# Лабораторная работа 3.5 Функциональные и нефункциональные требования

*(Не оценивается отдельно, является следствием из первых 3 лаб в курсовой работе)*

**Задача:**

Составить полные списки функциональных и нефункциональных требований к вашей системе.

**Теория:**

Функциональные и нефункциональные требования (Functional and Non-functional Requirements)

Требования к программной системе часто классифицируются как функциональные, нефункциональные и требования предметной области.

**Функциональные требования** задают “что” система должна делать; **нефункциональные** – с соблюдением “каких условий” (например, скорость отклика при выполнении заданной операции); часто функциональные требования представляют в виде сценариев (вариантов использования) Use Сase.

1. Функциональные требования. Это перечень сервисов, которые должна выполнять система, причем должно быть указано, как система реагирует на те или иные вход­ные данные, как она ведет себя в определенных ситуациях и т.д. В некоторых слу­чаях указывается, что система не должна делать.
2. Нефункциональные требования. Описывают характеристики системы и ее окружения, а не поведение системы. Здесь также может быть приведен перечень ограничений, накладываемых на действия и функции, выполняемые системой. Они включают временные ограничения, ограничения на процесс разработки системы, стандарты и тд.
3. Требования предметной области. Характеризуют ту предметную область, где будет эксплуатироваться система. Эти требования могут быть функциональными и не­функциональными.

В действительности четкой границы между этими типами требований не существует. Например, пользовательские требования, касающиеся безопасности системы, можно отнести к нефункциональным. Однако при более детальном рассмотрении такое требование можно отнести к функциональным, поскольку оно порождает необходимость включения в систему средства авторизации пользователя. Поэтому, рассматривая далее эти виды требований, мы должны всегда помнить, что данная классификация в значительной степени искусственна.

Классический пример (см. рисунок 4.3) высокоуровневого структурирования групп требований как требований к продукту описан в работах одного из классиков дисциплины управления требованиями – Карла Вигерса.

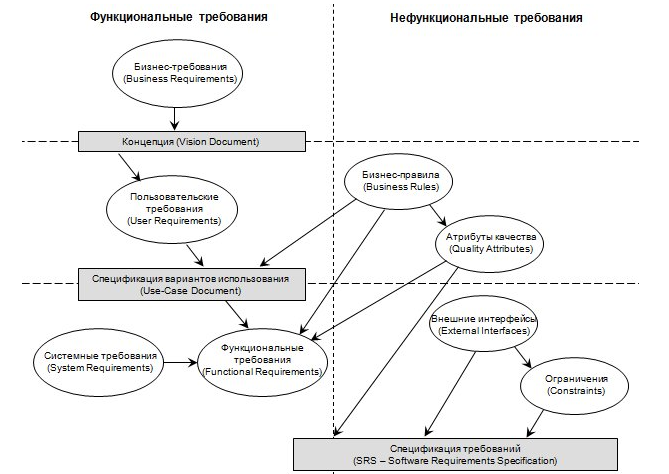


Рисунок 4.3. Уровни требований по Вигерсу

* *Группа функциональных требований*
  + *Бизнес-требования (Business Requirements)* – определяют высокоуровневые цели организации или клиента (потребителя) – заказчика разрабатываемого программного обеспечения.
  + *Пользовательские требования (User Requirements)* – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы. Эти требования часто представляют в виде *вариантов использования (Use Cases)*.
  + *Функциональные требования (Functional Requirements)* – определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.
* *Группа нефункциональных требований (Non-Functional Requirements)*
  + *Бизнес-правила (Business Rules)* – включают или связаны с корпоративными регламентами, политиками, стандартами, законодательными актами, внутрикорпоративными инициативами (например, стремление достичь зрелости процессов по CMMI 4-го уровня), учетными практиками, алгоритмами вычислений и т.д. На самом деле, достаточно часто можно видеть недостаточное внимание такого рода требованиям со стороны сотрудников ИТ-департаментов и, в частности, технических специалистов, вовлеченных в проект. Business Rules Group дает понимание *бизнес-правила,* как “положения, которые определяют или ограничивают некоторые аспекты бизнеса. Они подразумевают организацию структуры бизнеса, контролируют или влияют на поведение бизнеса”. Бизнес-правила часто определяют распределение ответственности в системе, отвечая на вопрос “кто будет осуществлять конкретный вариант, сценарий использования” или диктуют появление некоторых функциональных требований. *В контексте дисциплины управления проектами* (уже вне проекта разработки программного обеспечения, но выполнения бизнес-проектов и бизнес-процессов) такие правила обладают высокой значимостью и, именно они, часто определяют ограничения бизнес-проектов, для автоматизации которых создается соответствующее программное обеспечение.
  + *Внешние интерфейсы (External Interfaces)* – часто подменяются “пользовательским интерфейсом”. На самом деле вопросы организации пользовательского интерфейса безусловно важны в данной категории требований, однако, конкретизация аспектов взаимодействия с другими системами, операционной средой (например, запись в журнал событий операционной системы), возможностями мониторинга при эксплуатации – все это не столько функциональные требования (к которым ошибочно приписывают иногда такие характеристики), сколько вопросы интерфейсов, так как функциональные требования связаны непосредственно с *функциональностью* системы, направленной на решение *бизнес-потребностей*.
  + *Атрибуты качества (Quality Attributes)* – описывают дополнительные характеристики продукта в различных “измерениях”, важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности, устойчивости и т.п.
  + *Ограничения (Constraints)* – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных, ...), которые, в свою очередь, могут относиться, например, к внешним интерфейсам.
* *Системные требования (System Requirements)* – иногда классифицируются как составная часть группы функциональных требований (не путайте с как таковыми “функциональными требованиями”). Описывают высокоуровневые требования к программному обеспечению, содержащему несколько или много взаимосвязанных подсистем и приложений. При этом, система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть персонал, выполняющий определенные функции *системы*, например, авторизация выполнения определенных операций с использованием программно-аппаратных подсистем.

# Функциональные требования

Эти требования описывают поведение системы и сервисы (функции), которые она выполняет, и зависят от типа разрабатываемой системы и от потребностей пользовате­лей. Если функциональные требования оформлены как пользовательские, они, как прави­ло, описывают системы в обобщенном виде. В противоположность этому функциональ­ные требования, оформленные как системные, описывают систему максимально подроб­но, включая ее входные и выходные данные, исключения и т.д.

Функциональные требования для программных систем могут быть описаны разными способами. Рассмотрим для примера функциональные требования к библиотечной системе университета, предназначенной для заказа книг и документов из других библиотек.

1. Пользователь должен иметь возможность проводить поиск необходимых ему книг и документов или по всему множеству доступных каталожных баз данных или по определенному их подмножеству.
2. Система должна предоставлять пользователю подходящее средство просмотра библиотечных документов.
3. Каждый заказ должен быть снабжен уникальным идентификатором (ORDERJD), который копируется в формуляр пользователя для постоянного хранения.

Эти функциональные пользовательские требования определяют свойства, которыми должна обладать система. Они взяты из документа, содержащего пользовательские требования, и показывают, что функциональные требования могут быть описаны с разным уровнем детализации (сравните первое и третье требования).

Многие проблемы, возникающие при разработке систем, связаны с неточностью и "размытостью" спецификации требований. Естественно, разработчики интерпретируют требования, допускающие двоякое толкование, так, чтобы систему было проще реализо­вать. Но это толкование может не совпадать с ожиданиями заказчика. Такая ситуация приводит к разработке новых требований и внесению изменений в систему. Это, в свою очередь, ведет к задержке сдачи готовой системы и ее удорожанию.

Рассмотрим второе требование к библиотечной системе из приведенного выше списка и обратим внимание на выражение "подходящее средство просмотра документов". Библио­течная система может предоставлять документы в широком спектре форматов. В требова­нии подразумевается, что система должна предоставить средства для просмотра документов в любом формате. Но поскольку это условие четко не выписано, разработчики в случае де­фицита времени могут использовать простое средство для просмотра текстовых документов и настаивать на том, что именно такое решение следует из данного требования.

В принципе спецификация функциональных требований должна быть комплексной и непротиворечивой. Комплексность подразумевает описание (определение) всех систем­ных сервисов. Непротиворечивость означает отсутствие несовместимых и взаимоисклю­чающих определений сервисов. На практике для больших и сложных систем крайне труд­но разработать комплексную и непротиворечивую спецификацию функциональных тре­бований. Причина кроется частично в сложности самой разрабатываемой системы, а частично — в несогласованных опорных точках зрения на то, что должна де­лать система. Эта несогласованность может не проявиться на этапе первоначального формулирования требований — для ее выявления необходим более глубокий анализ спе­цификации. Когда несогласованность системных функций проявится на каком-либо этапе жизненного цикла программы, в системную спецификацию придется внести соответст­вующие изменения.

**Нефункциональные требования**

Как следует из названия, нефункциональные требования не связаны непосредственно с функциями, выполняемыми системой. Они связаны с такими интеграционными свойствами системы, как надежность, время ответа или размер системы. Кроме того, нефункциональные требования могут определять ограничения на систему, например на пропускную способность устройств ввода-вывода, или форматы данных, используемых в системном интерфейсе.

Многие нефункциональные требования относятся к системе в целом, а не к отдельным ее средствам. Это означает, что они более значимы и критичны, чем отдельные функцио­нальные требования. Ошибка, допущенная в функциональном требовании, может снизить качество системы, ошибка в нефункциональных требованиях может сделать систему не­работоспособной.

Вместе с тем нефункциональные требования могут относиться не только к самой программной системе: одни могут относиться к технологическому процессу создания ПО, другие — содержать перечень стандартов качества, накладываемых на процесс разработки. Кроме того, в спецификации нефункциональных требований может быть указано, что проектирование системы должно выполняться только определенными CASE-средствами, и приведено описание процесса проектирования, которому необходимо следовать.

Нефункциональные требования отображают пользовательские потребности; при этом они основываются на бюджетных ограничениях, учитывают организационные возможности компании-разработчика и возможность взаимодействия разрабатываемой системы с другими программными и вычислительными системами, а также такие внешние факторы, как правила техники безопасности, законодательство о защите интеллектуальной собственности и т.п. На рис. 4.4 показана классификация нефункциональных требований.

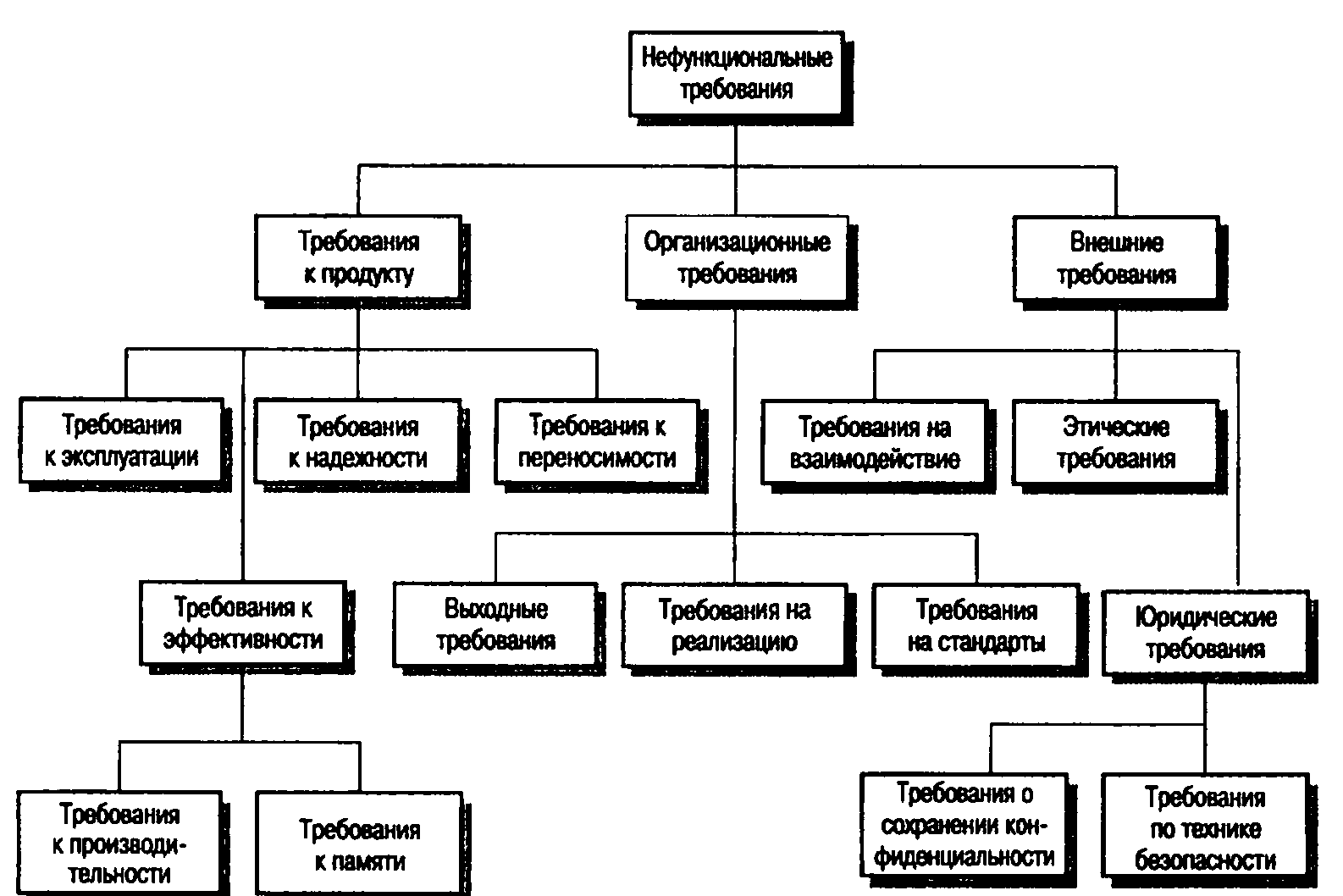


Рис. 4.4. Типы нефункциональных требований

Все нефункциональные требования, показанные на рис.4.4. разбиты на три большие группы.

1. Требования к продукту. Описывают эксплуатационные свойства программного продукта. Сюда относятся требования к производительности системы, объему необходимой памяти, надежности (определяет частоту возможных сбоев в системе), переносимости системы на разные компьютерные платформы и удобству эксплуатации.
2. Организационные требования. Отображают политику и организационные процедуры заказчика и разработчика ПО. Они включают стандарты разработки программного продукта, требования к реализации ПО (т.е. к языку программирования и методам проектирования), выходные требования, которые определяют сроки изготовления программного продукта, и сопутствующую документацию.
3. Внешние требования. Учитывают факторы, внешние по отношению к разрабатываемой системе и процессу ее разработки. Они включают требования, определяющие взаимодействие данной системы с другими системами, юридические требования, следование которым гарантирует, что система будет разрабатываться и функционировать в рамках существующего законодательства, а также этические требования. Последние должны гарантировать, что система будет приемлемой для пользователей или заказчика.

Основная проблема нефункциональных требований состоит в том, что их выполнение трудно проверить. Часто они пишутся для того, чтобы отобразить общие цели заказчика системы, такие, как простота эксплуатации, возможность восстановления после сбоев или быстрый ответ на запросы пользователя. Реализация подобных требований может оказаться сложной для системных разработчиков, поскольку они нечетко сформулированы и открывают простор для различных толкований. Подобную ситуацию иллюстрирует пример 2. Здесь одним из основных показателей (целей) системы указана простота эксплуатации, что в виде нефункциональных требований можно выразить различными способами. В данном случае требование сформулировано так, что его можно проверить.

Пример 2. Системные цели и проверка требований.

**Системная цель.**

Система должна быть простой, в эксплуатации для опытного оператора и сводить количество его ошибок к минимуму.

**Проверяемое нефункциональное требование.**

Опытному оператору должны быть доступны все системные функции после двух часов обучения работе с данной системой. После такого обучения число ошибок оператора не должно превышать двух за рабочий день.

В идеале нефункциональные требования должны выражаться через количественные показатели, которые можно объективно измерить. В табл. 4.1 приведены показатели, с помощью которых можно специфицировать нефункциональные системные свойства. На практике выразить нефункциональные требования с помощью количественных по­казателей весьма затруднительно. Часто заказчик ПО не может оформить свое видение будущей системы посредством требований, выраженных количественными показателями. Либо некоторые системные требования, например удобство сопровождения, вообще нельзя выразить через количественные показатели. Кроме того, затраты на объективное измерение количественных нефункциональных требований могут оказаться крайне высокими.

Нефункциональные требования часто вступают в конфликт с другими требованиями, предъявляемыми системе. Например, в соответствии с одним из системных требований размер системы не должен превышать 4 Мбайт, поскольку она должна полностью поместиться в постоянное запоминающее устройство ограниченной емкости. Другое требование обязывает использовать для написания системы язык программирования Ada, который часто применяется для создания критических систем реального времени. Но, допустим, откомпилированная системная программа, написанная на языке Ada, занимает более 4 Мбайт. Итак, одновременное выполнение этих требований невозможно. В этой ситуации следует отказаться от одного из требований. Можно или применить другой язык программирования, или увеличить объем памяти, выделяемый для системы.

Таблица 4.1. Количественные показатели нефункциональных требований

|  |  |
| --- | --- |
| Показатель | Единицы измерения |
| Скорость | Количество выполненных транзакций в секунду; |
|  | время реакции на действия пользователя; |
|  | время обновления экрана |
| Размер | Килобайты; |
|  | количество модулей памяти |
| Простота эксплуатации | Время обучения персонала; |
|  | количество статей в справочной системе |
| Надежность | Средняя продолжительность времени между двумя последовательными проявлениями ошибок в системе;  вероятность выхода системы из строя;  коэффициент готовности системы |
| Устойчивость к сбоям | Время восстановления системы после сбоя; процент событий, приводящих к сбоям; вероятность порчи данных при сбоях |
| Переносимость | Процент машинно-зависимых операторов; |
|  | количество машинно-зависимых подсистем |