мнение эксперта о полноте данного фрагмента описания

[7] Идея использовать динамические модели для идентификации структурных составляющих системы содержится в [10]. Однако там она использовалась в контексте «прямого» проектирования (*forward engineering*), в то время как мы использовали ее в контексте возвратной инженерии.

системы[8]. Важно отметить, что активно обсуждаемых чертежей в данном проекте было совсем немного — всего четыре. На каждой встрече экспертов и технического писателя обсуждались одни и те же чертежи, эксперты к ним привыкли и не тратили много времени на их узнавание и изучение. Чертежи «схватывали» узловые аспекты системы, поэтому дорабатывались долго, параллельно с тем, как углублялись и совершенствовались знания о системе технического писателя. В то же время они служили средоточием обсуждений, нитью дискуссий, как это описывалось в работе [11]. При включении этих чертежей в итоговый документ они были существенно доработаны и улучшены. Также было создано два дополнительных чертежа для декомпозиции информации, содержащейся в исходных. Со второго месяца работы технический писатель, накопив достаточно связной информации о комплексе автоматизации телевещания, начал создавать итоговый документ. Документ и содержащиеся в нем UML-чертежи стали обсуждаться с менеджером проекта и компании. При этом обнаружилось, что чертежи нарисованы непривычным для инженеров способом, а терминология документации не соответствует той, которая принята в телеиндустрии. Около месяца ушло на устранение этих недостатков.

4. Архитектура средств автоматизации

Со второй половины проекта (начало 3-го месяца работы) разработка документации происходила методом внесения изменений в базовую версию[9]. К этому моменту, кроме чертежей, появились первые версии итогового документа: была создана его структура, количество и общий вид иллюстраций были также определены и впоследствии почти не менялись. Кроме того, обнаружилось, что документация имеет довольно много повторов в силу того, что она должна была быть во многом справочником. Например, краткое

[8] Объем информации о сложных системах, к которым относится и комплекс автоматизации телевещания, как правило, значительно превышает возможности восприятия изучающего (в силу его квалификации, временных и других ограничений). Кроме того, весь этот массив информации со всеми деталями часто изучающему и не нужен. Трудность этой ситуации заключается в том, что сам изучающий определяет, когда нужно остановиться в изучении данного аспекта системы и к какому аспекту перейти дальше [3].

[9] Фактически, при разработке документации мы следовали методу, предложенному Ватсом Хамфри в знаменитой монографии [12] применительно к разработке всего ПО.

описание каждого интерфейса компоненты встречалось два раза — первый раз при перечислении интерфейсов, для краткой характеристики, второй раз — предваряя его детальное описание. Наконец, возникла потребность в профессиональном полиграфическом оформлении документации для создания брошюры. Было также желательно представить ее в виде on-line help системы. В силу этих причин было решено использовать технологию DocLine [5], которая, реализуя идею single sourcing [6], позволяла создавать на основе XML один общий исходный текст документации и генерировать по нему различные представления — PDF-формат, HTML и WinHelp. Кроме того, данная технология позволяла повторно использовать фрагменты текста. В этот момент к проекту был привлечен программист для ввода документации и UML-чертежей в формат DocLine, оперативного внесения изменений в документацию, а также для доработки технологии. Последнее понадобилось для того, чтобы автоматизировать процесс публикации в конечном тексте измененных UML-чертежей. С UML-чертежами удобно работать в CASE-пакете. В данном случае использовался UML Addon пакета Microsoft Visio. Но чертежи должны вставляться в конечную документацию в полиграфических форматах, например jpg, eps, emf и т. д., с соответствующей компрессией. Если не автоматизировать процедуру создания новых версий UML-иллюстраций из чертежей CASE-пакета и их вставку в итоговые документы, то велика вероятность потери каких-либо версий, так как поток изменений чертежей в течение работы был значительным, а также растянутым во времени. Эта автоматизация и была реализована в DocLine в рамках данного проекта.

В целом техническая сторона создания новой версии документа выглядела так, как показано на рис. 4. Процесс внесения изменений в документацию был следующим. Технический писатель посылал программисту очередной измененный документ в виде файла в формате Microsoft Word. Все новые изменения были помечены в нем в виде исправлений (специальный режим работы в этом пакете). Программист вносил эти исправления в итоговую версию и высылал техническому писателю автоматически сгенерированные PDF-, HTML- и WinHelp-варианты документации. Необходимо отметить, что не все исправления высылались программисту таким способом. Поскольку технический писатель читал распечатанные документы с листов, мелкие исправления он делал ручкой и отдавал программисту правку в неэлектронном виде.

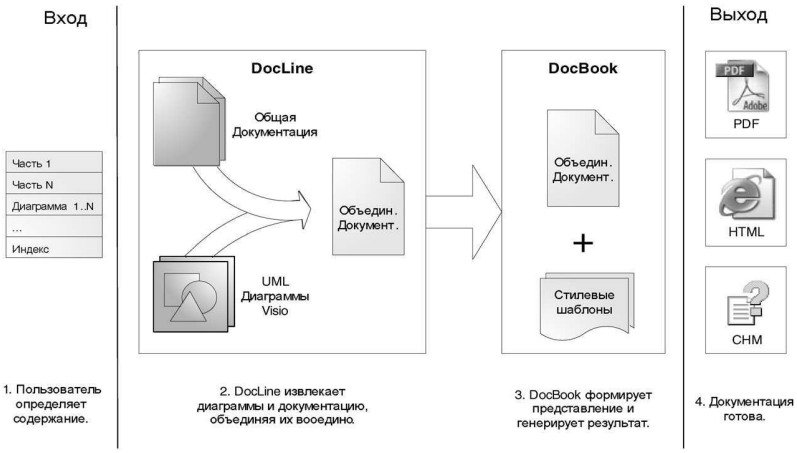

Рис. 4. Схема процесса разработки документации

5. Особенности использования UML

UML в данном проекте использовался для изучения существующей системы [3]. Его роль была особенно велика, поскольку в качестве основного источника информации о системе выступали эксперты[10], и для налаживания эффективных коммуникаций с ними требовались дополнительные средства. В итоге UML использовался двумя способами:

1) как средство извлечения знаний и организации коммуникаций технического писателя с экспертами;

2) в качестве средства создания иллюстраций для итогового текстового документа.

[10] Технический писатель не пользовался программным кодом системы как источником информации. Иначе UML использовался бы по-другому — в режиме автоматического возвратного проектирования, для визуализации программного кода. Соответствующий подход к использованию визуального моделирования описан, например, в [13]. Однако такая автоматизация оказывается востребованной в контексте реинжинирига системы, то есть в ситуации, когда программный код является основным источником информации, а задачей изучающих систему является не просто ее описание, а трансформация, сопровождение и поддержка системы. Если же новый программист вливается в уже существующий коллектив, то для извлечения знаний он, как правило, читает код и задает вопросы товарищам — в большинстве случаев, как показывает опыт, этого оказывается вполне достаточно.

В первом случае его использование фактически основывалось на цикле «читатель/рецензент» методологии SADT [4]. Однако этот цикл был существенно «облегчен», то есть не соблюдались многочисленные формальности общения. Кроме того, использовались диаграммы UML, а не диаграммы структурного анализа. Поскольку целью моделирования было описание статики системы, а не ее алгоритмов[11], то с помощью статических моделей фиксировалась структура ПО, которая уточнялась и дополнялась путем построения динамических моделей. Из множества динамических видов моделей, представленных в UML, использовались диаграммы кооперации. Родственные им диаграммы последовательностей не использовались, поскольку в изображаемых алгоритмах было слишком много сущностей, каждая из которых не очень интенсивно взаимодействовала с остальными. Как видно из рис. 3, в среднем каждая сущность принимает/посылает 1–3 сообщения. Диаграммы последовательностей лучше использовать, когда взаимодействующих сущностей меньше, а количество сообщений, связанных с каждой из них, больше. Тогда начинает «работать» преимущество этих диаграмм, связанное с удобством изображения временных последовательностей событий. В нашем проекте использовалось небольшое количество UML-чертежей. Основная работа по моделированию выражалась не в создании все новых и новых чертежей, а в уточнении и доработке уже имеющихся. Чертежи служили, с одной стороны, для аккумуляции и сгущения знаний технического писателя, повышению их концентрации, с другой стороны, для «вливания» технического писателя (и, как следствие, его продуктов) в атмосферу, терминологию, бытие системы. Два эти процесса нераз-

[11] Описание алгоритмов ПО — это непростая задача. Для уже существующей системы не всегда это можно сделать на более высоком уровне абстракции, чем программный код, так, чтобы описание получилось полное и легко воспринималось неподготовленным читателем. Ситуация упрощается, когда для более высокого уровня абстракции существует подходящая метафора. Примером такой метафоры может служить конечно-автоматная модель, которая, с одной стороны, имеет хорошие средства визуализации в виде диаграмм состояний и переходов, с другой стороны, подходит для описания различного ПО. Такое ПО принято называть реактивными системами (подробнее про использование конечных автоматов при описании ПО можно прочитать в работе [14]). Однако если данная метафора не использовалась при создании ПО, то «навязать» ее при составлении описаний фактически невозможно. Если все же это попытаться сделать, то система будет фактически перепроектирована в рамках ее изучения. При этом окажется, что абстрактные алгоритмы отличаются от их прообразов в программном коде.

рывны, но на практике первый часто оказывается без второго. То есть человек вникает в информацию, изучает ее, но использует для создания какого-то своего мира, своей системы. Его чертежи и тексты трудно понимать, он изобретает свою терминологию предлагает свои концепции. В частности, он иногда без устали создает UML-диаграммы, пытаясь задействовать максимум возможностей данного языка. В этой ситуации чем образованней оказывается изучающий, тем ужасней последствия его работы! Диаграммы, созданные нами в процессе извлечения знаний о системе, использовались в дальнейшем как иллюстрации в итоговом документе. Однако при этом они подверглись существенной переработке. Наивно полагать, что UML-спецификации, созданные для одной цели в рамах какого-то определенного вида деятельности, могут быть повторно использоваться «as is» в рамках другого вида деятельности. Этот вопрос подробно обсуждался в работе [3], где доказывалось, что это связано с психологическими особенностями использования UML. В связи с этим, при использовании UML-чертежей, созданных техническим писателем в процессе изучения системы, в качестве иллюстраций в итоговом документе, они подверглись следующей обработке:

1. Тщательному обсуждению с менеджерами компании на предмет соответствия их структуры и терминологии стандартам телеиндустрии; по результатам этих обсуждений все чертежи были полностью перерисованы, однако с полным сохранением исходной информации.

2. Чертежи подверглись дополнительной доработке в плане эстетических свойств: для разных сущностей использовались различные размеры в соответствии с их доминированием[12], выбирались различные величины для толщины линий при изображении разных сущностей и связей, использовался оптимальный размер стрелок и т. д.; пример представлен на рис. 1.

3. Чертежи выверялись на предмет точности закладываемых в них психологических характеристик, что важно для формиро-

[12] В SADT доминирование выражалось с помощью расположения — «более важные» сущности отображались выше «менее важных» и зависимых [4]. Однако вертикально-горизонтальная иерархия сущностей на чертеже может иметь много различных оттенков семантики, поэтому явно доминирующие компоненты изображались большими размерами, а их границы — более «толстыми» линиями.

вания определенного впечатления у читателей и позволяет создавать благоприятную атмосферу при изучении материала, как это описано в [3, 15]; например, чертеж может вызывать чувство ясности, легкости или он перегружен деталями, смутен, агрессивен; эти впечатления в фоновом режиме сильно влияют на процесс восприятия материала; первое впечатление от чертежа формируется у читателя, как правило, очень быстро, часто — с первого взгляда на чертеж, то есть еще до того, как человек начал его подробно изучать[13].

Наконец, необходимо отметить, что для целей данного проекта на основе UML был создан специальный диалект — визуальный язык проекта. Феномен таких языков подробно обсуждался в работах [16, 17]. Внимательный читатель может обнаружить на чертежах, представленных на рис. 1 и 3, черты, отсутствующие в классическом UML. Однако все изменения UML выражаются с помощью его extension-механизма [1].

Заключение

В проекте по созданию документации, представленном в данной работе, был достигнут баланс между UML-чертежами и текстовыми описаниями системы. UML-чертежи способствовали созданию текста, а также гармонично вошли в итоговое описание системы. Само качество чертежей лишь отразило качество проектирования системы. Гармонично изобразить структуру ПО возможно лишь в том случае, когда эта гармоничная структура существует, то есть ПО хорошо спроектировано. UML же является лишь средством, инструментом, который позволяет, при умелом использовании, выразить существующее положение вещей, но не создать это положение. То есть, если структура ПО хаотичная, смутная, едва угадывается или не угадывается вообще, невозможно создать удачные чертежи[14]. В меньшей степени удалось создать процесс сопровождения

[13] Очевидно, что динамический чертеж, представленный на рис. 3, проигрывает статическому чертежу, представленному на рис. 1, в богатстве психологической информации. Это можно считать недостатком данного проекта. Но необходимо отметить, что динамика на чертежах выражается сложнее, чем статика.

[14] При «прямом» проектировании процесс создания UML-чертежей не заменяет процесса проектирования, то есть создания той самой структуры, которая с помощью UML только изображается. Особенности использования UML при проектировании обсуждались нами в [3].

документации, легко осуществляемый и сопровождаемый в дальнейшем. Основной проблемой оказалась трудность внесения изменений в документацию с помощью XML. Технический писатель не стал заниматься этой работой и для этого был привлечен специальный программист. При дальнейшем сопровождении необходимо решить этот вопрос по-другому, поскольку слишком дорого содержать двух человек — технического писателя и программиста — для сопровождения документации, а технические писатели нечасто и неохотно соглашаются пользоваться XML[15]. Один из возможных вариантов решения этой задачи — интеграция DocLine с Microsoft Word[16]. В качестве дальнейшего развития описанного в работе подхода может быть названо исследование особенностей применимости других типов UML-чертежей к процессу извлечения знаний о существующей системе, изложенному в данной статье. В частности, интересно исследовать диаграммы последовательностей.

Список литературы

[1] Unified Modeling Language (UML) Specification: Superstructure. Version 2.0. — http://www.omg.org.

[2] *Sommerville I.* Software Engineering. — Addison-Wesley, 6th edition, 2001. — 693 p. (Русский перевод: *Соммервилл И.* Инженерия программного обеспечения. — Издательский дом «Вильямс», 2002. — 623 с.)

[3] *Кознов Д. В., Перегудов А. Ф.* «Человеческие» особенности использования UML // Наст. сборник.

[4] *Marca D. A., McGowan C. L.* SADT Structured Analysis and Design Technique. — McGraw-Hill, 1988.

[15] Создание документации как текста и программирование на XML (даже простейшее) оказываются существенно разными психическими функциями, плохо сочетающимися в одном человеке. Более того, эти функции трудно сочетать одномоментно одному человеку в рамках одного проекта — создавать понятный, хорошо читаемый текст и сразу же формировать для него четкую структуру. Структурирование, как правило, вторично — нужно, чтобы сначала образовалось то, что необходимо структурировать. Таким образом, в частности, целесообразно проводить XML-оформление документа отдельно и/или «скрывать» его автоматизацией от технического писателя.

[16] В качестве примера может быть приведен пакет для формализации требований — IBM Requisite PRO. При работе с требованиями также необходимо поддерживать некоторую гиперструктуру (требования с зависимостями) и также нужно предоставить удобный интерфейс для работы, не сильно отличающийся от редактирования текста. В итоге в этом продукте поддержана разметка требований прямо в пакете Microsoft Word.

[5] *Романовский К.* Поддержка вариативности общих активов в семействах программных продуктов // Материалы 9-й Санкт-Петербургской ассамблеи молодых ученых. — Санкт-Петербург, 2004.

[6] *Ament K.* Single Sourcing: Building Modular Documentation. — William Andrew Publishing/Noyes Publications, 2002.

[7] *Гласман К. Ф., Перегудов А. Ф.* Автоматизация телевизионного вещания // Информационно-технический журнал «625». — №6. — 1998. — С. 5–8.

[8] *Перегудов А. Ф.* Сетевые решения в телевизионном производстве // Информационно-технический журнал «625». — № 10. — 2004. — С. 10–15.

[9] *Koznov D., Kartachev M., Zvereva V., Gagarsky R., Barsov A.* Roundtrip Engineering of Reactive Systems // Proc. 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA). — 2004. — P. 343–346.

[10] *Douglass B. P.* Real Time UML: Advances in the UML for Real-Time Systems. — Addison-Wesley, 2004.

[11] *Koznov D. V.* Visual Modeling and Software Project Management // Proc. 2nd International Workshop «New Models of Business: Managerial Aspects and Enabling Technology». — 2002. — P. 161–169.

[12] *Humphrey W.* Managing the Software Process. — Addison-Wesley, 1990.

[13] *Бабурин Д. Е., Бульонков М. А., Емельянов П. Г., Филаткина Н. Н.* Средства визуализации при перепроектировании программ // Программирование. — № 2. — 2001.

[14] *Кознов Д. В.* Языки визуального моделирования: проектирование и визуализация программного обеспечения. Учебное пособие. — СПб: Изд-во СПБГУ, 2004. — 170 с.

[15] *Кознов Д. В.* Коммуникативный аспект визуального моделирования при разработке программного обеспечения // Ежегодник российского психологического общества: материалы III всероссийского съезда психологов. — Т. 4. — 2003. — С. 303–308.

[16] *Кознов Д. В., Ольхович Л. Б.* Визуальные языки проектов // «Системное программирование». — СПб: 2004. — С. 148–168.

[17] *Ольхович Л. Б., Кознов Д. В.* Метод автоматической валидации UML-спецификация на основе языка OCL // Программирование. — № 6. — 2003. — С. 44–50.