

Функциональное проектирование

Технологическая
специализация системный
аналитик



Оглавление

Введение	2
Термины, используемые в лекции	2
Роль и место функционального проектирования	3
Описание вариантов использования системы	5
Варианты использования и пользовательские роли визуализируются в графическом виде.	8
Общие рекомендации по составлению диаграммы вариантов использования (Use Case):	8
Функциональное моделирование	8
Что можно почитать еще?	11
Используемая литература	11

Введение

На первой лекции мы познакомились с концептуальным проектированием и изучили основные этапы построения концептуальной модели, освоили на практике построение контекстной диаграммы, карты пользовательских историй и диаграммы сущность-связь.

Предметом данной лекции является построение функциональной модели деятельности, позволяющей наглядно выделить функции системы, проанализировать их взаимосвязи и назначение.

Цель настоящей лекции – предоставить системному аналитику знания и навыки обращения с инструментарием функционального проектирования систем, позволяющим решить вопрос выделения отдельных функций в системе и снижающим количество ошибок проекта.

Термины, используемые в лекции

Система – совокупность взаимодействующих или взаимосвязанных элементов.

Функция системы – это свойство системы в динамике, приводящее к достижению цели или другими словами функции выражают поведение системы.

Модель системы – формализованное описание системы.

Системный анализ – совокупность приемов научного познания представляющий собой последовательность действий по установлению структурных связей между переменными или элементами исследуемой системы.

Точка зрения (аспект) – позиция, относительно которой производится описание системы, процесса или явления.

Наблюдатель – лицо обозначающее точку зрения, относительно которой происходит описание системы.

Концепция – комплекс взглядов на что-либо, связанных между собой и образующих единую понятийную систему, являющую генеральной линией для анализа и проектирования системы

Концепт – воплощение идеи

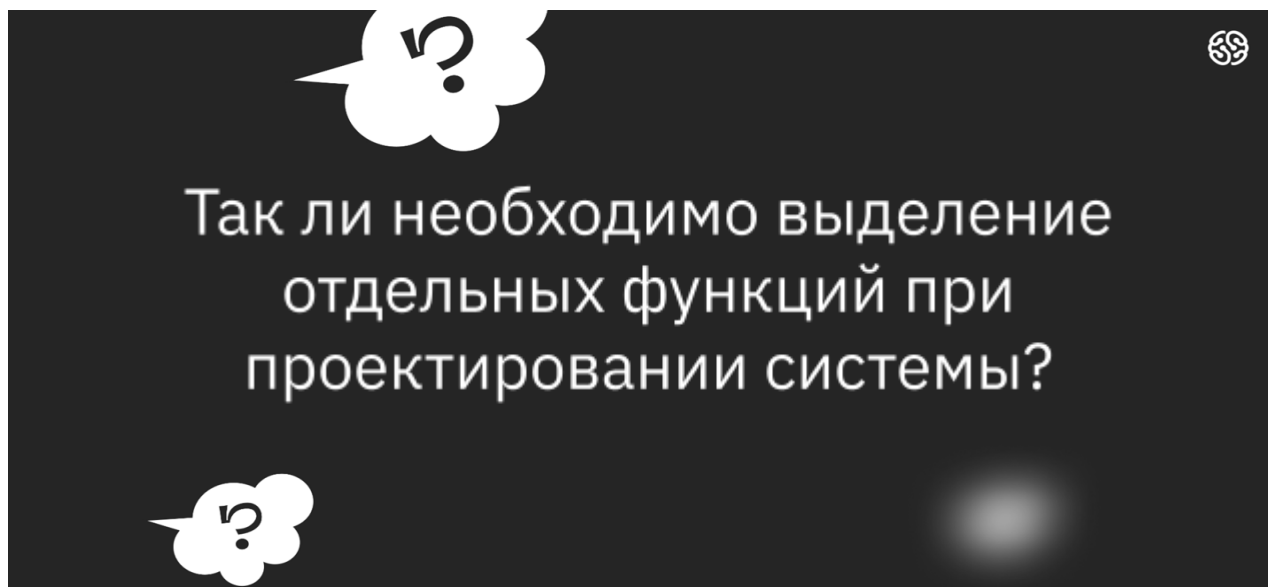
User Story – пользовательская история, служащая для выявления требований пользователя системы

MVP – минимально жизнеспособный продукт, являющийся концептом системы

Use Case – варианты использования, служат для выявления функций системы

Роль и место функционального проектирования

Для начала давайте ответим себе на вопрос, а так ли необходимо выделение отдельных функций при проектировании системы, и что произойдет если игнорировать данный этап анализа?



Ответ: на предыдущем курсе мы познакомились с понятием системы и системного анализа, разобрались с ролью и местом системного аналитика при проектировании систем, научились выявлять требования и узнали из чего состоит процесс разработки системы. На предыдущей лекции мы детально рассмотрели понятие концептуального проектирования, узнали для чего и в каких случаях нужно проведение концептуального исследования, познакомились с понятиями концепт и концепция, а также пользовательская история.

Концептуальное проектирование имеет своей целью определение целостного взгляда на систему и позволяет ответить на принципиальные вопросы качественного характера: «Для чего служит система?», «Как достигается её эффективность?», «Кем используется система?», «Какую ценность система несет потребителю?» и подобные.

Но целостный взгляд на систему не позволяет ответить на вопросы детального характера, раскрывающие аспекты динамики системы.

В качестве таких детализирующих аспектов выступают её функции, определяющие способ выделения частей из общего целого, что достигается за счет использования приема декомпозиции по функциональному признаку, то есть признакам заметным при наблюдении.

Итак, Функциональный признак определяет назначение подсистемы, а также её основные цели, задачи и функции. Структура информационной системы может быть представлена как совокупность её функциональных подсистем, а функциональный признак может быть использован при классификации информационных систем.

Игнорируя функциональные признаки существует реальная возможность совершить ошибку при выделении подсистем, что в конечном счете может привести к невозможности реализовать сложную систему.

💡 Функциональное проектирование в контексте системного анализа, можно определить как построение функциональной модели системы, раскрывающей композицию функций системы обеспечивающих её свойства в соответствии с заданной целью

Функциональное проектирование определяет детали модели системы необходимые для начала её реализации.

Важным вопросом на который отвечает функциональное проектирование, является вопрос минимизации функций системы, так чтобы описание функций и их реализация имела как можно более общий, инвариантный характер.

💡 Минимизация числа функций обеспечивает снижение стоимости её реализации.

Основой функционально-ориентированного проектирования выступает структурный анализ системы. Он заключается в следующем:

1. Выделение значимых функций системы и рассмотрение их относительно требований потребителей,
2. Иерархическое упорядочивание функций с учетом выявляемых логических связей и информационных потоков,
3. Представление всей информации в виде графической нотации – синтез функциональной модели системы доступной для однозначного понимания всеми участниками процесса разработки.

Определение функциональных границ проектирования системы

Когда основные требования получены от заинтересованных лиц и согласованы со всеми участниками, мы можем приступить к определению ключевых функций проектируемой системы. Предварительная оценка функций системы, дающая

представления о границах проектирования крайне важна с прикладной точки зрения управления проектом создания системы.

Стоит отметить, что для систем автоматизированного управления технологическими процессами наряду с функциональными границами выделяют задачи определения границы устойчивости системы и границы заданного качества управления. Но эти задачи относятся к задачам обеспечения качества проектирования отдельных функций и не входят в текущую лекцию.

Поговорим о функциональном наполнении системы. Дело в том, что в реальном мире системы не проектируются ради их проектирования. У заказчика есть не только требования, но и ограниченный бюджет, который выделяется на проект с условием создания системы, отвечающим заявленным требованиям, прежде всего функциональным.

Таким образом, на этапе концептуального проектирования системы определяются функциональные границы системы, в свою очередь определяющие границы проекта по созданию системы с учетом отдельных этапов, обычно выделяемых на реализацию групп функций. То есть, как основа для проектного управления, функциональное проектирование дает возможность верхнеуровневого оценить или подтвердить оценку стоимости и длительности проекта, направленного на создание конечного продукта.

Зачастую, в результате этого процесса выясняется, что не хватает либо времени, либо ресурсов, либо и того и другого для получения качественного результата в предусмотренные сроки. Детализация функций, рассмотрение их взаимосвязей и очередности исполнения важнейший этап функционального проектирования и обуславливает ведущую роль системного аналитика в рабочем процессе.

С одной стороны, умение эффективно определять границы проекта и управлять ими тем ценнее, чем меньше ресурсов заказчик готов выделить на проект по её созданию.

С другой, можно сказать, чем грамотнее аналитик умеет управлять функциональными границами проектирования, тем реализация системы будет проходить эффективней.

Цель этой деятельности сводится к максимально полному определению набора функций, который должны быть воплощены в разрабатываемой системе. Полный

набор функций служит для удовлетворения выявленных потребностей заказчика с учетом всех ожидаемых возможностей по его использованию.

Границы же проекта (project scope) показывают, какая область конечного продукта будет реализована в текущем проекте. Другими словами, определяется черта между тем, что мы будем делать сейчас и тем, что отложим на потом или от чего вынуждены отказаться. Для этого в арсенале аналитика и команды разработки должен быть инструмент, позволяющий не просто строить модели создаваемого продукта, а помогающий наглядно очертить функциональные рамки автоматизируемых процессов, а также предоставлять возможность легко выносить процессы за границу или включать их обратно. Это очень важно для осознания и более качественного планирования объемов работ на всех этапах. Подобный инструмент полезен не только для противодействия желанию заказчика разработать систему не затратив ресурса, но и для эффективного управления разработкой.

Для управления границами проекта, как на начальных стадиях, так и в течение всего проекта, очень удобно использовать функциональное моделирование.

В качестве примера определения границ проекта можно привести следующую ситуацию: для автоматизации обработки заявки требуется реализовать функции автоматизации ее приема, формато-логического контроля, классификации и исполнения.

Может иметь место попытка использовать не функции, а сущности предметной области или элементов системы в качестве факторов определяющих границы проекта. Но сущности предметной области или элементы системы не несут в себе достаточной информации для определения практической трудоемкости реализации проекта.

Если в данном примере в качестве границы проекта обозначить “создание системы заявок”, то такое описание не будет указывать на реализуемые функции и таким образом, это скорее всего не позволит оценить или подтвердить стоимость реализации проекта по созданию системы, тем более не позволит и определить этапность выполнения проекта.

Разумеется необходимо уточнить, если функции, или как ещё можно сказать, методы обработки одного типа заявки спроектированы, то трудоемкость реализации отдельной сущности в виде нового типа заявок может быть формализована за счет переиспользования опыта, но с учетом особенностей новой реализации.



Выделение сущностей или элементов системы без рассмотрения их функциональности не может служить основанием для практической оценки реализуемости и стоимости проекта по созданию системы

Так, если функционал автоматизации кредитной заявки без поручительства определён, то для реализации заявки с возможностью поручительства возможно переиспользовать отдельные функции и их оценки в новом проекте, тем самым снизив трудоемкость проектирования и реализации.

Определение границ проектирования является обязательным этапом, позволяющим эффективно управлять проектом реализации.

В качестве возможного результата деятельности по определению границ проектирования системы выступает перечень реализуемых функций и их краткое описание, а также указание на очередность реализации функций, обозначаемое через этап проекта, на котором предполагается реализация функции.



Без определения функциональных границ проектируемой системы сложно запустить реальный проект её разработки

Но, так как одни и те же функции системы могут быть задействованы в разных процессах разными участниками, и их представление при этом в различных аспектах может сильно отличаться, то в качестве следующего этапа функционального проектирования можно назвать описание вариантов использования.

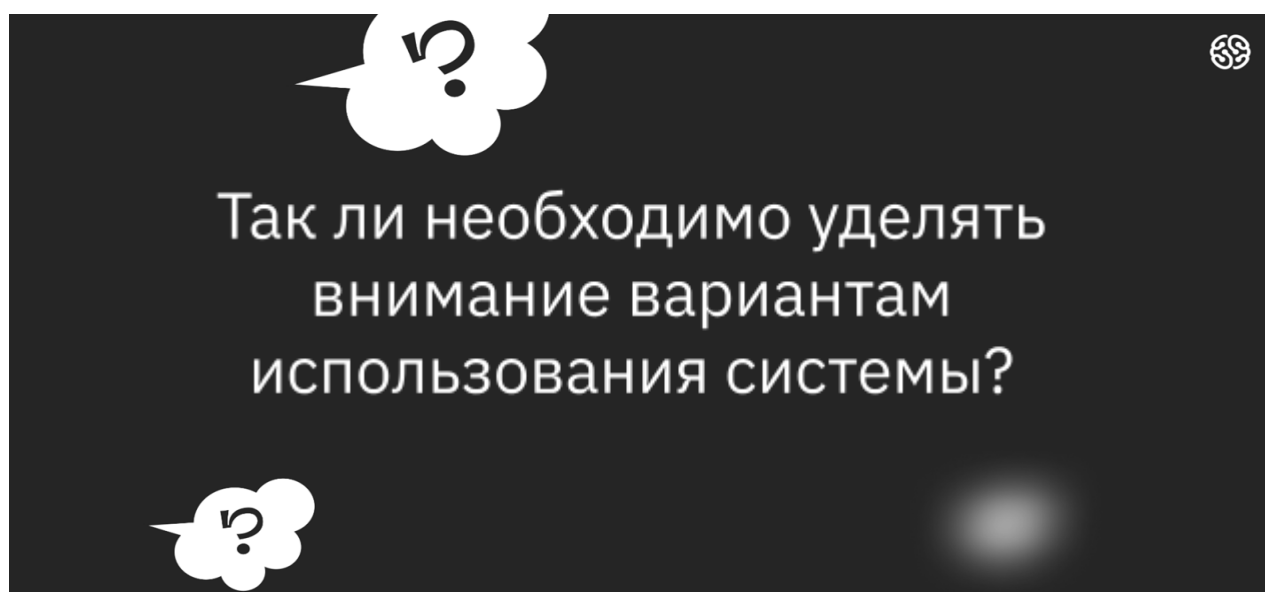
Описание вариантов использования системы

Сценарий использования или другими словами вариант использования (англ. Use Case) — в разработке программного обеспечения и системном анализе это описание поведения системы, когда она взаимодействует с определенным субъектом из внешней среды.

Вариант использования системы является совокупностью системной функции и пользовательской роли (англ. actor).

Прежде чем вдаваться в подробности применения подобной методологии описания сценариев использования системы давайте разберемся с необходимостью проведения подобного анализа.

Допустим, при проведении концептуального проектирования уже были уточнены требования заинтересованных сторон и определены границы и состав системы. Может быть выявление вариантов использования системы является опциональным? Например, сложно придумать варианты использования для планеты земля, если Вы просто живете на ней. Человек живет на планете Земля. Что может быть проще.



Ответ: разумеется уделять внимание вариантам или сценариям использования системы не требуется если они тривиальны. К сожалению или к счастью системного анализа, в сложных системах просто необходимо уделять пристальное внимание вариантам их использования для раскрытия значимых функциональных свойств и особенностей. Решение этой задачи позволяет сформулировать и многократно уточнить функциональные и системные требования.

Каждый вариант использования раскрывает последовательность действий, выполнение которых создает ценность для потребителя. Другими словами, **вариант использования однозначно определяет контракт – то есть зафиксированные требования к поведению пользователя и системы для достижения целесообразного результата.**

Для примера ценности подобного подхода давайте рассмотрим уже знакомую систему – банкомат (англ. АТМ).

Очевидно, что система взаимодействует с клиентом и банком, позволяя «Клиенту» получать базовые сервисы наличных денежных средств в отношении своего банковского счета. Вроде бы все очевидно и анализ избыточен.

Но банкомат вскоре бы опустел, не будь поблизости инкассатора, способного его наполнить деньгами и инженера, способного его обслужить и устранить технический сбой, обновить драйвер прошивки датчика и так далее.

Сами по себе роли могут быть определены на этапе построения концепции, но из требований заинтересованных сторон отдельные системные требования могут не проистекать. То есть об отдельных особенностях взаимодействия и технической реализации заинтересованные стороны могут быть не осведомлены, что вызовет пропуск определенных функциональных или системных требований на этапе концептуального проектирования.

Например, представьте, что существует еще роль - «Злоумышленник». Допустим злоумышленник ничем не отличается от клиента банка, но пользовательская роль, целью которой является завладение средствами банка или средствами клиента. Разумеется в реальной жизни все заинтересованные лица осведомлены о подобных вариантах использования системы, а представители безопасности должны выставить соответствующие требования на этапе построения концепции системы, блокирующие неправомерные действия.

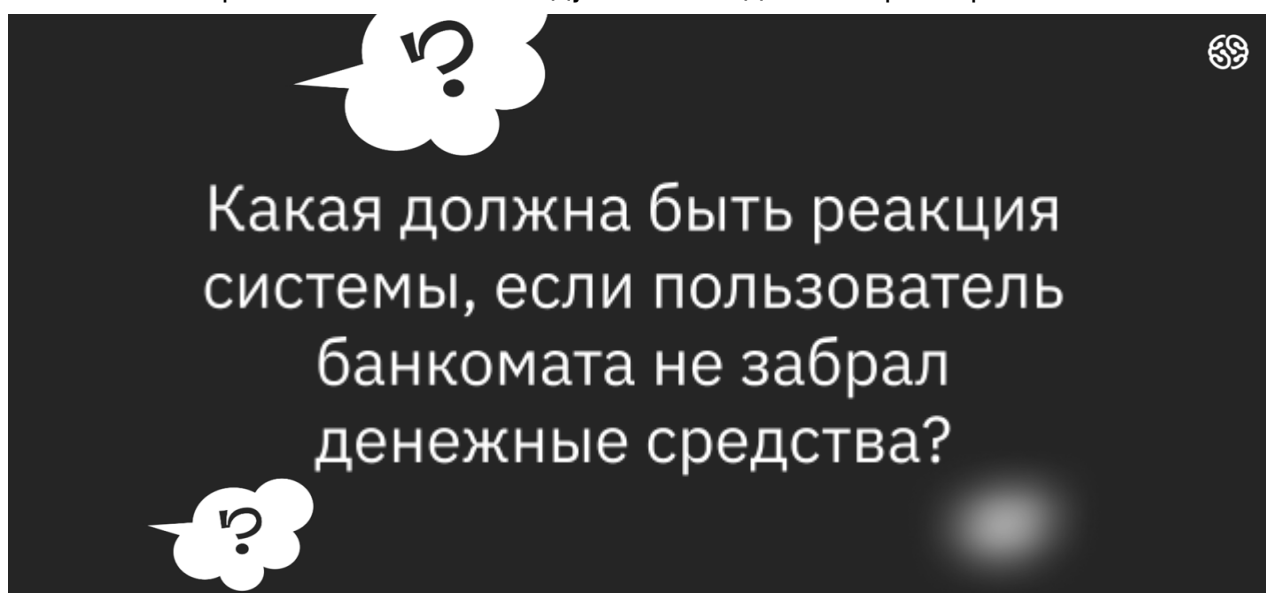
Но требования предъявляются верхнеуровнево, как «банкомат не должен позволять злоумышленнику завладеть деньгами клиента, а помещение в котором установлен банкомат должно быть оборудовано средствами видеофиксации ...». Но концептуальный проект не раскрывает всех деталей взаимодействия.

К примеру, рассмотрим следующий альтернативный вариант использования банкомата злоумышленником: банкомат должен иметь инновационное свойство идентификации клиента по профилю лица через распознавание видеосъемки выполняемой внутренней камерой банкомата без использования аутентификации по банковской карте. Заинтересованные стороны предъявили требования, а задача системного анализа их исполнить или обосновать риски, делающие такое использование нецелесообразным.

Допустим систему концептуально проработали, разработали и внедрили. Но после внедрения возник кейс (англ. case), то есть фактический опыт использования, который показал, что банкоматом могут пользоваться злоумышленники, сделавшие актерский грим идентичный лицам клиентов. Таким образом, функция идентификации имеет принципиальные уязвимости. Это может полностью разрушить концепцию, а может не повлиять на концепцию, если бизнес принимает риски. Так или иначе любой вариант использования расширяет функциональную модель системы и уже на уровне функций производится анализ и выявление всех взаимосвязей и требований как функциональных, так и системных.

В ходе выявления сценарием использования дается ответ на вопрос «как именно» внешний субъект взаимодействует с системой, и «кто именно» является рассматриваемым субъектом.

Рассмотрим другой альтернативный сценарий использования банкомата: клиент не забрал денежные средства. Как в этом случае будет реагировать система, если это не описано в требованиях. А как вы думаете она должна среагировать?



Ответ: система должна зафиксировать событие, выполнить пересчет и проверку купюр, поместить их во внутреннее хранилище банкомата и выполнить откат транзакции.

Как вы уже поняли, выделяют две группы сценариев использования:

- основной сценарий – служит для скорейшего приобретения ценности (услуги) потребителем, за счет исполнения функции системы,
- альтернативные сценарии – служат для раскрытия логики реакции системы на ошибки или нестандартное поведение участников процесса взаимодействия с системой.

•

Так системами может пользоваться множество людей, поэтому как мы отметили в примере, необходимо выделять различные группы пользователей системы. У каждой такой группы могут быть свои права и возможности в системе, что в итоге приведет к появлению уникальных вариантов её использования.

Такой анализ проводится не только для улучшения потребительских свойств системы, повышения её стабильности, но и для скрупулёзного рассмотрения системы с разных точек зрения, что в свою очередь позволяет проверять концепцию еще до реализации.

Вариант использования формализуется в виде совокупности следующих характеристик:

- Идентификатор – уникальный идентификатор варианта использования
- Описание – краткое описание варианта использования
- Роль – наименование взаимодействующей роли относительно которой строится вариант использования системы
- Предусловия – необходимые условия для выполнения сценария
- Иницилирующее событие – событие служащее точкой входа сценария
- Основной (базовый или успешный) сценарий – содержит пошаговое описание взаимодействия системы и роли, раскрывающие скорейшее получение положительного результата
- Альтернативные (неуспешные) сценарии – содержат пошаговые описания взаимодействия, служащие для проектирования требований к функционированию системы при иных значимых вариантах действия роли.

Примерный вид спецификации варианта использования:

Идентификатор	UC-2
Описание	Пользователь осуществляет выход из своей учетной записи
Какие сценарии должны быть пройдены	UC-1 – авторизация пользователя в ИС.
Предусловия	Пользователь авторизован.
Успешный сценарий	Пользователь вышел из ИС.
Неуспешный сценарий	Пользователь не вышел из ИС.
Участники сценария	Пользователь, Система

Условия начала	Пользователь находится на странице «ЛК».	
Описание шагов	№ шага	Действие
	1	Пользователь нажимает на свою учетную запись в левом нижнем углу и выбирает графу «Выход».
	2	Система завершает и очищает данные сеанса пользователя.
	3	Система перенаправляет пользователя на страницу авторизации в сценарии UC-1.

Если вариантов использования немного, а описываемые процессы просты, то сценарии использования выглядят адекватным инструментом анализа в целях выделения функций системы. Но данный подход ограничен в силу большого объема трудно воспринимаемого текста. Если же речь идет о рабочем документе проекта, то чем больше вариантов использования, тем сложнее поддерживать такое целостное описание.



Описание вариантов использования в текстовом виде подходит только для описания достаточно простых взаимодействий одной роли с системой предоставляющей определенный сервис

Ситуация ухудшается с развитием системы. Текстовые описания вариантов использования требуют постоянной актуализации, что ведет к увеличению трудозатрат на системный анализ.

В 1986 году Ивар Якобсон, позже соавтор Унифицированного Языка Моделирования (UML) и Рационального Унифицированного Процесса (RUP), впервые сформулировал методику визуального моделирования для описания сценариев использования.

Первоначально он использовал несколько иные термины — англ. usage scenarios и usage case, но ни один из них не был естественным для английского языка. И в конечном счете он остановился на термине use case — вариант использования. Предложенный подход позволил выделять множество сценариев использования, чтобы определить необходимый набор функциональных свойств проектируемой системы.

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал создаваемой системы доступен каждой группе пользователей.

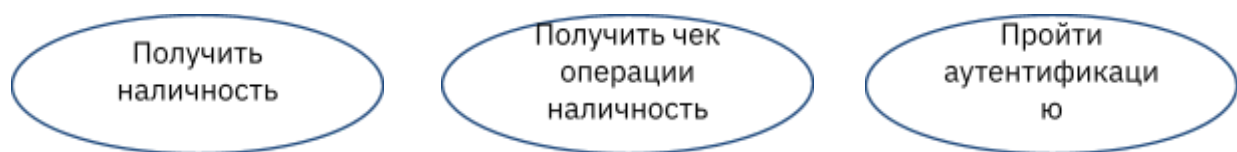
Для построения диаграммы все пользовательские роли (англ. actor) разбиваются на отдельные группы, представляющие понятийные классы. В своем примере мы выделили группы действующих лиц «Клиент», «Инженер сопровождения», «Инкассатор», «Злоумышленник». На диаграмме они изображаются стандартным символом actor.



«Клиент» «Инженер сопровождения» «Инкассатор» «Злоумышленник»

Далее на диаграмме вариантов использования изображаются функции системы в виде эллипсов, внутри которых записывается имя функции в форме глагола с пояснительными словами.

Например:



В свою очередь, варианты использования и пользовательские роли получают логические связи, позволяющие формировать однозначно трактуемое полное описание возможных вариантов использования системы. Для этого используются следующие связи представляющие отношения между понятийными сущностями:

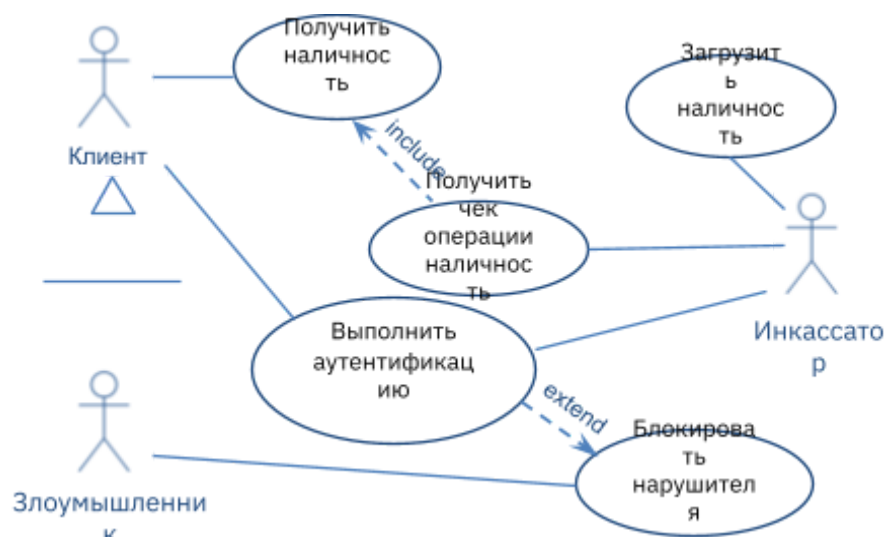
Отношение ассоциации 

Отношение обобщения 

Отношение включения 

Отношение расширения 

Варианты использования и пользовательские роли визуализируются в графическом виде.



Общие рекомендации по составлению диаграммы вариантов использования (Use Case):

1. Диаграмма служит инструментом накопления и систематизации функциональных и системных требований, с развитием системы она может быть многократно доработана и уточнена.
2. Не нужно углубляться в детализацию, и наполнять диаграмму слишком мелкими действиями. Объедините все общие действия в одну группу под общим названием, чтобы было просто читать диаграмму.
3. По возможности не допускайте пересечения соединительных линий. Это может затруднить её восприятие.
4. Не дублируйте варианты использования на диаграмме.
5. Пользуйтесь специализированными средствами моделирования для упрощения.



Направление стрелок определить просто: стрелки на диаграмме направлены от независимых сущностей к зависимым

Но раскрытие вариантов использования не определяет структурную композицию функциональных составляющих системы и не лучшим образом подходит для систем со сложным человеко-машинным интерфейсом.



Чем больше система тем больше сценариев необходимо проработать

В свою очередь, чем больше сценариев проработано, тем больше описание сценариев, что делает подобное описание малоинформативным и трудно воспринимаемым. Просто представьте проблему того аналитика, которому поручено модифицировать систему, для которой описаны 50 вариантов использования и каждый вариант содержит ещё по три альтернативные сценария, и уместается на двух страницах печатного текста.

Быстро понять логику функционирования такой системы будет сложно даже при наличии соответствующих диаграмм, поэтому применяется ряд приемов, позволяющих снизить сложность системы для построения её модели, максимально простой и однозначной для восприятия всеми участниками процесса. В качестве одного из подходов выступает проектирование на основе пользовательских путей применяется совместно с другими приемами функционального моделирования.

Проектирование на основе пользовательских путей в системах со сложным человеко-машинным интерфейсом

Варианты использования отлично подходят для функционального описания простых систем, с простой логикой взаимодействия с потребителем. И что самое важное они рассматривают процессы со стороны системы.

Но как быть в ситуации, если речь идет о функциональном моделировании системы имеющей множество нюансов пользовательского взаимодействия. При попытке описать все сценарии вы наверняка столкнетесь со значительным объемом текстового описания, не позволяющего выявить все особенности взаимодействия и лаконично описать варианты использования, как основные, так и альтернативные. Это особенно актуально при проектировании вэб или мобильных приложений, являющихся стандартными интерфейсами сложных информационных систем. На помощь приходит описание взаимодействий на основе пользовательского пути.

Пользовательский путь (англ. User flow) – это визуальное представление последовательности действий, которые пользователь выполняет для достижения своей цели. Может охватывать как какую-то отдельную функцию, так и полностью весь продукт.

Как и при построении вариантов использования, при проектировании системы на основе пользовательского пути необходимо ответить на три основных вопроса:

- Кто является пользователем?
- Какова его цель?
- Какие шаги он должен предпринять для достижения этой цели?

Это главные вопросы, на которые иногда бывает сложно дать четкие ответы, поэтому могут помочь вспомогательные вопросы:

- Для чего пользователь будет использовать приложение?
- Что побуждает пользователя достигнуть этой цели?
- Как приложение поможет достигнуть цели?
- Что может удержать пользователя от использования приложения?
- Какие качества продукта или услуги наиболее важны для заказчика и пользователей?
- В чем заключаются вопросы, сомнения и колебания?
- Какие качества приложения наиболее важны для пользователя?
- Какая информация нужна пользователям для выполнения действия?
- Какой эмоциональный триггер побуждает их к действию?

Подход на основе пользовательского пути позволяет взглянуть на приложение глазами пользователя. Правильное визуальное восприятие или интерпретация пользователем важнейший фактор успешности проектирования таких приложений. Все мы любим удобные и интуитивно понятные интерфейсы, любим, когда все работает логично и результат любого взаимодействия предсказуем. И чтобы все это было, нужно тщательно проработать каждую деталь. Любая мелочь способна сделать приложение недружелюбным пользователю и заставить его отказаться от использования.

Анализ на основе пользовательского пути поможет определить конкретные маршруты достижения пользователями цели, вычислить позитивные и негативные сценарии на выбранном пути достижения цели наших потенциальных пользователей. Он дает возможность понять, все ли процессы в продукте имеют

логическое завершение и выстроены эффективно, так, чтобы пользователь тратил минимум времени для достижения цели.

На следующем шаге, при функциональном моделировании, результаты текущего анализа используется для построения структурной схемы системы описывающей взаимосвязь всех функций системы.



Проектирование на основе пользовательских путей помогает исключить распространенную ошибку, когда создают список требуемых функций и на основе них просто проектируют интерфейсы.

Такой подход, разумеется, возможен. Но это может привести к тому, что сценарии взаимодействия с интерфейсом заставят пользователя проходить слишком много шагов для выполнения простых операций.

Важнейшим участником проектирования на основе пользовательских путей является дизайнер, в чью зону ответственности входит не только внешнее оформление интерфейсных элементов, но и их логическое сопряжения для построения дружелюбного интерфейса, в максимальной мере соответствующего запросам пользователя.

Представление на основе пользовательских путей может быть разным и зависит от того, насколько детально надо все проработать.

Обычно выделяют три уровня описания:

- Уровень задач (англ. Task flow)
- Уровень переходов (англ. Wire flow)
- Уровень экранов (англ. Screen flow)

Уровень задач формализуется блок-схемой алгоритма выполнения функции.

Уровень переходов формализуется совокупностью экранных форм и вариантами переходам между ними.

Самый низкий уровень абстракции представляет собой описание экранных форм в динамике процесса их использования пользователем.

Таким образом, проектирование на основе пользовательского пути, охватывая все вопросы пользовательского интерфейса позволяет выявить дополнительные требования для построения полноценной функциональной модели системы.

Структурное моделирование функций сложной системы

В системной инженерии, разработке программного обеспечения и информатике функциональная модель представляет собой структурированное представление функций (действий, процессов и операций) в моделируемой системе.

В действующей системе мало понимать, что определенный компонент работает неправильно, или не оптимально, важно понимать, каким образом он информационно взаимодействует с другими компонентами, отдельными элементами системы и внешними потребителями в контексте выполняемых функций.

Конечно следует помнить о том, что любая модель является не самим объектом, а его формализованным описанием.



Целью создания функциональной модели является такое описание функций и процессов которое позволяет с одной стороны однозначно интерпретировать модель для создания системы, а с другой позволяет синтезировать и оптимизировать внутреннюю структуру системы для повышения её надежности и эффективности, что выражается прежде всего в повышении её потребительских свойств и снижении себестоимости функционирования

Функциональная модель с одной стороны опирается на концептуальный анализ и начинается с главной функции системы с описанием информационных потоков, с другой стороны её детализация должна удовлетворить всем вариантам использования.

Функции системы описываются с одной точки зрения.



Точка зрения должна быть выбрана один раз до начала функционального моделирования

Например, в сквозном примере лекции при проектировании банкомата требуется определиться как описывать функциональную модель системы, со стороны пользователя – клиента банка или со стороны банка. В зависимости от точки зрения функции могут принимать различное описание «получение наличных» или «выдача наличных клиенту».

В качестве одного из инструментов функционального моделирования выступает группа стандартов IDEF.

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и структурного описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Стандарт IDEF0 был разработан в 1981 году в США департаментом Военно-воздушных сил для автоматизации промышленных предприятий. В процессе разработки программного обеспечения разработчики столкнулись с необходимостью разработки новых методов анализа бизнес-процессов. В результате появилась методология функционального моделирования IDEF0, в которой для анализа применяются специальные нотации IDEF0.

Применение диаграмм IDEF0 имеет следующие преимущества перед аналогами:

- Декомпозиция при анализе автоматизируемых процессов производится сверху вниз — от одного самого высокоуровневого к составляющим его подпроцессам. Последовательное разветвление процессов с уточнением их слой за слоем, позволяет с одной стороны, не упустить важные деловые процессы из поля зрения команды, а с другой работать всегда с одним выбранным узлом, содержащим небольшое количество элементов.
- Способ моделирования, при котором за основу берутся входящие потоки данных и управляющие на них воздействия, позволяет максимально точно и полно выявить все вложенные в процесс подпроцессы, обрабатывающие эти входящие данные и предоставить на выходе процесса результат – исходящий поток.
- Обязательное для каждого результата процесса (исходящего потока) — сопоставление другого процесса потребляющего его на входе (как входящий поток), позволяет не упустить функции в цепочке обработки и преобразования данных.

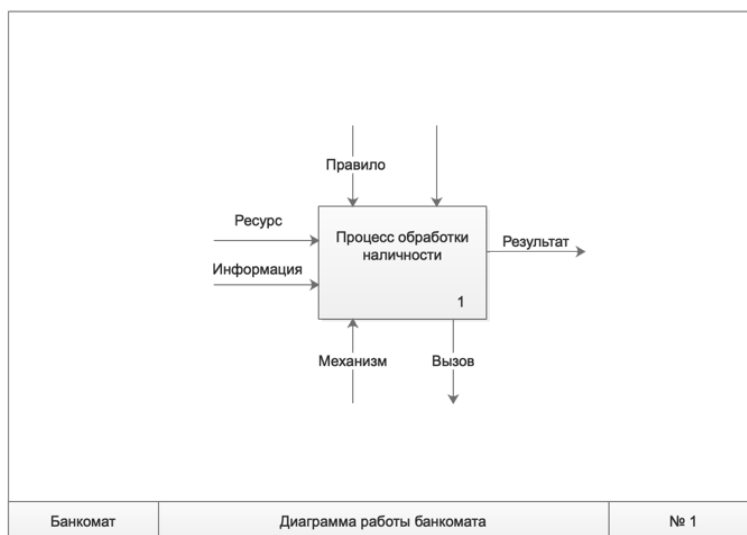
При помощи стандарта IDEF0 строится модель, описывающая основные функции, выполняемые системой и их взаимодействие в виде информационных потоков, в том числе с внешней средой. Таким образом на диаграммах IDEF0 легко читаются границы системы и ее окружение. Еще одно достоинство диаграмм IDEF0 является то, что вместо разработки одной большой, громоздкой модели, поэтапно создается несколько небольших, относительно простых моделей.

Таким образом, структура сложного процесса представляется в виде абстракции функций высокого уровня, которая раскладывается на более детальные процессы, увеличивая степень точности, слой за слоем.

Функциональная модель IDEF0 представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Наиболее важная функция расположена в верхнем левом углу.

А соединяются функции между собой при помощи стрелок и описаний функциональных блоков. При этом каждый вид стрелки или активности имеет собственное значение. Данная модель позволяет описать все основные виды процессов, как административные, так и организационные.

Стрелки могут быть:

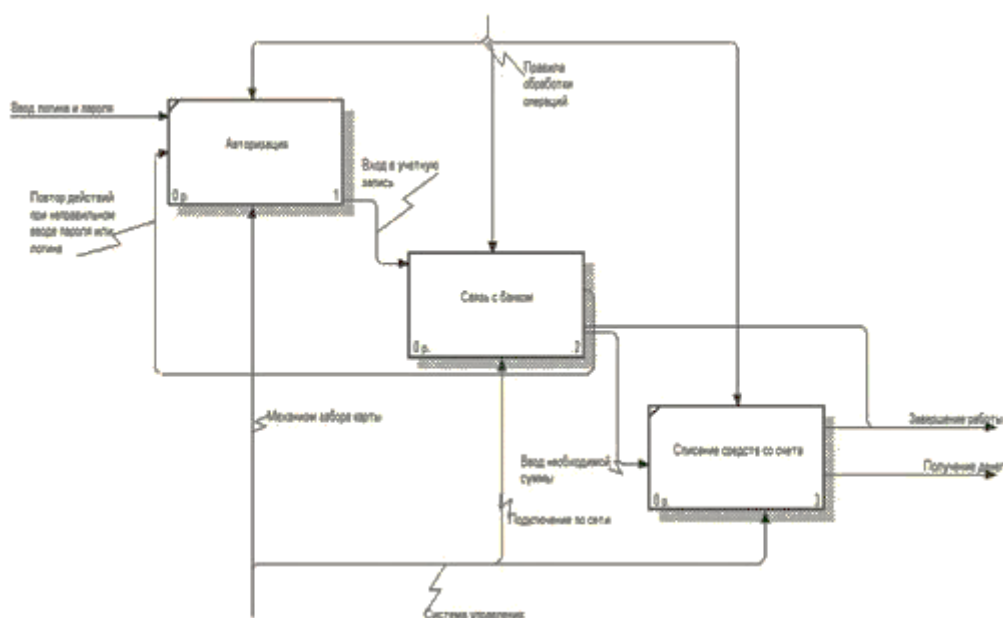


- Входящие – вводные, которые ставят определенную задачу.
- Исходящие – выводящие результат деятельности.
- Управляющие (сверху вниз) – механизмы управления (положения, инструкции и пр).

Механизмы (снизу вверх) – что используется для того, чтобы произвести необходимую работу.

Стрелки подписываются при помощи имен существительных (опыт, план, правила), а блоки – при помощи глаголов, т.е. в них описываются действия, которые производятся (создать товар, заключить договор, произвести отгрузку).

IDEFO – это очень простой и одновременно наглядный язык описания бизнес-процессов. С помощью этого стандарта возможна передача информации между разработчиками, консультантами и пользователями. Стандарт очень тщательно разрабатывался, он удобен для проектирования, и в определенной степени универсален.




Требования стандарта IDEF0:

1. В левом верхнем углу всегда – главный элемент.
2. Все элементы должны иметь входящие и исходящие стрелки, то есть ресурс и результат.
3. Входящие стрелки всегда слева, исходящие – справа.
4. Сверху – управляющие элементы, снизу – механизмы, необходимые для выполнения процесса.
5. Необходимо стремиться создавать схемы таким образом, чтобы пересечение стрелок было сведено к необходимому минимуму.


6. На одном листе (уровне детализации) нужно стараться располагать не более 6 функциональных блоков.

7. Разумное использование цветных элементов, с учетом того что ранжирование функций недопустимо.


В качестве типичных ошибок моделирования выступают нарушения требований стандарта, например применение цветовых решений для выделения функций, избыточное число функциональных блоков, множественные пересечения линий, смена точки зрения при описании.

 При создании модели нужно выбрать точку зрения и четко сформулировать цель создания функциональной модели

Такое моделирование не только наглядно, но и очень удобно не только при разработке системы, но и для принятия эффективных управленческих решений по её модернизации. При этом в качестве рассматриваемой системы может выступать как информационная системы, техническое средства, организационная системы, так и их объединение.

 Функциональные элементы выделяются до тех пор, пока не становится очевидным и однородным их техническая и информационная реализация

Главное полезное свойство, структурное описание системы позволяет выделить в системе функциональные блоки, описание которых может быть выполнено в виде совокупности пользовательских историй, пользовательских путей или процессов, что позволяет выполнить детализацию требований и определить взаимосвязи между требованиями, для последующей их совместной переработки.

 Дополнительно, этот вид моделирования позволит определить процессы, которые были упущены из поля зрения на этапах концептуального проектирования

При обнаружении противоречий между функциональностью и концепцией следует либо скорректировать концептуальное видение, либо выполнить перепроектирование отдельных функций, так чтобы концепция не была нарушена. Проще всего это показать на примере разработки обычного замка. Концепция замка предполагает наличие определенного ключа.

Но при проектировании системы мы можем столкнуться с необходимостью открытия замка специальными службами, не обладающими ключем. Как быть в такой ситуации? Либо менять концепцию, либо отказываться от функций, которые не поддаются реализации.

Так смена концепции может заключаться в наличии мастер ключа открывающего все замки, по примеру гостиниц или специальной команды на безусловное открытие. Реализация таких функций может быть крайне сомнительна, поэтому при проектировании эти функции могут быть выведены за границы проекта и специальные службы воспользуются специальным оборудованием для открытия замка, как это делается сейчас.

Цикл функционального проектирования

Все системы обладают своим жизненным циклом. Изначальный проект системы, сопровождается её реализацией и полезным использованием. В ходе этого процесса требования и функциональная модель системы многократно уточняются и перерабатываются. Какие-то функции добавляются, какие-то перестают существовать.

В своем развитии функциональная модель системы проходит через стадии цикла непрерывного улучшения (PDCA – цикл Деминга), где каждой фазе соответствует своя стадия проектирования.

Так, P (англ. Plan) - соответствует разработке функциональной модели, D (англ. Do) – соответствует реализации требований в соответствии с функциональной моделью, C (англ. Check) – проверка требований, A (англ. Act) – внесение изменений в функциональную модель.

Для реализации цикла необходимо обеспечивать трассировку требований, то есть вести учет взаимного влияния функциональных требований сгруппированных по функциям и сценариев их проверки.

Для решения этой задачи лучше всего подходит матрица трассировки требований (англ. traceability matrix).

Пример:

	ТС - 1	ТС - 2	ТС - 3	ТС - 4
Функциональный компонент 1	+	+	+	
UC-1.1	+			
UC-1.2	+	+		
UC-1.3			+	
Функциональный компонент 2		+		+
UF-2.1		+		+
UF-2.2				+

Табличный вид матрицы может быть дополнен следующими атрибутами, позволяющими однозначно идентифицировать связи требования с элементами проекта:

- Номер и описание задачи на разработку из системы управления задачами.
- Сущность, к которой принадлежит задача.
- Указание на тип требования - атомарное требование или критерий по которому производится приемка системы.
- Приоритет.
- Номер и описание соответствующего тестового артефакта.

Трассируемость требований позволяет:

- Следить за актуальным состоянием реализации.
- Декомпозировать требования.
- Отслеживать, есть ли требования, с незапланированной реализацией.
- Отслеживать, реализовано ли требование в данный момент.
- Отслеживать актуальность модели тестирования.
- Следить за соблюдением приоритетов требований при их реализации.
-

Можно выделить следующие этапы составления матрицы трассировки требований: В начале требования декомпозируются и подлежат приоритизации с учетом мнения команды разработки и заинтересованных лиц, включая заказчика. Результатом

этапа становится структурированный и приоритезированный список всех требований по данной функциональности.

Вторым этапом будет общение с командой разработки и наполнение матрицы заданиями на разработку к соответствующим требованиям. В результате мы можем отследить трассируемость требований и задач на разработку.

Третий этап — разработка модели тестирования с учетом тестовых сценариев.

4 этап — включение тестовых сценариев в матрицу.

По результатам всего процесса мы получаем задачи на разработку, тест-кейсы на тестирование и матрицу трассируемости, объединяющую их и требования.

5 этап — поддержка матрицы в актуальном состоянии. Изменения должны вноситься при любых модификациях требований. Также следует учитывать интеграционные связи между двумя матрицами, которые описывают разные компоненты системы.

Такой подход позволяет определить функции и сценарии проверки, которые необходимо доработать при изменении требований, но и требования, а также рабочие артефакты команды разработки, которые необходимо актуализировать при изменении функций.

А это, в свою очередь позволяет поддерживать функциональную модель системы в актуальном состоянии на всех этапах жизненного цикла системы и эффективно управлять проектом создания системы.



Игнорирование необходимости трассировки требований позволяет получить небольшое облегчение на старте создания системы, но всегда оборачивается головной болью её дальнейшего развития

Но разумеется не стоит выполнять трассировку требований для экспериментальных систем с коротким жизненным циклом, так как их главная задача проверить концепцию, на что может быть потрачена не одна итерация, и лишь при условии что концепция верна, следует выполнять полноценную трассировку требований.

Что можно почитать еще?

1. Для более глубокого изучения нотации функционального моделирования «Методология функционального моделирования IDEF0» ГОССТАНДАРТ РОССИИ
2. [Опыт использования стандарта IDEF0](#)
3. [Матрица трассабилити](#)

Используемая литература

1. В.В.Бахтизин, Л.А.Глухова, «Методология функционального проектирования IDEF0 », учебное пособие
2. А. А. Голованов, «Проектирование информационных систем», учебное пособие
3. Ф.А. Новиков, «Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML»