УДК 602-507

НОВАЯ ИНИЦИАТИВА В ПРОГРАММИРОВАНИИ. ДВИЖЕНИЕ ЗА ОТКРЫТУЮ ПРОЕКТНУЮ ДОКУМЕНТАЦИЮ

А. А. Шалыто,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Обосновывается целесообразность разработки открытой проектной документации на программное обеспечение.

Here we proved advantages of developing of open project documentation for software.

Простота требует проектирования и хорошего вкуса.

Л. Торвальдс

Глубоко ошибается тот, кто думает, что изделиями программистов являются программы, которые он пишет. Программист обязан создавать заслуживающие доверия решения и представлять их в форме убедительных доводов, а текст написанной программы является лишь сопроводительным материалом, к которому эти доказательства применимы.

Э. Дейкстра

Недавно я стал свидетелем того, как один выдающийся программист (участник двух финалов командных чемпионатов мира по программированию, проводимых организацией «Association for Computing Machinery» — ACM), в течение весьма продолжительного времени (15 мин) не мог понять программу из *шести строк* на языке Си, про которую было известно, что она решает классическую задачу (другое решение которой программист знал). После этого мы вышли в Интернет и сравнительно быстро нашли ту работу, где был весьма внятно описан алгоритм, использованный в указанной программе.

Почему исходные тексты не решают проблему понимания программ?

Данный факт в очередной раз подтвердил мое мнение, что «Движение за открытые исходные тексты» («Open Source Software») не обеспечивает даже в этом достаточно простом (относительно реальных проектов) случае решения проблемы понимания программ. К счастью, так думаю не только я один. Н. Безруков, ведущий аналитик корпорации BASF, профессор университета Fairleigb Dickinson (NJ), Web-мастер www.softpanorama.org — Open Source Software, считает [1]: «Центральный вопрос в практике программирования — вопрос о понимании программных текстов. Всегда хорошо иметь исходники, но проблема состоит в том, что зачастую их недостаточно. Чтобы понять некоторую нетривиальную программу, обычно требуется дополнительная документация. Эта потребность растет экспоненциально с ростом объема кода. Анализ текстов программ, направленный на восстановление первоначальных проектных решений, принятых разработчиками, и понимание программ являются двумя важными ветвями технологии программирования, существование которых неразрывно связано с недостаточностью исходных текстов для понимания программ. В качестве примера попробуйте понять структуру нетривиального компилятора при условии, что вы не располагаете определением того языка, который им компилируется.

Каждый, кто участвовал в крупном проекте по реконструкции ПО, навсегда запомнит то *чувство беспомощности и потерянности*, которое испыты-

ваешь, когда впервые видишь гору плохо документированных (но не всегда плохо написанных) исходных текстов. Доступность исходных текстов не слишком помогает, когда отсутствует доступ к ключевым разработчикам. Если программа написана на сравнительно низкоуровневом языке типа Си и плохо документирована, то все основные проектные решения обычно растворяются в деталях кодирования и требуют реконструкции. В таких ситуациях ценность документации более высокого уровня, такой как спецификация интерфейсов и описание архитектуры, может *превышать ценность самого исходного текста*.

Осознание неадекватности исходных текстов для понимания программ привело к попыткам объединить код и документацию более высокого уровня. Одна из наиболее известных попыток решения указанной проблемы была предпринята Д. Кнутом в книге «Грамотное программирование» [2]. Возможно, самой известной запрещенной в истории компьютерных наук была книга «Commentary on Unix:With Source Code» [3], содержавшая высокоуровневое описание исходных текстов UNIX наряду с используемыми алгоритмами. Она нелегально копировалась и распространялась более 20 лет с момента ее первой публикации в 1977 г.

Если вы не принимали участия в работе над проектом, начиная с его ранних стадий, то сложность и объем фактически «закрывают» от вас исходные тексты, при условии, что отсутствует документация высокого уровня. Понимание «доисторического» кода в случае отсутствия первоначальных разработчиков или адекватной документации, позволяющей разобраться с соответствующими архитектурными решениями, является, вероятно, одним из труднейших видов программистской деятельности.

И еще. «Существуют ли какие-нибудь достойные свободные программы, вид которых не вызывает отвращения? До сих пор, несмотря на обилие свободного кода, нормальных программ среди него ужасно мало» [4].

О том, к чему это приводит, хорошо сказал великий русский математик, академик Л. С. Понтрягин: «Только хорошо выполненная работа дает радость! Выполненная небрежно, она вызывает отвращение и постепенно вырабатывает в человеке аморальное отношение к труду» [5].

Почему программы не проектируются?

Итак, без исходных текстов плохо, но и с ними также нехорошо. Чего же не хватает для полного счастья? Ответ прост: *проектной* документации, выполненной весьма подробно и аккуратно, в которую *программная* документация входит как одна из составляющих.

Мне на ум приходят только три вещи, которые создаются без разработки проектной документации: дети, картины и программы. При этом использование документации не зависит от серийности — платье и в домашних условиях обычно шьется не кое-как, а по выкройке.

Мосты, дороги и небоскребы без документации обычно не строятся, а вот о программах этого не скажешь. Кстати, в археологии тоже нельзя копать, где и как Бог на душу положит, так как при этом могут возникать серьезные неприятности.

В программировании сложилась ситуация, определяемая вторым законом Вейнберга: «Если бы строители строили дома так, как программисты пишут программы, достаточно было бы одного-единственного дятла, чтобы разрушить цивилизацию» [6].

Почему на аппаратуру выпускается море подробной и внятной проектной документации, в которой специалист средней квалификации может сравнительно быстро разобраться и изменить ее даже через много лет после выпуска, а для программ такая документация либо вовсе отсутствует, либо она пишется чисто формально и для ее корректировки (если разработчик отсутствует) требуется специалист более высокой квалификации?

Видимо, причина этого заключается в следующем. Во-первых, аппаратуру разрабатывают одни люди (организации, подразделения), а изготавливают другие. Поэтому при некачественной документации разработчик проведет на «заводе» остаток жизни, чего ему делать явно не хочется. При создании программ имеет место другая ситуация, так как в этом случае обычно и разработчик, и изготовитель программ — одно и то же лицо, и поэтому какую номенклатуру документов ни создавай, их содержимое будет, как правило, весьма поверхностным. Во-вторых, аппаратура — «жесткая», а программа — «мягкая». Это упрощает внесение изменений в программу, но не служит основанием для того, чтобы вовсе не выпускать проектную документацию. Известно, что большинство программистов патологически не желают читать и уж тем более писать документацию [7].

Опыт показывает, что практически ни один молодой программист, даже очень толковый, не умеет писать проектную документацию. Несмотря на то, что многие программисты слушали и сдавали большие и сложные курсы математики, это почти никак не отражается на логике и строгости написания документации. Они, например, никак не могут во всей документации (вне зависимости от ее размера) использовать одно и то же обозначение, и поэтому один и тот же предмет называется, например, то лампа, то лампочка, то lamp, то с большой буквы, то с маленькой. И это еще не предел их фантазии!

Это, видимо, происходит потому, что при программировании компилятор подсказывает несоответствия, в то время как при написании проектной документации подсказывать некому.

Вопрос о качестве документации на программное обеспечение приобретает все большее социальное значение. Так, в статье [8], опубликованной в газете «Известия», первым среди вопросов, «о чем мы должны подумать задолго до того, как кончится нефть», является качество указанной документации. «Наиболее запомнившееся и поразившее меня при выполнении совместного проекта с фирмой IBM отличие состояло в том, как *они* относятся к доку-

ментации. У *них* было принято, и это соответствовало практике, — что написано в документации, то так и есть, так и работает. У *нас* — никогда! У *них* из этого проистекала корпоративная уверенность в себе — на все вопросы *они* находили ответ в документации, утвержденной начальством. У *нас* не только документация плоха, но и мысли изложить толком никто не умеет. Я много раз сравнивал *их* доклады и *наши* „доклады". Однажды мы пригласили одного из лучших переводчиков. Он потом извинялся, что не смог толком перевести на английский, но я его успокоил, что и на русском-то ничего не понял... Этому нужно научиться. Эту часть работы следует любить и уважать». Несомненно, что и у *них* с документацией не так все хорошо, как говорит автор, но тенденция просматривается...

Разработка программ все больше напоминает шоу-бизнес с его погоней за прибылью. Все делается в дикой спешке, без раздумий о том, что будет с продуктом через некоторое время, особенно длительное. Как и в шоу-бизнесе, в программировании используются критерии «выгодно и невыгодно», а не «хорошо и плохо». В большинстве случаев хороша не та технология, которая действительно хороша, а та, которая выгодна.

Нежелание писать проектную документацию, вероятно, связано и с тем, что, чем более закрыт (не документирован) проект, тем более незаменим его автор.

Подобный стиль работы, к сожалению, распространяется и на создание программного обеспечения для особо ответственных систем. Поэтому не удивителен случай, который стал недавно широко известен. В центре Парижа из-за отказа компьютерной системы управления в новом дорогом автомобиле чуть не погибли люди, так как кондиционер перестал работать, двери не открывались, а бронированные стекла даже снаружи удалось разбить далеко не сразу.

Это во многом связано с тем, что в большинстве случаев программы пишутся, а не проектируются. «При проектировании любая техника сложнее CRC-карточек [9] или диаграмм использования [10] считается слишком сложной и не применяется. Программист всегда может отказаться от любой технологии, сказав начальнику, что он не укладывается в срок» [11]. Все это приводит к тому, что даже «пользователи не считают ошибочное программное обеспечение чем-то из ряда вон выходящим» [7].

Заметим, что такая ситуация в программировании имела место далеко не всегда — при использовании «больших» машин программы либо проектировались, либо писались очень тщательно, так как после нахождения ошибки следующий «заход» на машину происходил обычно не ранее, чем через сутки. Таким образом, технический прогресс привел к менее ответственному отношению к программированию.

Что дает проектная документация?

При наличии качественной проектной документации программист не сможет «управлять» менеджерами. После его увольнения на продолжение проекта можно нанять человека с более низкой квалификацией и зарплатой, а не с более высокой, как это обычно бывает. В конечном счете, в цивилизованной стране средний программист не должен получать больше среднего школьного учителя.

Можно ли учиться проектированию и реализации программ по книгам? Да, но по разным: проектированию — по одним [12], реализации — по другим [13]. Книг, в которых прослеживаются оба этих этапа, почему-то нет. Их отсутствие могут восполнить *открытые* проекты.

Наконец-то появилось третье издание книги [14], содержащей более 1150 страниц, в которой проектирование и реализация рассматриваются совместно на примере моделирования работы лифта для двухэтажного здания, на каждом этаже которого может находиться не более одного человека. Однако, во-первых, эта книга одна, во-вторых, этот пример один. Это напоминает библиотеку, содержащую только одну книгу. Поэтому задача создания библиотеки с большим числом открытых проектов является актуальной.

Программное обеспечение с подробной открытой проектной документацией, в которой программная документация является лишь составной частью, может рассматриваться в качестве новой разновидности паттернов проектирования [15], позволяющих достаточно просто оценить как достоинства, так и недостатки выполненных проектов. Рефакторинг (изменение структуры программы без изменения ее функциональности) [16] или перепроектирование программы в этом случае может быть выполнено значительно проще, чем при наличии только исходных текстов. Проектная документация должна содержать, в частности, формальную спецификацию, по крайней мере, на логику разрабатываемой программы, так как «то, что не специфицировано формально, не может быть проверено, а то, что не может быть проверено, не может быть безошибочным» [17], а также в связи с тем, что, «если нет спецификации, то нет и ошибок» [18].

Кроме того, проектная документация должна содержать «протоколы (истории вычислений), которые являются конструкциями, вскрывающими механизм работы программы, и поэтому среди теоретиков программирования складывается представление, что множество протоколов лучше характеризует программу, нежели сам исходный программный текст» [19].

Отметим также, что без качественной проектной документации одно из основных достоинств объектно-ориентированного программирования — повторное использование кода — может приводить к большим неприятностям [20].

И самое важное. Проектная документация необходима, так как из теории алгоритмов известно (теорема Райса), что, в общем случае, по тексту программы невозможно алгоритмически доказать истинность никаких нетривиальных свойств функции, вычисляемой при помощи данной теории.

Для понимания программ (в отличие от их исполнения), следуя Тьюрингу, требуется «проницательность и изобретательность». Но так как анализ текстов программ все равно не поддается автоматизации, а анализ «вручную» требует колоссальных затрат времени, то кажется, что по одному лишь тексту программы невозможно сделать выводы о ее свойствах.

Однако это не так. Математики нашли решение аналогичной проблемы еще в античности. Это запись доказательства понятным человеку языком, что и отличало греческую математическую школу от древнеегипетской. В последней в качестве решения геометрической задачи представлялся чертеж, снабженный «пояснительной» надписью «Смотри!». Это сродни представлению текста программы для определения того, что она делает.

Для того чтобы другие могли понять не только *как* работает наша программа, но и *что* она делает и *каким образом* она это делает, им необходимо представлять подробное описание (на понятном человеку даже «средней» квалификации формализованном языке) как процесса создания программы, так и ее свойств (статических и динамических), что и является, по сути, проектной документацией. Она и служит подтверждением того, что представленная программа вычисляет данную функцию — соответствует техническому заданию.

Движение за открытую проектную документацию

Исходя из изложенного и не найдя в Интернете исчерпывающей документации практически ни на один программный проект, мы (автор и профессор В. Г. Парфенов, декан факультета информационных технологий и программирования С.-Петербургского государственного института точной механики и оптики (технического университета)) решили «вызвать огонь на себя» и объявили в ноябре 2002 г. в Санкт-Петербурге на открытии полуфинальных соревнований Северо-Восточного Европейского региона командного чемпионата мира по программированию ACM об образовании «Движения за открытую проектную документацию» (Stream for Open Project Documentation), в поддержку которого был создан сайт http://is.ifmo.ru.

Для придания импульса этому движению я начал педагогический эксперимент со студентами кафедры «Компьютерные технологии» СПбГИТМО (ТУ), суть которого состояла в том, что студенты разделились примерно на 40 групп (от одного до трех человек), каждая из которых должна была выполнить заинтересовавший ее проект на основе автоматного программирования (программирование с явным выделением состояний) [21], но так, чтобы работа завершилась размещением проекта на указанном сайте.

В отличие от многих студенческих работ, «выложенных» в Интернет в соответствии с лицензией *AS IS* («как есть» — без ответственности за ошибки) [22], мы очень старались. Выполнение каждого проекта занимало у студентов несколько десятков часов, из них не менее десяти часов мы проводили вместе.

Это привело к тому, что эксперимент, начавшийся в осеннем семестре 2002–2003 учебного года, завершится (почти в полном составе от стартовавших) лишь в осеннем семестре следующего учебного года, когда появятся новые «экспериментаторы».

Перечислим некоторые проекты, реализованные или реализуемые в настоящее время:

- построение визуализаторов алгоритмов для обучения дискретной математике и программированию;
- автоматная реализация интерактивных сценариев образовательной анимации с использованием *Macromedia Flash*;
- обучающая и тестирующая программа с примером настройки для изучения английского языка;
- совместное использование теории построения компиляторов и автоматного программирования;
- скелетная анимация;
- XML-формат для описания внешнего вида видеопроигрывателя (www.crystalplayer.com);
- управление различными объектами (дизель-генератор, турникет, кодовый замок, банкомат, светофор, кофеварка, телефон, лифт и т. д.);
- система безопасности банка;
- клиент — серверные приложения;
- построение пользовательских интерфейсов;
- реализация сетевого протокола SMTP;
- классические «параллельные» задачи («Синхронизация цепи стрелков» и «Обедающие философы»);
- моделирование игры «Пятнашки» для робота «LEGO»;
- игры («Robocode» [21], «Terrarium», «CodeRally», «Морской бой», «Lines», «Automatic Bomber»,»Tron», «Однорукий бандит» и «Завалинка»).

Заметим при этом, что среди сотен танков для игры «Robocode», как отмечено в работе [24], только танк, разработанный нами, содержит проектную документацию. Такая же ситуация имеет место и для другой, еще более известной программистской игры — «Terrarium». Известны сотни созданных по правилам этой игры существ, но проектную документацию, похоже, разработали только мы. Из изложенного следует, что, если мир начал двигаться в сторону открытых исходных текстов, то можно предположить, что со временем будет разрабатываться и открытая проектная документация. Это позволит отказаться от мучительного чтения чужих программ, заменив его рассмотрением проектной документации, а когда захочется «пошевелить» мозгами, то вместо чтения программ можно будет, например, взяться за решение японских кроссвордов.

Первые выводы

Один из наших проектов после восьмой (что далеко не рекорд) корректировки (которая привела к тому, что мой соавтор-студент, подобно чеширскому коту, потерял улыбку), по моему мнению, можно было завершить: программа имела хороший пользовательский интерфейс, нормально работа-

ла и была разработана подробная, аккуратно выполненная проектная документация, содержащая в числе прочего и исходный код.

Последнее нас и «погубило», так как открытая проектная документация, как отмечалось выше, весьма наглядно демонстрирует не только достоинства, но и все недостатки программы. Один выдающийся программист (призер чемпионатов мира по программированию) взглянул на этот код, и студент вновь начал переделывать проект. Это (с перерывами) продолжается уже более полугода (выполнено еще несколько корректировок), и я надеюсь, что хотя бы в этом проекте (игра «Морской бой») все будет по-чеховски прекрасно: и лицо (пользовательский интерфейс), и одежда (документация), и душа (исходный код программы), и мысли (работа программы). Отметим, что все это происходит после того, как зачет по проекту был давно получен. На основе данного примера можно представить, как выглядят программы и документация на них, если они выполнены по лицензии *AS IS*.

Заключение

Высокая трудоемкость создания качественной проектной документации в программистском шоубизнесе вряд ли привьется. Эта технология является «тяжелой», в то время как сейчас все шире пропагандируются «легкие» и «шустрые» (agile) технологии [25], такие как, например, экстремальное программирование.

Однако есть и другие области программирования, в которых без «тяжелых» технологий не обойтись, и появляются новые люди, которым нравятся программы с хорошей проектной документацией. Один из студентов, впервые увидев проектную документацию, разработанную на основе предлагаемого подхода, воскликнул: «Она лучше, чем на телевизор! Видимо, она такая же, как для подводной лодки».

Даже если разработка проектной документации не привьется широко, то с точки зрения педагогики движение будет весьма полезным для его участников. Оно важно также с познавательной и эстетической сторон для тех, кто просто знакомится с проектами, даже в случае, если они сами ее документацию не разрабатывают — ведь не каждый посетитель музея пишет картины.

«Движение за открытую проектную документацию — это действительно то, чего разработчикам не хватает. К сожалению, практически вся документация в коммерческих проектах является собственностью заказчика, который не торопится ее обнародовать. Поэтому в Интернете так мало примеров реальных проектов. Сайтов с исходными текстами — море, а сайтов с примерами проектных решений — единицы. Я тут же залез в документацию в разделе «Проекты» вашего сайта и сравнил свои решения в аналогичных проектах с решениями, предложенными авторами. Жаль, что у меня не было доступа к этим текстам раньше. Я бы потратил на разработку меньше времени!» [26].

Работа выполняется при поддержке Российского фонда фундаментальных исследований по гранту № 02-07-90114 «Разработка технологии автоматного программирования». Вариант настоящего текста был ранее опубликован в журнале Мир ПК — ДИСК. 2003. № 8.

Литература

1. **Безруков Н.** Повторный взгляд на «собор» и «базар» //BYTE/Россия. — 2000. — № 8.
2. **Literate** Programming. — Stanford: Center for the Study of Language and Information, 1992.
3. **Lions J.** Lions' Commentary on UNIX: With Source Code. — Annabooks, 1977.
4. **Протасов П.** Снизу //Компьютерра. — 2003. — № 19.
5. **Исследователь** «руля» //Информатика. — 2003. — № 11.
6. **Блох А.** Закон Мерфи //ЭКО. — 1983. — № 1–3.
7. **Демин В.** Проблемы выхода российских разработчиков на Запад //PC WEEK/RE. — 2001. — № 32.
8. **Скрипников А.** Когда кончится нефть // «Известия». 09.09.2003.
9. **Бадд Т.** Объектно-ориентированное программирование. — СПб.: Питер, 1997.
10. **Буч Г., Рамбо Д., Джекобсон А.** UML. Руководство пользователя. — М.: ДМК, 2000.
11. **Фаулер М.** Новые методологии программирования //www.spin.org.ua.
12. **Буч Г.** Создание будущего // Подборка статей на тему «Программы следующего десятилетия» // «Открытые системы». — 2001. — № 12.
13. **Страуструп Б.** Язык программирования C++. — М.: Бином; СПб.: Невский диалект, 2001.
14. **Дейтел Х. М., Дейтел П. Дж.** Как программировать на C++. — М.: Бином, 2003.
15. **Гамма Э., Хелм Р., Джонсон Р., Влиссидис Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб.: Питер, 2001.
16. **Фаулер М.** Рефакторинг. — М.: Вильямс, 2003.
17. **Зайцев С. С.** Описание и реализация протоколов сетей ЭВМ. — М.: Наука, 1989.
18. **Аллен Э.** Типичные ошибки проектирования. — СПб.: Питер, 2003.
19. **Ершов А. П.** Смешанные вычисления // В мире науки. — 1984. — № 6.
20. **Тэллес М., Хсих Ю.** Наука отладки. — М.: КУДИЦ-ОБРАЗ, 2003.
21. **Шалыто А. А., Туккель Н. И.** Программирование с явным выделением состоянием // Мир ПК. — 2001. — № 8, 9. http://is.ifmo.ru, раздел «Статьи».
22. **Романовский И. В.** Дискретный анализ. — СПб.: БХВ-Петербург, Невский диалект, 2003.
23. **Шалыто А. А., Туккель Н. И.** Танки и автоматы // YTE/Россия. — 2003. — № 2. http://is.ifmo.ru, раздел «Статьи».
24. **Озеров А.** Четыре танкиста и компьютер // Магия ПК. — 2002. — № 11. http://is.ifmo.ru, раздел «О нас».
25. **Cockburn A.** Agile Software Development. — NJ: Addison-Wesley, 2001.
26. **Трофимов С.** info@caseclub.ru.