**4) Приведите классификацию требований. Опишите уровни требований к ПО. Опишите типы требований к ПО.**

Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований.

Требования можно разделить на следующие виды:

1. Бизнес-требования

2. Нефункциональные требования:

1. Бизнес-правила;
2. Внешние интерфейсы;
3. Ограничения;
4. Атрибуты качества.

3. Пользовательские требования;

4. Системные требования;

5. Функциональные требования;

Существует три основных уровня требований к ПО:

1. Требования бизнес-уровня

2. Требования пользовательского уровня

3. Требования уровня реализации

1) **Требования бизнес-уровня** обычно отвечают на вопросы: зачем создаётся продукт, какие и чьи проблемы он решает, какие возможности он кому предоставляет, и, отчасти, как он будет создаваться. Это обычно ограничения на продукт, которые уже известны до начала его создания, или известны, поскольку продукт уже существует, и мы его развиваем.

Кроме требований собственно к продукту, отчасти обычно на этом уровне описывают требования к тому, как будет выполняться проект по его реализации. Что будет в первой версии, на чём нужно сосредоточить внимание, главные приоритеты — это тоже, собственно, является частью бизнес-требований. Это определяет или очень сильно влияет на следующие уровни. Для первого уровня есть свои методы разработки требований, их можно обобщённо назвать разработкой Концепции (разработка бизнес-требований).

2) **Требования пользовательского уровня** — кто и как взаимодействует с продуктом. Для второго уровня тоже есть свои методы разработки требований — пожалуй, лучше всего проработанные. Всем известный, наверное, метод юзкейсов (вариантов использования) как раз относится к этому уровню.

3) **Требования уровня реализации** — это очень многообразный класс требований, относящихся к разным аспектам, к разным точкам зрения на продукт, которые, собственно, описывают, как он устроен.

**Бизнес-требования** - описание высокоуровневых целей организации или заказчика, достигаемых посредством разрабатываемой системы.

**Пользовательские требования** - описывают конкретный способ использования продукта конечным пользователем.

**Атрибуты качества** - Свойство продукта, выраженное через описание характеристик, важных для пользователей или разработчиков.

**Ограничения** - условия, которые модифицируют требование или набор требований, сужая выбор возможных решений. Обычно это внешние по отношению к системе условия или принятые ранее решения.

**Внешние интерфейсы** - описание интерфейса между системой и пользователем, другой системой или оборудованием.

**Системные требования** - это требования к продукту, содержащему несколько подсистем. То есть это некоторые требования, которые описывают то, как эти подсистемы должны между собой взаимодействовать.

**Функциональные требования** - это описание требований к системе в разнообразных форматах, которые, собственно, описывают, как она должна функционировать.

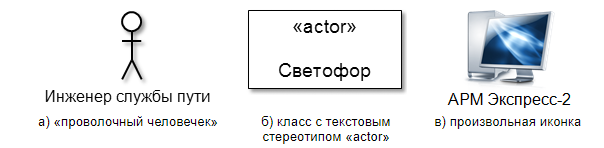
**14) Основные элементы диаграммной техники описания вариантов использования? Место диаграммы прецедентов в интегрированной модели системы. Привести примеры.**

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем [28]: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом, актором) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

В структурном подходе аналогом диаграммы вариантов использования являются диаграммы IDEF0 и DFD, вариантов использования – работы (IDEF0) и процессы (DFD), актеров – внешние сущности (DFD).

Согласно UML актера графически можно отобразить тремя способами.



Первый способ отображения в виде «проволочного человечка» является самым распространенным.

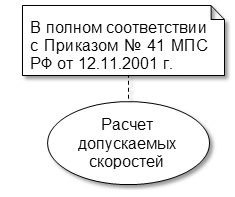
Вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его описание, обозначающее выполнение какой-либо операции или действия.



Вариант использования, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того, как система закончит обработку запроса актера, она должна возвратиться в состояние, в котором готова к выполнению следующих запросов.

Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы.

Примечания предназначены для включения в диаграмму произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемой системы. В качестве такой информации могут быть комментарии разработчика и ограничения. Графически примечания отображаются прямоугольником с загнутым верхним правым уголком, внутри которого содержится текст примечания. Линия, соединяющая примечание и элемент диаграммы, называется якорем (фиксацией).



Связи между актерами и вариантами отображаются с использованием отношений четырех видов:

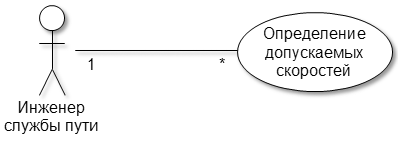
- ассоциаций;

- обобщения;

- включения;

- расширения.

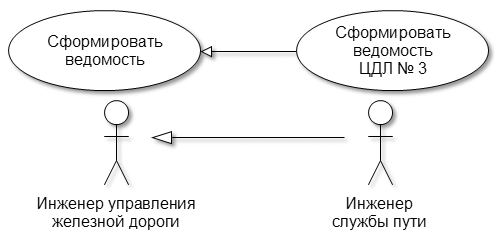
Применительно к рассматриваемой диаграмме отношение ассоциации служит для обозначения взаимодействия актера с вариантом использования.



Ассоциация может отображаться в виде однонаправленной или двунаправленной стрелки, показывающей направление потоков информации или управляющих сигналов.

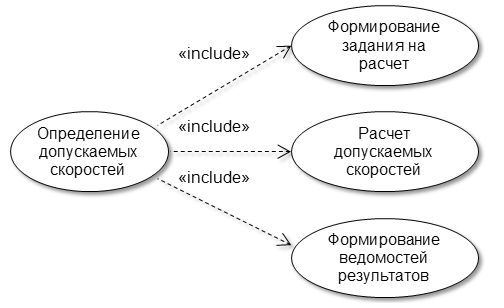
Отношение обобщения служит для указания того факта, что некоторая сущность А может быть обобщена до сущности В. В этом случае сущность А будет являться специализацией сущности В. На диаграмме данный вид отношения можно отображать только между однотипными сущностями (между двумя вариантами использования или двумя актерами).

Графически данное отношение обозначается сплошной линией со стрелкой, в виде незакращенного треугольника, от потомка к родителю.



Отношения включения и расширения являются частным случаем отношения зависимости и могут иметь место только между двумя вариантами использования. Они отображаются штриховой стрелкой с указанием стереотипа.

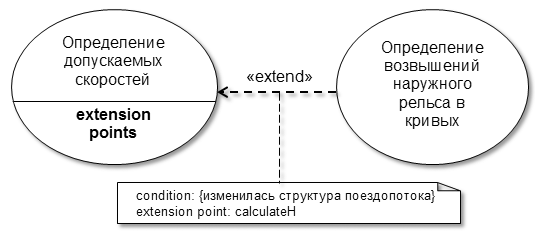
Отношение включения указывает, что некоторое заданное поведение одного варианта использования обязательно включается в качестве составного компонента в последовательность поведения другого варианта использования.



Стрелка включения должна быть направлена от базового (составного) варианта к включаемому и помечена стереотипом «include» (англ. включает) или «uses» (англ. использует).

В отличие от отношения включения, отношение расширения определяет потенциальную возможность включения поведения одного варианта использования в состав другого. Т. е. дочерний вариант использования может как вызываться, так и не вызываться родительским.

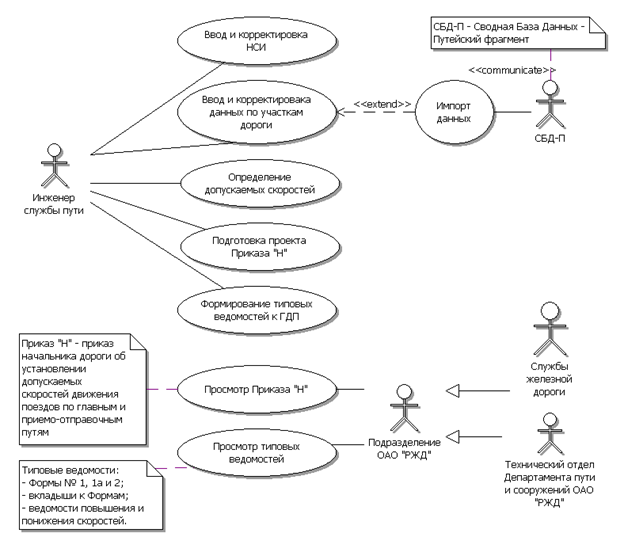
Стрелка расширения должна быть направлена от включаемого варианта к базовому и помечена стереотипом «extend» (англ. расширяет).



Ввиду того, что допускаемая скорость в кривых участках пути зависит в том числе и от возвышения наружного рельса, перед определением допускаемых скоростей может потребоваться определение и установление новых возвышений, которые в свою очередь зависят от структуры пропускаемого поездопотока.

Варианты использования, которые расширяют базовый, подключаются к нему (активируются при его выполнении) через так называемые точки расширения (англ. extension points). Каждая точка расширения маркируется меткой (на рис. 12.8 - calculateH) и условием (англ. condition) активации. Обычно перечень точек расширения указывается в базовом варианте использования ниже горизонтальной линии.

Пример:



**27) CASE-инструменты и технологии управления требованиями в реализации программных проектов**

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

* сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;
* наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);
* отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
* необходимость интеграции существующих и вновь разрабатываемых приложений;
* функционирование в неоднородной среде на нескольких аппаратных платформах;
* разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
* существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

В 70-х и 80-х годах при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако, широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

* неадекватная спецификация требований;
* неспособность обнаруживать ошибки в проектных решениях;
* низкое качество документации, снижающее эксплуатационные качества;
* затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе (феномен "сапожника без сапог").

Перечисленные факторы способствовали появлению программно-технологических средств специального класса - CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т.д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

* подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
* широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
* внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно обзору передовых технологий (Survey of Advanced Technology), составленному фирмой Systems Development Inc. в 1996 г. по результатам анкетирования более 1000 американских фирм, CASE-технология в настоящее время попала в разряд наиболее стабильных информационных технологий (ее использовала половина всех опрошенных пользователей более чем в трети своих проектов, из них 85% завершились успешно). Однако, несмотря на все потенциальные возможности CASE-средств, существует множество примеров их неудачного внедрения, в результате которых CASE-средства становятся "полочным" ПО (shelfware). В связи с этим необходимо отметить следующее:

* CASE-средства не обязательно дают немедленный эффект; он может быть получен только спустя какое-то время;
* реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
* CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения процесса их внедрения.

Ввиду разнообразной природы CASE-средств было бы ошибочно делать какие-либо безоговорочные утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

* широкое разнообразие качества и возможностей CASE-средств;
* относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
* широкое разнообразие в практике внедрения различных организаций;
* отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
* широкий диапазон предметных областей проектов;
* различная степень интеграции CASE-средств в различных проектах.

Вследствие этих сложностей доступная информация о реальных внедрениях крайне ограничена и противоречива. Она зависит от типа средств, характеристик проектов, уровня сопровождения и опыта пользователей. Некоторые аналитики полагают, что реальная выгода от использования некоторых типов CASE-средств может быть получена только после одно- или двухлетнего опыта. Другие полагают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла ИС, когда технологические улучшения могут привести к снижению эксплуатационных затрат.

Для успешного внедрения CASE-средств организация должна обладать следующими качествами:

* *Технология.* Понимание ограниченности существующих возможностей и способность принять новую технологию;
* *Культура.* Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;
* *Управление.* Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться неудачей независимо от степени тщательности следования различным рекомендациям по внедрению.

Для того, чтобы принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и противоречивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных "подводных камней" использования CASE-средств. Среди наиболее важных проблем выделяются следующие:

* достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;
* внедрение CASE-средств может представлять собой достаточно длительный процесс и может не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя может утратить интерес к CASE-средствам и прекратить поддержку их внедрения;
* отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, которые используются в данной организации, может привести к дополнительным трудностям;
* CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми различными средствами, так и проблемами передачи данных и управления от одного средства к другому;
* некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, при этом, тем не менее, можно извлечь выгоду из той дисциплины, к которой обязывает их применение;
* негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств должны быть готовы к необходимости долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Несмотря на все высказанные предостережения и некоторый пессимизм, грамотный и разумный подход к использованию CASE-средств может преодолеть все перечисленные трудности. Успешное внедрение CASE-средств должно обеспечить такие выгоды как:

* высокий уровень технологической поддержки процессов разработки и сопровождения ПО;
* положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;
* приемлемый уровень отдачи от инвестиций в CASE-средства.

**CASE-инструменты и технологии управления требованиями в реализации программных проектов**

CASE (Computer-Aided Software/System Engi­neering — Технология автоматизированной разработки програм­много обеспечения). CASE-технология представляет собой сово­купность методологий анализа, проектирования, разработки и со­провождения сложных систем программного обеспечения (ПО), поддерживаемую комплексом взаимосвязанных средств автомати­зации. CASE — это инструментарий для системных аналитиков, разработчиков и программистов, позволяющий автоматизировать процесс проектирования и разработки ПО.

CASE позволяет не только создавать "правильные" продукты, но и обеспечивать "правильный" процесс их создания. Основная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки, а также скрыть от разработчиков все детали среды разработки и функционирования ПО. Предполагается, что чем больше деятельности будет вынесено из кодирования в проектирование, тем лучше,

В большинстве современных CASE-систем применяются мето­дологии структурного анализа и проектирования, основанные на на­глядных диаграммных техниках. При этом для описания модели проектируемой системы используются графы, диаграммы, таблицы и схемы. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структу­ру со все большим числом уровней.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов сис­темной и программной инженерии, CASE обладают следующими основными достоинствами; они улучшают качество создаваемо­го ПО за счет средств автоматического контроля (прежде всего, контроля проекта); позволяют за короткое время создавать про­тотип будущей системы, что дает возможность оценить ожидае­мый результат на ранних этапах; ускоряют процесс проектирова­ния и разработки; освобождают разработчика от рутинной рабо­ты, позволяя ему целиком сосредоточиться на творческой части разработки; поддерживают развитие и сопровождение разработ­ки; поддерживают технологии повторного использования ком­понентов разработки.

Интегрированный CASE-пакет в совокупности должен:

* поддерживать графические модели;
* контролировать ошибки;
* организовывать и поддерживать репозитарии;
* поддерживать процесс проектирования и разработки.