Не переходи мост, пока не дошел до него.

Народная пословица

Лекция 4. ВНЕШНЕЕ ОПИСАНИЕ ПРОГРАММНОГО СРЕДСТВА

Понятие внешнего описания, его назначение и роль в обеспечении качества программного средства. Определение требований к программному средству. Спецификация качества программного средства. Основные примитивы качества программного средства. Функциональная спецификация программного средства. Контроль внешнего описания.

4.1. Назначение внешнего описания программного средства и его роль в обеспечении качества программного средства.

Разработчикам больших программных средств приходится решать весьма специфические и трудные проблемы, особенно, если это ПС должно представлять собой программную систему нового типа, в плохо компьютеризированной предметной области. Разработка ПС начинается с процесса формулирования требований к ПС, в котором, исходя из довольно смутных пожеланий заказчика, должен быть создан документ, достаточно точно определяющий задачи разработчиков ПС. Этот документ мы называем *внешним описанием ПС*.

Очень часто требования к ПС путают с требованиями к процессам его разработки (технологическим процессам). Последние не следует включать во внешнее описание, если только они не связаны с оценкой качества ПС. В случае необходимости требования к технологическим процессам можно оформить в виде самостоятельного документа, который будет использоваться при управлении разработкой ПС.

Внешнее описание ПС играет роль точной постановки задачи, решение которой должно обеспечить разрабатываемое ПС. Более того, оно должно содержать всю информацию, которую необходимо знать пользователю для применения ПС. Оно является исходным документом для трех параллельно протекающих процессов: разработки текстов (конструированию и кодированию) программ, входящих в ПС, разработки документации по применению ПС и разработки существенной части комплекта тестов для тестирования ПС. Ошибки и неточности во внешнем описании, в конечном

счете, трансформируются в ошибки самой ПС и обходятся особенно дорого, во-первых, потому, что они делаются на самом раннем этапе разработки ПС, и, во-вторых, потому, что они распространяются на три параллельных процесса. Это требует принятия особенно серьезных мер по их предупреждению.

Исходным документом для разработки внешнего описания ПС является *определение требований* к ПС. Но так как через этот документ передается от заказчика (пользователя) к разработчику основная информация относительно требуемого ПС, то формирование этого документа представляет собой довольно длительный и трудный итерационный процесс взаимодействия между заказчиком и разработчиком, с которого и начинается этап разработки требований к ПС [4.2]. Трудности, возникающие в этом процессе, связаны с тем, что пользователи часто плохо представляют, что им на самом деле нужно: использование компьютера в «узких» местах деятельности пользователей может на самом деле потребовать принципиального изменения всей технологии этой деятельности (о чем пользователи, как правило, и не догадываются). Кроме того, проблемы, которые необходимо отразить в определении требований, могут не иметь определенной формулировки [4.1], что приводит к постепенному изменению понимания разработчиками этих проблем. В связи с этим определению требований часто предшествует процесс *системного анализа*, в котором выясняется, насколько целесообразно и реализуемо «заказываемое» ПС, как повлияет такое ПС на деятельность пользователей и какими особенностями оно должно обладать. Иногда бывает полезным разработка упрощенной версии требуемого ПС, называемую *прототипом* ПС. Анализ «пробного» применения прототипа позволяет выявить действительные потребности пользователей и существенно уточнить требования к ПС.

В определении внешнего описания легко бросаются в глаза две самостоятельные его части. Описание поведения ПС определяет функции, которые должна выполнять ПС, и потому его называют *функциональной спецификацией* ПС. Функциональная спецификация определяет допустимые фрагменты программ, реализующих декларированные функции. Требования к качеству ПС должны быть сформулированы так, чтобы разработчику были ясны цели [4.2], которые он должен стремиться достигнуть при разработке этого ПС. Эту часть внешнего описания будем называть *спецификацией качества* ПС (в литературе ее часто называют *нефункциональной спецификацией* [4.1], но она, как правило, включает и требования к технологическим процессам). Она, в отличие от функциональной спецификации, представляется в неформализованном виде и играет роль тех ориентиров, которые в значительной степени определяют

выбор подходящих альтернатив при реализации функций ПС, а также определяет стиль всех документов и программ требуемого ПС. Тем самым, спецификация качества играет решающую роль в обеспечении требуемого качества ПС.

Обычно разработка спецификации качества предшествует разработке функциональной спецификации ПС, так как некоторые требования к качеству ПС могут предопределять включение в функциональную спецификацию специальных функций, например, функции защиты от несанкционированного доступа к некоторым объектам информационной среды. Таким образом, структуру внешнего описания ПС можно выразить формулой:

Внешнее описание ПС = определение требований
+ спецификация качества ПС
+ функциональная спецификация ПС

Внешнее описание определяет, что должно делать ПС и какими внешними свойствами оно должно обладать. Оно не отвечает на вопросы, как обеспечить требуемые внешние свойства ПС и как это ПС должно быть устроено. Внешнее описание должно достаточно точно и полно определять задачи, которые должны решить разработчики требуемого ПС. В то же время оно должно быть понято представителем пользователем – на его основании заказчиком достаточно часто принимается окончательное решение на заключение договора на разработку ПС. Внешнее описание играет большую роль в обеспечении требуемого качества ПС, так как спецификация качества ставит для разработчиков ПС конкретные ориентиры, управляющие выбором приемлемых решений при реализации специфицированных функций.

4.2. Определение требований к программному средству.

Определение требований к ПС являются исходным документом разработки ПС – заданием, отражающим в абстрактной форме потребности пользователя. Они в общих чертах определяют замысел ПС, характеризуют условия его использования. Неправильное понимание потребностей пользователя трансформируются в ошибки внешнего описания. Поэтому разработка ПС начинается с создания документа, достаточно полно характеризующего потребности пользователя и позволяющего разработчику адекватно воспринимать эти потребности.

Определение требований представляет собой смесь фрагментов на естественном языке, различных таблиц и диаграмм. Такая смесь, должна быть понятной пользователю, не ориентирующемуся

в специальных программистских понятиях. Обычно в определении требований не содержится формализованных фрагментов, кроме случаев достаточно для этого подготовленных пользователей (например, математически) – формализация этих требований составляет содержание дальнейшей работы коллектива разработчиков.

Неправильное понимание требований заказчиком, пользователями и разработчиками связано обычно с различными взглядами на роль требуемого ПС в среде его использования [4.1]. Поэтому важной задачей при создании определения требований является установление контекста использования ПС, включающего связи между этим ПС, аппаратурой и людьми. Лучше всего этот контекст в определении требований представить в графической форме (в виде диаграмм) с добавлением описаний сущностей используемых объектов (блоков ПС, компонент аппаратуры, персонала и т.п.) и характеристики связей между ними.

Известны три способа разработки определения требований к ПС [4.2]:

- управляемая пользователем разработка,
- контролируемая пользователем разработка,
- независимая от пользователя разработка.

В *управляемой пользователем* разработке определения требований к ПС определяются заказчиком, представляющим организацию пользователей. Это происходит обычно в тех случаях, когда организация пользователей (заказчик) заключает договор на разработку требуемого ПС с коллективом разработчиков и требования к ПС являются частью этого договора. Роль разработчика ПС в создании этих требований сводится, в основном, в выяснении того, насколько понятны ему эти требования с соответствующей критикой рассматриваемого документа. Это может приводить к созданию нескольких редакций этого документа в процессе заключения указанного договора.

В *контролируемой пользователем* разработке требования к ПС формулируются разработчиком при участии представителя пользователей. Роль пользователя в этом случае сводится к информированию разработчика о своих потребностях в ПС, а также к контролю того, чтобы формулируемые требования действительно выражали его потребности в ПС. Разработанные требования, как правило, утверждаются представителем пользователя.

В *независимой от пользователя* разработке требования к ПС определяются без какого-либо участия пользователя (на полную ответственность разработчика). Это происходит обычно тогда, когда разработчик решает создать ПС широкого применения в расчете на

то, разработанное им ПС найдет спрос на рынке программных средств.

С точки зрения обеспечения надежности ПС наиболее предпочтительным является контролируемая пользователем разработка.

4.3. Спецификация качества программного средства.

Разработка спецификации качества сводится, по существу, к построению своеобразной модели качества требуемого ПС [4.2, 4.3]. В этой модели должен быть перечень всех тех достаточно элементарных свойств, которые необходимо обеспечить в требуемом ПС и которые в совокупности образуют приемлемое для пользователя качество ПС. При этом каждое из этих свойств должно быть в достаточной степени конкретизировано с учетом определения требований к ПС и возможности оценки его наличия у разработанного ПС или необходимой степени обладания им этим ПС.

Для конкретизации качества ПС по каждому из критериев используется стандартизованный набор достаточно простых свойств ПС [4.3-4.6], однозначно интерпретируемых разработчиками. Такие свойства мы будем называть *примитивами качества* ПС. Некоторые из примитивов могут использоваться по нескольким критериям. Ниже приводится зависимость критериев качества от примитивов качества ПС.

Функциональность: завершенность.

Надежность: завершенность, точность, автономность, устойчивость, защищенность.

Легкость применения: П-документированность, информативность (только применительно к документации по применению), коммуникабельность, устойчивость, защищенность.

Эффективность: временна́я эффективность, эффективность по ресурсам (по памяти), эффективность по устройствам.

Сопровождаемость. С данным критерием связано много различных примитивов качества. Однако их можно распределить по двум группам, выделив два подкритерия качества: изучаемость и модифицируемость. *Изучаемость* – это характеристики ПС, которые позволяют минимизировать усилия по изучению и пониманию программ и документации ПС. *Модифицируемость* – это характеристики ПС, которые позволяют автоматически настраивать на условия применения ПС или упрощают внесение в него вручную необходимых изменений и доработок.

Изучаемость: С-документированность, информативность (здесь применительно к документации по сопровождению), понятность, структурированность, удобочитаемость.

Модифицируемость: расширяемость, модифицируемость (в узком смысле, как примитив качества), структурированность, модульность.

Мобильность: независимость от устройств, автономность, структурированность, модульность.

Ниже даются определения используемых примитивов качества ПС [4.3-4.5].

Завершенность (*completeness*) – свойство, характеризующее степень обладания ПС всеми необходимыми частями и чертами, требующимися для выполнения своих явных и неявных функций.

Точность (*accuracy*) – мера, характеризующая приемлемость величины погрешности в выдаваемых программами ПС результатах с точки зрения предполагаемого их использования.

Автономность (*self-containedness*) – свойство, характеризующее способность ПС выполнять предписанные функции без помощи или поддержки других компонент программного обеспечения.

Устойчивость (*robustness*) – свойство, характеризующее способность ПС продолжать корректное функционирование, несмотря на задание неправильных (ошибочных) входных данных.

Защищенность (*defensiveness*) – свойство, характеризующее способность ПС противостоять преднамеренным или нечаянным деструктивным (разрушающим) действиям пользователя.

П-документированность (*и. documentation*) – свойство, характеризующее наличие, полноту, понятность, доступность и наглядность учебной, инструктивной и справочной документации, необходимой для применения ПС.

Информативность (*accountability*) – свойство, характеризующее наличие в составе ПС информации, необходимой и достаточной для понимания назначения ПС, принятых предположений, существующих ограничений, входных данных и результатов работы отдельных компонент, а также текущего состояния программ в процессе их функционирования.

Коммуникабельность (*communicativeness*) – свойство, характеризующее степень, в которой ПС облегчает задание или описание входных данных, и способность выдавать полезные сведения в достаточно простой форме и с простым для понимания содержанием.

Временна́я эффективность (*time efficiency*) – мера, характеризующая способность ПС выполнять возложенные на него функции в течение определенного отрезка времени.

Эффективность по ресурсам (*resource efficiency*) – мера, характеризующая способность ПС выполнять возложенные на него функ-

ции при определенных ограничениях на используемые ресурсы (используемую память).

Эффективность по устройствам (*device efficiency*) – мера, характеризующая экономичность использования устройств машины для решения поставленной задачи.

С-документированность (*documentation*) – свойство, характеризующее с точки зрения наличия документации, отражающей требования к ПС и результаты различных этапов разработки данного ПС, включающие возможности, ограничения и другие черты ПС, а также их обоснование.

Понятность (*understandability*) – свойство, характеризующее степень, в которой ПС позволяет изучающему его лицу понять его назначение, сделанные допущения и ограничения, входные данные и результаты работы его программ, тексты этих программ и состояние их реализации. Этот примитив качества синтезирован нами из таких примитивов ИСО [4.4], как согласованность, самодокументированность, четкость и, собственно, понятность (текстов программ).

Структурированность (*structuredness*) – свойство, характеризующее программы ПС с точки зрения организации взаимосвязанных их частей в единое целое определенным образом (например, в соответствии с принципами структурного программирования).

Удобочитаемость (*readability*) – свойство, характеризующее легкость восприятия текста программ ПС (отступы, фрагментация, форматированность).

Расширяемость (*augmentability*) – свойство, характеризующее способность ПС к использованию большего объема памяти для хранения данных или расширению функциональных возможностей отдельных компонент.

Модифицируемость (*modifiability*) – мера, характеризующая ПС с точки зрения простоты внесения необходимых изменений и доработок на всех этапах и стадиях жизненного цикла ПС.

Модульность (*modularity*) – свойство, характеризующее ПС с точки зрения организации его программ из таких дискретных компонент, что изменение одной из них оказывает минимальное воздействие на другие компоненты.

Независимость от устройств (*device independence*) – свойство, характеризующее способность ПС работать на разнообразном аппаратном обеспечении (различных типах, марках, моделях компьютеров).

4.4. Функциональная спецификация программного средства.

С учетом назначения функциональной спецификации ПС и тяжелых последствий неточностей и ошибок в этом документе, функциональная спецификация должна быть математически точной. Это не означает, что она должна быть формализована настолько, что по ней можно было бы автоматически генерировать программы, решающие поставленную задачу. А означает лишь, что она должна базироваться на понятиях, построенных как математические объекты, и утверждениях, однозначно понимаемых разработчиками ПС. Достаточно часто функциональная спецификация формулируется на естественном языке. Тем не менее, использование математических методов и формализованных языков при разработке функциональной спецификации весьма желательно, поэтому этим вопросам будет посвящена отдельная лекция.

Функциональная спецификация состоит из трех частей:

- описания внешней информационной среды, к которой должны применяться программы разрабатываемой ПС;
- определение функций ПС, определенных на множестве состояний этой информационной среды (такие функции будем называть *внешними функциями* ПС);
- описание нежелательных (исключительных) ситуаций, которые могут возникнуть при выполнении программ ПС, и реакций на эти ситуации, которые должны обеспечить соответствующие программы.

В первой части должны быть определены на концептуальном уровне все используемые каналы ввода и вывода и все информационные объекты, к которым будет применяться разрабатываемое ПС, а также существенные связи между этими информационными объектами. Примером описания информационной среды может быть концептуальная схема базы данных или описание сети датчиков и приборов, которой должна управлять разрабатываемая ПС.

Во второй части вводятся обозначения всех определяемых функций, специфицируются все входные данные и результаты выполнения каждой определяемой функции, включая указание их типов и заданий всех соотношений (или ограничений), которым должны удовлетворять эти данные и результаты. И, наконец, определяется семантика каждой из этих функций, что является наиболее трудной задачей функциональной спецификации ПС. Обычно эта семантика описывается неформально на естественном языке – примерно так, как это делается при описании семантики многих языков программирования. Эта задача может быть в ряде случаев существенно облегчена при достаточно четком описании внешней информацион-

ной среды, если внешние функции задают какие-либо манипуляции с ее объектами.

В третьей части должны быть перечислены все существенные случаи, когда ПС не сможет нормально выполнить ту или иную свою функцию (с точки зрения внешнего наблюдателя). Примером такого случая может служить обнаружение ошибки во время взаимодействия с пользователем, или попытка применить какую-либо функцию к данным, не удовлетворяющим соотношениям, указанным в ее спецификации, или получение результата, нарушающего заданное ограничение. Для каждого такого случая должна быть определена (описана) реакция ПС.

4.5. Методы контроля внешнего описания программного средства.

Разработка внешнего описания обязательно должна завершаться проведением тщательного и разнообразного контроля правильности внешнего описания. Целью этого процесса является найти как можно больше ошибок, сделанных на этом этапе. Учитывая, что результатом этого этапа является, как правило, еще неформализованный текст, то здесь на первый план выступают психологические факторы контроля. Можно выделить следующие методы контроля, применяемые на этом этапе:

- статический просмотр,
- смежный контроль,
- пользовательский контроль,
- ручная имитация.

Первый метод предполагает внимательное прочтение текста внешнего описания разработчиком с целью проверка его полноты и непротиворечивости, а также выявления других неточностей и ошибок.

Смежный контроль спецификации качества сверху – это ее проверка со стороны разработчика требований к ПС, а смежный контроль функциональной спецификации – это ее проверка разработчиками требований к ПС и спецификации качества. Смежный контроль внешнего описания снизу – это его изучение и проверка разработчиками архитектуры ПС и текстов программ, а также разработчиками документации по применению и разработчиками комплекта тестов.

Пользовательский контроль внешнего описания выражает участие пользователя (заказчика) в принятии решений при разработке внешнего описания и его контроле. Если разработка требований к ПС велась под управлением пользователя, то пользовательский контроль внешнего описания, по существу, означает его смежный

контроль сверху. Однако, если представителю пользователя оказывается трудно самостоятельно разобраться во внешнем описании, создается специальная группа разработчиков, выполняющая роль пользователя (и взаимодействующая с ним) для проведения такого контроля.

Ручная имитация выражает своеобразный динамический контроль внешнего описания, точнее говоря, функциональной спецификации ПС. Для этого необходимо подготовить исходные данные (тесты) и на основании функциональной спецификации осуществить имитацию поведения (работы) разрабатываемого ПС. При этом такую имитацию осуществляет специально назначенный разработчик, выполняющий, по существу, роль будущих программ ПС. Разновидностью такого контроля является имитация за терминалом. В этом случае данные вводятся в компьютер человеком, играющего роль пользователя, и передаются с помощью несложной программы на другой терминал, за которым сидит разработчик, выполняющий роль программ ПС. Полученные результаты передаются через компьютер на первый терминал.

Упражнения к лекции 4.

4.1. Что такое определение требований к программному средству (ПС)?
4.2. Что такое спецификации качества ПС?
4.3. Что такое устойчивость (*robustness*) *ПС?*
4.4. Что такое защищенность (*defensiveness*) *ПС?*
4.5. Что такое коммуникабельность (*communicativeness*) *ПС?*
4.6. Что такое функциональная спецификация ПС?
4.7. Что такое ручная имитация внешнего описания ПС?

Литература к лекции 4.

[4.1] Ian Sommerville. Software Engineering. — Addison-Wesley Publishing Company, 1992.
[4.2] Г. Майерс. Надежность программного обеспечения. — М.: Мир, 1980. — С. 49-77.
[4.3] Е.А. Жоголев. Введение в технологию программирования (конспект лекций). — М.: «ДИАЛОГ-МГУ», 1994.
[4.4] Criteria for Evaluation of Software. ISO TC97/SC7 #383.
[4.5] Revised version of DP9126 — Criteria of the Evaluation of Software Quality Characteristics. ISO TC97/SC7 #610. — Part 6.
[4.6] Б. Боэм, Дж. Браун, Х. Каспар и др. Характеристики качества программного обеспечения. — М.: Мир, 1981. — С. 61-87.