УДК 303.372.4

СИСТЕМНЫЙ АНАЛИЗ ДОКУМЕНТАЛЬНОГО СОПРОВОЖДЕНИЯ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Кочержинская Ю.В.

Аннотация. Область разработки программного обеспечения (ПО) – одна из самых динамично развивающихся в наши дни. Между тем, это уже сложившаяся индустрия со своими правилами, традициями, стандартами, наработками. Требования к актуальности создаваемого продукта теперь приравниваются по значимости к требованиям к созданию качественного продукта в определённый срок, не выходя за рамки бюджета проекта. Разработана и продолжает уточняться и пополняться нормативно-правовая база. Если раньше основу её составляли законы, охраняющие авторское право сторонних разработчиков, то сейчас они существенно переработаны в сторону защиты интеллектуальных и иных прав всех участников процесса создания и эксплуатации программных продуктов. Соответственно, пришёл более формальный подход и к самому процессу разработки ПО. Это потребовало упорядочивания в такой неизменно сопутствующей сфере, как документальное сопровождение процесса разработки. Однако именно в этой сфере наблюдается достаточно большая энтропия. С одной стороны, создание документации требует финансовых и человеческих ресурсов, что приводит к удорожанию любого проекта, с другой стороны – ещё сильны традиции неформального подхода большинства компаний-разработчиков ПО к процессу создания своей продукции. В статье сделана попытка системного подхода к анализу процессов документального сопровождения технологического цикла разработки программного обеспечения. Рассмотрены варианты создаваемых программных продуктов, документация к каждому из них и стадии технологического процесса, на которых она создается. Технологический процесс привязан к моделям жизненного цикла программного продукта, популярным при коллективной разработке ПО.

Ключевые слова: разработка программного обеспечения, документация, жизненный цикл, стандарты, виды программных продуктов, анализ документации.

SYSTEM ANALYSIS DOCUMENTARY SUPPORT SOFTWARE DEVELOPMENT PROCESS

Kocherzhinskaya Yu.V.

Abstract. Software development (software) – one of the most that dynamically growing today. Meanwhile, it has developed the industry with its own rules, traditions, standards, operating time. Requirements for the relevance of the created product is now equal in importance to the requirements for the creation of a quality product at a certain time, not going beyond the project budget. Developed and continues to change and be supplemented by the legal base. If before the foundation of its laws were copyrighted third-party developers, but now they are significantly redesigned in the direction of protection of intellectual and other rights of all participants in the process of creating and operating software. Accordingly, came a more formal approach to the process of software development. This necessitated the ordering in this field invariably concomitant as documentary support of the development process. However, it is large enough entropy observed in this area. On the one hand, the creation of the documentation required financial and human resources, which leads to a rise in the cost of any project, on the other hand – still a strong tradition of informal approach of most software companies in the process of creating their products. The paper attempts a systematic approach to the analysis of processes of documentary support of the technological cycle of software development. The variants of the software products, documentation for each of them and the stage of the process in which it is created. The technological process is tied to models of software product life cycle, popular with the collective software development.

Keywords: software development, documentation, life cycle, standards, types of software, documentation analysis.

Введение

Ещё в 1970 году Уокер Ройс (Walker Royce) отметил, что процессу разработки программного обеспечения присуща большая интеллектуальная свобода, которой необходимо управлять и которую нужно контролировать. Многие известные авторы в своих трудах [1-4] также отмечали, что главной проблемой процесса разработки программного обеспечения является именно гибкость самой отрасли программирования. На протяжении последних 50 лет идет спор о том, чем именно является процесс разработки ПО, чего в нём больше – искусства или конвейерного производства по типу создания самих вычислительных машин?

Анализируя этот спор можно отметить, что, во-первых, 50 лет – длительный процесс и за эти годы отрасль разработки ПО претерпела определенные изменения. Появились шаблоны, среды разработки, унифицированные языки создания проектов и моделей. Аналогично автомобильной промышленности, проводя параллели, можно констатировать, что если в начале 20 века каждый автомобиль был штучной продукцией, то в послевоенное время – уже доступной роскошью, что определило рост производства (в т.ч. конвейерного), а в последней четверти 20 века автомобиль и вовсе стал просто средством передвижения. Однако если бы все было настолько просто, все люди ездили бы на автомобилях одного «оптимального» с точки зрения динамики дизайна, все автомобили были бы окрашены в какой-нибудь «немар-

кий» цвет, имели средние технические характеристики и приблизительно одинаковую стоимость. Нечто подобное наблюдалось, когда происходило первичное насыщение рынка. Но ничего подобного не произошло и не происходит. Поскольку предпочтения потребителя оказывают существенное влияние на дизайн, а география и специфика условий использования отражают потребность в различных технических характеристиках. Сейчас можно наблюдать этот же процесс в отрасли разработки ПО – проникновение ЭВМ буквально во все сферы жизни вносит требование дифференциации программного обеспечения, обычный офисный пакет уже не в состоянии удовлетворить нужды потребителя, работающего в офисе. Специфика отрасли диктует потребность в специальных функциях, различных уровнях защиты и проч. Сейчас отрасль разработки ПО – это уже полноценная индустрия, имеющая свои технологические циклы, «заводы и фабрики» по изготовлению ПО и его частей, следующая экономическим и маркетинговым закономерностям.

Существование любой промышленной продукции немыслимо без документации. Любая аппаратура, программа или система окружена документами все время: от замысла до деинсталляции, на каждом этапе своего жизненного цикла. Невозможно представить себе создание хоть сколько-нибудь сложного технического решения без проектной документации, а применение – без технической и пользовательской документации. Информативная, полная, понятная, то есть, другими словами, качественная техническая документация – одно из существенных составляющих успеха программного продукта на всех этапах его жизненного цикла.

Документация на программное обеспечение – это документы, сопровождающие программное обеспечение – программу или программный продукт.

Традиционно выделяют четыре основных типа документации на ПО:

– проектная. Она содержит обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО;

– техническая. Это документация на код, алгоритмы, интерфейсы, *API*;

– пользовательская. В её состав входят руководства для конечных пользователей, администраторов системы и другого персонала;

– маркетинговая. Её основная задача – обеспечить внимание потенциальных пользователей и других заинтересованных лиц к разрабатываемому программному продукту.

Документирование – это важная часть в разработке программного обеспечения, но часто ей уделяется недостаточно внимания. Кратко рассмотрим, что входит в состав каждого вида документации.

Проектная документация обычно описывает продукт в общих чертах. Не описывая того, как что-либо будет использоваться, она скорее отвечает на вопрос «почему именно так?» Например, в проектном документе программист может описать обоснование того, почему структуры данных организованы именно таким образом. Описываются причины, почему какой-либо класс сконструирован определённым образом, выделяются паттерны, в некоторых случаях даже даются идеи, как можно будет выполнить улучшения в дальнейшем. Ничего из этого не входит в техническую или пользовательскую документацию, но всё это действительно важно для проекта.

Техническая документация – это именно то, что подразумевают под термином «*документация»* большинство разработчиков ПО. При создании программы, одного лишь кода, разумеется, недостаточно. Должен быть предоставлен некоторый текст на понятном языке, описывающий различные аспекты того, что именно делает тот или иной логически завершённый фрагмент кода. Такая документация часто включается непосредственно в исходный код или предоставляется вместе с ним.

Этот вид документации имеет сильно выраженный технический характер и в основном используется для определения и описания *API*, структур данных и алгоритмов.

Часто при составлении технической документации используются автоматизированные средства – генераторы документации. Они получают информацию из специальным образом оформленных комментариев в исходном коде, и создают справочные руководства в каком-

либо формате, например, в виде текста или *HTML*. Использование генераторов документации и документирующих комментариев многими программистами признаётся удобным средством, по различным причинам. В частности, при таком подходе документация является частью исходного кода, и одни и те же инструменты могут использоваться для сборки программы и одновременной сборки документации к ней. Это также упрощает поддержку документации в актуальном состоянии.

В отличие от технической документации, сфокусированной на коде и том, как он работает, пользовательская документация описывает то, как использовать готовый программный продукт. В случае, если продуктом является программная библиотека, пользовательская документация и документация на код становятся очень близкими, почти эквивалентными понятиями. Но в общем случае пользовательская документация представляет из себя руководство пользователя, которое описывает каждую функцию программы, а также шаги, которые нужно выполнить для использования этой функции. Хорошая пользовательская документация идёт ещё дальше и предоставляет инструкции о том, что делать в случае возникновения проблем. Очень важно, чтобы документация не вводила в заблуждение и была актуальной. Логическая связность и простота также имеют большое значение.

Существует три подхода к организации пользовательской документации. Вводное руководство (англ. *tutorial*), наиболее полезное для новых пользователей, последовательно проводит по ряду шагов, служащих для выполнения каких-либо типичных задач. Тематический подход, при котором каждая глава руководства посвящена какой-то отдельной теме, больше подходит для совершенствующихся пользователей. В последнем, третьем подходе, команды или задачи организованы в виде алфавитного справочника – часто это хорошо воспринимается продвинутыми пользователями, хорошо знающими, что они ищут. Жалобы пользователей обычно относятся к тому, что документация охватывает только один из этих подходов, и поэтому хорошо подходит лишь для одного класса пользователей.

Во многих случаях разработчики программного продукта ограничивают набор пользовательской документации лишь встроенной системой помощи (англ. *online help*), содержащей справочную информацию о командах или пунктах меню. Работа по обучению новых пользователей и поддержке совершенствующихся пользователей перекладывается на частных издателей, часто оказывающих значительную помощь разработчикам.

И, наконец, последний вид документации на ПО – маркетинговая документация. Для многих приложений необходимо располагать рядом рекламных материалов, с тем, чтобы заинтересовать людей, обратив их внимание на продукт. Такая форма документации имеет целью:

– подогреть интерес к продукту у потенциальных пользователей;

– информировать их о том, что именно делает продукт, с тем чтобы их ожидания совпадали с тем что они получат;

– объяснить положение продукта по сравнению с конкурирующими решениями.

Одна из хороших маркетинговых практик – предоставление «слогана» – простой запоминающейся фразы, иллюстрирующей то что мы хотим донести до пользователя, а также характеризующей *ощущение*, которое создаёт продукт.

Часто бывает так, что коробка продукта и другие маркетинговые материалы дают потенциальному пользователю более ясную картину о возможностях и способах использования программы, чем всё остальное [1-4].

Однако, для программистов, по понятным причинам, наибольший интерес представляет именно техническая документация. Именно она сопровождает процесс разработки программного обеспечения и именно с эти видом документов постоянно сталкиваются разработчики. В статье подробно анализируются особенности документирования процесса разработки применительно к различным моделям жизненного цикла программных продуктов.

Стандарты на документацию к программному обеспечению

Несмотря на изначально присущую отрасли разработки программного обеспечения космополитичность, среди отечественных разработчиков наибольшей популярностью на се-

годняшний день пользуются ГОСТы 34-й серии. Они включают три основных стандарта по документированию:

ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы

Самый популярный стандарт по разработке ТЗ. Он отличается внутренней логичностью структуры, широким, практически полным охватом аспектов, как процесса разработки, так и свойств и характеристик продукта. В пользу его качества говорит и то, что, несмотря на существенное изменение аппаратной составляющей и глобальное изменение целевого потребителя программной продукции, составление технического задания с его помощью является наиболее простым даже для специалиста, не обладающего квалификацией и опытом технического писателя.

Следующим стандартом серии является ГОСТ 34.201-89 Виды, комплектность и обозначения документов при создании автоматизированных систем. В нём приводится полный перечень документации ГОСТ 34, даны рекомендации по кодированию документов, распределению документов по стадиям проекта (стадии описываются в ГОСТ 34.601-90), а также по агрегации технических документов между собой.

Менее популярным отечественным стандартом на разработку технических документов является стандарт РД 50-34.698-90 Автоматизированные системы. Требования к содержанию документов. Он был введён в действие с первого января 1992 года. Он также ссылается на уже упомянутые стандарты ГОСТ 34.601, ГОСТ 34.602 и ГОСТ 34.201. В нём вводится понятие документации (ведомости) «эскизного проекта» и даются рекомендации по его выполнению. Кроме того, приведены требования к таким видам документации, как «Руководство пользователя», описанию проектных технических решений для монтажа автоматизированных систем и их компонентов и т.д. Несмотря на большой объём рекомендаций, этот стандарт не стал популярным ни среди разработчиков, ни среди заказчиков программного обеспечения, поскольку отличается неоднозначностью трактовок его положений, которые ввиду их расплывчатости, часто понимают по-разному разработчик и заказчик или даже члены команды разработчика.

Кроме того, следует упомянуть отечественный стандарт ГОСТ Р ИСО/МЭК ТО 9294-93 Информационная технология. Руководство по управлению документированием программного обеспечения. Он имеет более методическую направленность и содержит рекомендации по определению стратегий, стандартов, процедур, ресурсов и планов, которыми должны заниматься руководители проектов по разработке ПО («проектные менеджеры») для того, чтобы эффективно управлять процессами документирования программного обеспечения [5].

Международных стандартов, аналогичных по содержанию ГОСТам 34-й серии не существует. Однако, можно выделить некоторые стандарты, использование которых может помочь составлению грамотной документации на различных стадиях разработки ПО. В частности, следующие:

ISO/IEC 15289 Systems and software engineering – Content of systems and software life cycle process information products (Documentation) Системная и программная инженерия. Содержание информационных продуктов (документации) процессов жизненного цикла систем и программных средств.

ISO/IEC 26514 Software and systems engineering – User documentation requirements for documentation designers and developers Системная и программная инженерия. Требования для проектировщиков и разработчиков документации пользователя.

ISO/IEC 18019:2004 Software engineering – Guidelines for the design and preparation of user documentation for application software. Программная инженерия. Руководство по разработке и подготовке пользовательской документации на прикладные программные средства.

Стандарты на отдельные стадии процесса разработки и отдельные виды программных продуктов можно найти в каталоге *ISO* 35.080: Программное обеспечение.

Для обеспечения правильного ведения отдельных стадий жизненного цикла программных продуктов также существуют специально разработанные нормативные документы, такие как:

ISO/IEC/IEEE 29148-2011 Systems and software engineering – Life cycle processes – Requirements engineering. Системы и программная инженерия. Процессы жизненного цикла. Разработка и управление требованиями.
ISO/IEC/IEEE 42010-2011 Systems and software engineering – Architecture description. Системы и программная инженерия. Описание архитектуры.
ISO/IEC/IEEE 14764-2006 International Standard Software Engineering – Software Life Cycle Processes – Maintenance. Разработка программного обеспечения. Процессы жизненного цикла программного обеспечения. Сопровождение [6].
и многие другие.

Существуют отдельные стандарты на технические решения, семантику имён, разработку комментариев для систем открытой распределённой разработки, на такие части процессов разработки ПО, как определение функционального размера программного продукта, по семантике архитектуры, на оценку самих процессов разработки, по оцениванию и выбору *CASE*-средств, на гарантирование систем программного обеспечения, и даже руководства по принятию средств разработки [7-9]. Между тем, различные технические решения требуют и дифференцированного подхода к комплектованию технической документации (рис.1).

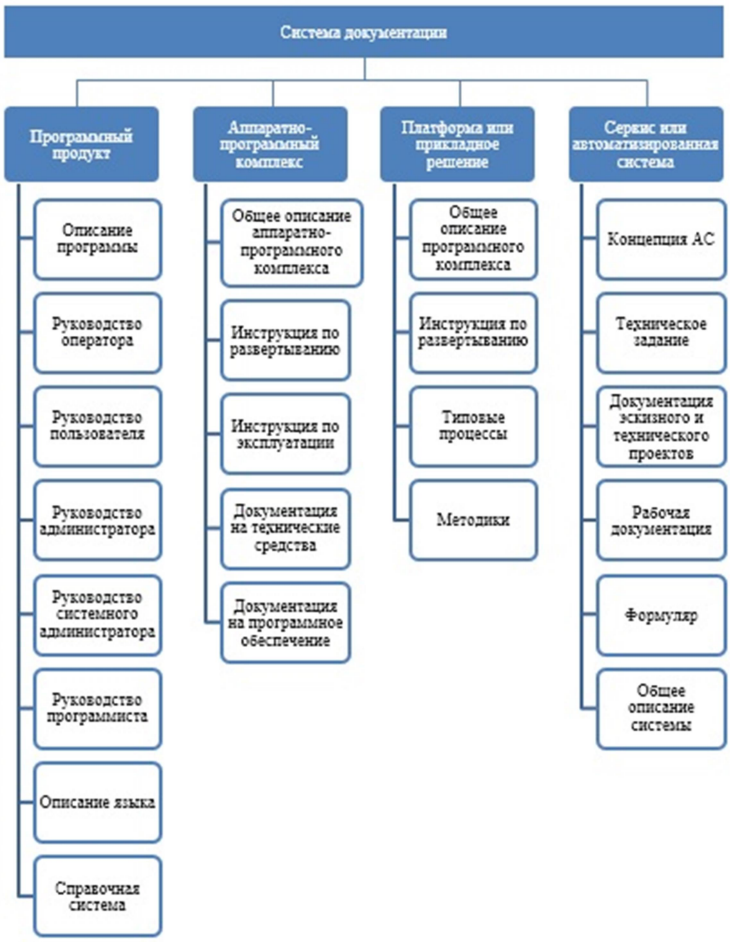

Рис. 1. Система документации различных видов ПО

Момент, когда начинается разработка каждого вида технической документации, её объём, потребность в актуализации и модернизации зависят от той модели жизненного цикла программного продукта, которая принята в организации-разработчике в целом или для конкретного проекта разработки.

Жизненный цикл ПО – период времени, за начало которого берется момент принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Жизненный цикл программного продукта соответствует

последовательности, устанавливаемой разработчиком той или иной моделью процесса разработки.

Исторически сложилось несколько стабильных моделей, имеющих авторство, однако, надо отметить, что в реальности они являются только каркасом, в который каждый автор (коллективный или индивидуальный) привносит своё наполнение. Рассмотрим их кратко.

Водопадная модель

Эту модель часть называют последовательной или каскадной. Автором её опубликования является Уокер Ройс. В 1970 году им был предложен следующий порядок этапов работ проекта по созданию ПО (см. рис. 2).

Рис. 2. Водопадная модель жизненного цикла ПО

Эта модель популярна до сегодняшнего дня прежде всего за свою предсказуемость с точки зрения калькуляции затрат времени и ресурсов на каждый из этапов проекта: строгая последовательность этапов исключает путаницу с документацией, поскольку по окончании каждого этапа появляется результирующий документ, который не подлежит изменениям, что обеспечивает стабильность работ на следующих этапах.

Спиралевидная модель

Спиралевидная модель была разработана в середине 1980-х годов Барри Боэмом (Barry Boehm). Она основана на классическом цикле Деминга (*Deming Cycle*) *PDCA* (plan-do-check-act). На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который удовлетворяет действительным требованиям заказчика и доводится до реализации.

Рис. 3. Спиралевидная модель жизненного цикла ПО

Спиралевидная модель относится в категории итерационных моделей. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу – как можно быстрее показать заказчику работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований (рис. 3).

У спиралевидной модели есть несколько существенных достоинств, например, конкурентоспособность продукта достаточно высокая. Однако поддержка документации в данной модели вызывает определённые трудности, особенно у молодых компаний-разработчиков, поскольку они часто испытывают проблемы с четким определением окончания стадий (вех) и, соответственно, с моментом, когда документация по проекту должна быть обновлена.

Инкрементальная модель

Эту модель также называют итерационной или эволюционной. Она предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом. Результат финальной итерации содержит всю требуемую функциональность продукта. Особенностью этой модели является синхронизированная с разработкой система документации. По окончании каждой стадии на каждой итерации появляется документация установленного образца – будь то спецификация требований или план управления программным проектом (рис.4).

Рис. 4. Инкрементальная модель жизненного цикла ПО

Она обладает рядом существенных достоинств, непосредственно касающихся разработки программной документации, в частности

1) изначально предполагает ведение актуальной и достоверной документации проекта;
2) разрабатываемый проект должен иметь четкую и правильную архитектуру;
3) основана на синхронизации всех частей проекта и постоянное тесное взаимодействие команды разработчиков [10].

Рациональная модель процесса разработки

Вершиной так называемых классических моделей процесса разработки программного обеспечения на сегодняшний день можно назвать *Rational Unified Process* (*RUP*) или просто Рациональную модель. Изначально она была разработана и поддерживается до сегодняшнего дня компанией *Rational Software*. Однако, со времени своей популяризации в конце 90-х, эта модель переросла в нечто большее, по сути, стала философией для разработчиков, серьёзно и методически подходящих к жизненному циклу процесса. Суть её составляет итерационный подход, при котором динамический процесс разработки разбивается на четыре фазы Начало, Проработка, Построение и Передача (*Inception-Elaboration-Construction-Transition*, *IECT*), каждая из которых состоит из итераций, позволяющих достичь хороших результатов. Статика рационального процесса неизменна и состоит из тех же классических стадий Сбора и анализа требований, Разработки архитектуры (Проектирования), Кодирования, Тестирования и Сопровождения. И прежде всего, это коллективный процесс разработки ПО.

Основными отличительными чертами этой модели являются:

1) очень внимательное отношение к выбору архитектуры и к стадии Проектирования;
2) использование визуального моделирования с применением *UML*;
3) включение концепции управления требованиями, что ведёт к большей предсказуемости процесса по линии качество-сроки исполнения;
4) использование Use Case-аппарата на всём протяжении работы над программным продуктом.

RUP предполагает работу с артефактами проекта. При этом, несмотря на строгий подход к стадиям процесса, один и тот же работник может совмещать несколько ролей в одном

проекте. *RUP* учитывает нагрузку на ту или иную роль в проекте. Один из адептов *RUP* Пер Кролл (*Per Kroll*) попытался в нескольких постулатах выразить основные принципы рационального процесса разработки ПО:
– атаковать риски как можно раньше, пока они сами не перешли в атаку;
– разрабатывать именно то, что нужно заказчику;
– главное внимание – исполняемой программе;
– приспосабливаться к изменениям с самого начала проекта;
– создавать архитектурный каркас как можно раньше;
– разрабатывать систему из компонентов;
– работать как одна команда;
– сделать качество стилем жизни [11-12].

На сегодняшний день *Rational Unified Process* можно назвать наиболее популярной методологией специалистов, придерживающихся классических процессов разработки ПО.

Проанализировав всё вышесказанное, можно составить следующую картину соответствия процессов жизненного цикла программного обеспечения создаваемой документации (рис. 5).

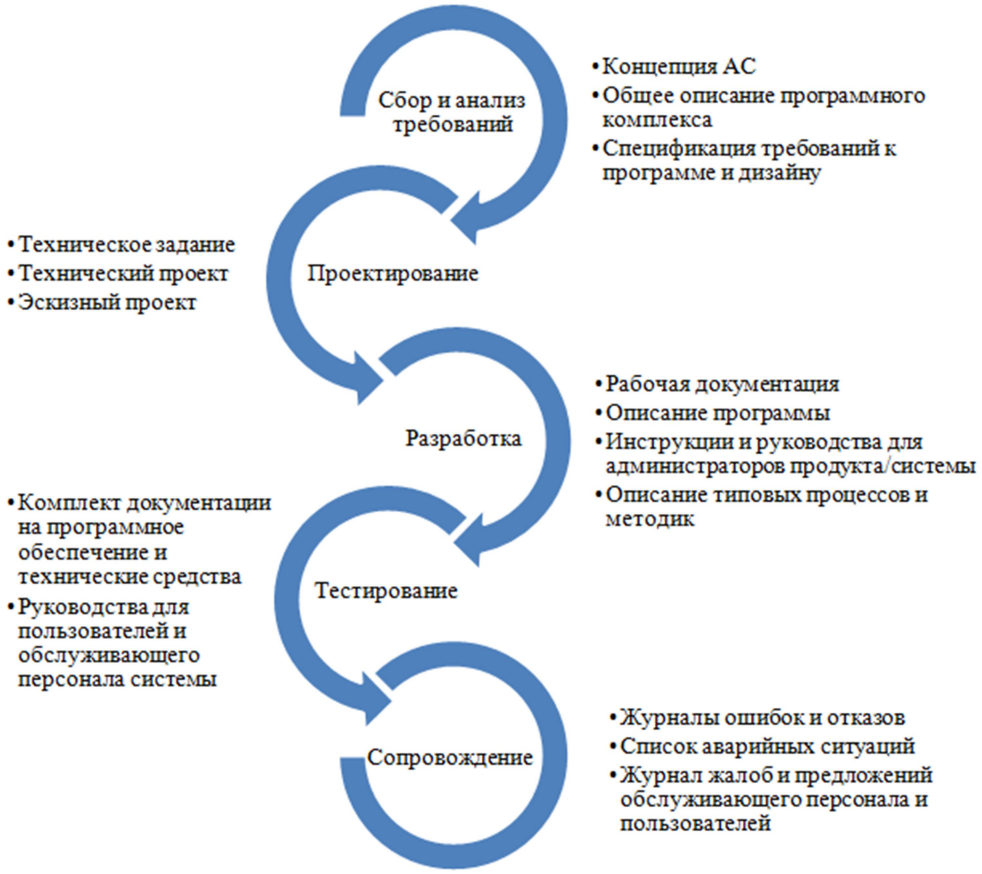

Рис. 5. Документация на различных этапах жизненного цикла ПО

Заключение

Создание качественного программного обеспечения, удовлетворяющего потребителя по стоимости и срокам невозможно без обширного документального сопровождения. Это тот случай, когда от полноты, точности и актуальности создаваемой документации зависит удовлетворённость конечного потребителя программной продукции, а также возможность дальнейшего сопровождения и перспективы её развития. Объёмы кода современных программ таковы, что даже имея в постоянной разработке некий программный продукт, специалист не в состоянии удержать в памяти дерево функций, особенности используемых методик расчёта, проблемные участки многотысячного кода. С другой стороны, широкое внедрение информационных технологий в официальные инстанции влечёт и соответствующие жесткие требования по качеству, защищенности, надёжности и другим важным характеристикам про-

граммного продукта. И в этом случае главным методом подтверждения заявленных качеств становятся, конечно же не слова разработчика и даже не его репутация, а документально зафиксированные источники и результаты разработки и всесторонней проверки программного продукта. Создание комплекта документов требует подготовки и специальных знаний как в сфере, к которой относится объект автоматизации, так и в области психологии, дизайна, программирования, стандартизации. То, какая именно документация создаётся зависит от вида создаваемого продукта, а на каких этапах разработки она будет создана – от используемой модели жизненного цикла. В глобальном мире соответствие программной продукции различным требованиям обеспечивается следованием различным международным стандартам ведения процесса разработки, содержания документации на процесс и результат.

В настоящей статье рассмотрены система документации на программное обеспечение, комплексы стандартов на этапы жизненного цикла и их документальное сопровождение, а также основные модели жизненного цикла программного обеспечения, используемые при коллективной и индивидуальной разработке и их влияние на объём производимой документации.

Список используемых источников

1. Рейнвотер, Дж. Ханк Как пасти котов. Наставление для программистов, руководящих другими программистами / Дж. Ханк Рейнвотер// СПб.: ПИТЕР, 2011. – 256 с.
2. Йордон, Э. Путь камикадзе. Как разработчику программного обеспечения выжить в безнадежном проекте /Э. Йордон// М.: Лори, 2012. – 256 с.
3. Нейгард, М. Release it! Проектирование и дизайн ПО для тех, кому не все равно/М. Нейгард// СПб.: ПИТЕР, 2016. – 320 с.
4. Самбук, А. Управление документацией в проектах разработки ПО // Открытые системы №7 2006 URL: http://www.osp.ru/os/2006/07/3290814/ (дата обращения: 20.02.2016).
5. Каталог стандартов // Росстандарт. Федеральное агенство по техническому регулированию и метрологии URL: http://www.gost.ru/wps/portal/pages.CatalogOfStandarts (дата обращения: 24.02.2016).
6. Publications and Standards // IEEE Xplore Digital Library URL: http://www.ieee.org/publications_standards/index.html (дата обращения: 20.03.2016).
7. Логунова О.С. Стандартизация и метрология программного обеспечения /О.С. Логунова, Е.А. Ильина, Н.С. Сибилева, А.Ю. Миков // Методические указания. – Магнитогорск: Изд-во МГТУ им. Г.И. Носова, 2016. – 24 с.
8. Глаголев, В.А. Разработка технической документации. Руководство для технических писателей и локализаторов ПО /В.А. Глаголев// СПб.: ПИТЕР, 2000. – 192 с.
9. Подход к оценке сроков создания технической документации // PhiloSoft Technical Communications URL: http://philosoft-services.com/metrics-idea.zhtml (дата обращения: 20.03.2016).
10. Брауде, Э. Технология разработки программного обеспечения /Э. Брауде // – СПб.: Питер, 2008. – 655 с.: ил.
11. Мак Коннел, С. Проект по разработке ПО. Руководство по выживанию: Библиотека программиста / С. Мак Коннел // – СПб.: Питер, 2006. –336 с.: ил.
12.Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха // СПб.: ПИТЕР, 2007. – 544 с.
13. Логунова, О.С. Человеко-машинное взаимодействие: Теория и практика / О.С. Логунова, И.М. Ячиков, Е.А. Ильина. – Ростов-на-Дону: Феникс, 2006. – 285 с.
14. Логунова, О.С. Структуризация лексикографической информации при разработке программного обеспечения / О.С. Логунова, Е.А. Ильина // МиПОС. – 2014. – № 1 (4). – С. 87-91.

Кочержинская Юлия Витальевна – канд. техн. наук, доцент кафедры вычислительной техники и программирования ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова». E-mail: Juliet@front.ru.

Кочержинская Ю.В. Системный анализ документального сопровождения процесса разработки программного обеспечения // Математическое и программное обеспечение систем в промышленной и социальной сферах. – 2016. – Т.4. – №1. – С. 33-41.

Kocherzhinskaya, Yu.V. (2016) System analysis documentary support software development process. Software of systems in the industrial and social fields, 4 (1): 33-41.