МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Менеджмент разработки ПО

Электронный учебно-методический комплекс
по дисциплине в LMSMoodle

Работа выполнена по мероприятию блока 1 «Совершенствование
образовательной деятельности» Программы развития СГАУ
на 2009 – 2018 годы по проекту «Разработка магистерской программы
«Программное обеспечение мобильных устройств» по направлению
230100.68 Информатика и вычислительная техника»
Соглашение № 1/12 от 3.06.2013 г.

САМАРА
2013

УДК33(075) + 004.9(075)
М502

Автор-составитель: **Иващенко Антон Владимирович**

Менеджмент разработки ПО [Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle/ Минобрнауки России, Самар.гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост.. А.В. Иващенко. - Электрон. текстовые и граф. дан.- Самара, 2013. –1 эл. опт.диск (CD-ROM).

В состав учебно-методического комплекса входят:

1. Курс лекций семестр В.
2. Вопросы к экзамену семестр В.
3. Рабочая программа курса.

УМКД «Менеджмент разработки ПО» предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника» в семестре В.

УМКД разработан на кафедре информационных систем и технологий.

Цели и задачи сбора и формализации требований к ПО мобильных устройств. Основные определения. Место в жизненном цикла ПО

Системный подход к проектированию и разработке ПО мобильных устройств заключается, в основном, в моделировании и всестороннем анализе требований к этой системе. Под моделированием при этом понимается процесс создания достаточно точного и адекватного графического описания системы, а также интерпретация полученного описания для определения оценочных значений ее некоторых характеристик.

Модель (model) – это искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Модель может описывать существующие (AS-IS), идеализированные (SHOULD-BE) и вновь создаваемые (TO-BE) процессы и функции. На разных стадиях разработки используются различные уровни детализации модели.

При необходимости, модель может быть создана с использованием обычных графических и текстовых редакторов. Для обеспечения согласованности модели и ее высокой степени формализации, которая требуется при описании технического решения, необходимо использовать современные технологии автоматизированного проектирования – CASE-технологии (Computer Aided Software/System Engineering).

CASE-технология представляет собой методологию проектирования, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область и производить ее анализ на всех этапах разработки и сопровождения системы.

Данное пособие содержит краткое описание основных подходов к проектированию АСОИУ, изложенных в наиболее распространенных методологиях (см. Таблицу 1) и предназначено для использования в качестве конспекта части лекций по курсу «Проектирование АСОИУ». Для более глубокого изучения в списке литературы указаны соответствующие источники.

Таблица 1. Соответствие понятий проектирования АСОИУ.

	Методология	**Нотация или язык**	**Стандарт**	**CASE средства**
	Общие методы и технологии проектирования	Правила построения диаграмм	Формализа-ция подходов и языка	Инструмент автоматизирован-ного построения
Соответствие	Функциональная	SADT	IDEF0	BPWin (AllFusion), IDEF Doctor, MS Visio
	Потоков данных	DFD	–	
	Процессная	–	IDEF3	
	Сущность-связь	ER диаграммы	IDEF1X	ERWin (AllFusion)
	Объектно-ориентированная	UML	UML 2.0	Rational Rose, Star UML, Magic Draw MS Visio
	Процессная	eEPC/PCD, VAD	ARIS	ARIS Toolset, MS Visio

Источники информации о требованиях. Организация взаимодействия с Заказчиком. Способы и технологии проектирования требований к ПО

Требования к программному изделию – основа любого программного проекта. Они формируют как проектное задание, так и само развитие проекта. Качество создаваемого программного обеспечения во многом определяется тем, насколько оно удовлетворяет требованиям. Однако требования меняются, и эту изменчивость нужно уметь отражать в развивающейся системе. Новые требования могут противоречить ранее принятым, и потому нужно иметь критерии отбора требований для реализации. Наконец, формулировка требований чаще всего поступает к разработчикам в таком виде, который не подлежит непосредственному воплощению, а потому обычно говорят об извлечении требований из пожеланий пользователей, заказчиков, других инициаторов работ.

Традиционные методологии рассматривают определение и анализ требований как работу, которая предшествует собственно разработке и имеет целью выявление всей информации для последующего конструирования. И сбор требований рассматривается в качестве обособленного предварительного этапа. Утверждается, что успешность дальнейшей работы над проектом напрямую зависит от того, насколько полно и тщательно выполнен аналитический этап, что внесение корректив в зафиксированные требования приводит к необходимости повторения проектирования и всех других последующих этапов. Иными словами, изменение требований в процессе разработки рассматривается как ошибка аналитического этапа. Однако эта парадигма в большинстве случаев явно противоречит практике, что нашло отражение в известном афоризме: любая полезная программа нуждается в модификации, а бесполезная – в документации.

Из сказанного выше следует, что организация работы с требованиями относится к числу первоочередных задач менеджмента программных проектов. Ее решение не ограничивается предварительным периодом подготовки к проекту, а распространяется на все время жизни проекта. Как интегрирующее звено коллектива исполнителей проекта, менеджер обязан построить процесс так, чтобы в результате создаваемые продукты в полной мере отражали изменчивое движение требований к программному изделию.

В наиболее общем виде понятие требований сводится к следующим двум аспектам, фиксируемым для выполнения конструкторских работ:

– средства программного изделия, в которых нуждается пользователь для решения своих проблем или достижения определенных целей;

– характеристики программного изделия, которым должна обладать система в целом или ее компонент, чтобы удовлетворять соглашениям, спецификациям, стандартам или другой формально установленной документации.

Даже это не очень точное определение понятия требования указывает, что в реальности очень трудно исходя из аморфных и противоречивых пожеланий выявить, что конкретно и в каком виде должно быть воплощено в программном изделии. Требования первичны по отношению к программной разработке, определяют все ее развитие, являются начальным звеном в слагаемых качествах конструируемых программ. А потому задача управления требованиями должна рассматриваться как одна из главных задач проекта, претендующего на реальную полезность для пользователя.

Основные проблемы управления требованиями, с которыми приходится сталкиваться при их анализе, сводятся к следующему.

– **Требования имеют много источников.**

Даже если программная система разрабатывается по заказу, существует широкий круг людей, так или иначе заинтересованных в развитии проекта, – инициаторы работ. Это, разумеется, и будущие пользователи, и заказчик, и другие лица, которые осознают как необходимость автоматизации деятельности с помощью данной системы, так и рамки, за которые выходить не стоит. Указанные и другие персоналии, в частности сами разработчики

и их руководители, имеют свои, как правило, противоречивые представления о задачах проекта. Это тем более так, когда разработка претендует на удовлетворение рыночной потребности. От инициаторов работ зависит, какие работы целесообразны для реализации в проекте.

– **Требования не всегда очевидны.**

Смысл утверждения в том, что инициаторы работ далеко не всегда знают, какими средствами должна обеспечиваться поддержка автоматизируемой деятельности, в каких интерфейсных формах эта поддержка должна быть выражена. Очень часто не получается четко выделить и саму автоматизируемую деятельность.

– **Требования не всегда легко выразить словами.**

Интуитивное представление о том, какие средства должны предоставляться, чаще всего не формулируются явно. Приводится множество противоречивых примеров, соображений, но не описание нужных средств. В этой связи одна из главных задач анализа – представить требования в виде согласованных между заказчиком и разработчиками (одинаково понимаемых) утверждений, схем, диаграмм, моделей и т.п.

– **Существует множество различных типов требований и различных уровней их детализации.**

Совокупность требований весьма многопланова и соотносится с различными аспектами проекта. Следовательно, одной из задач анализа является типизация имеющихся сведений о требованиях и распределение их по этапам и итерациям разработки.

– **Требования почти всегда взаимосвязаны и взаимозависимы, часто противоречивы.**

Связанность требований обусловлена в первую очередь тем, что они относятся к автоматизации определенных видов системы деятельностей одной предметной области и наследуют ее системные связи. Кроме того, из-за автоматизации появляются и дополнительные связи. Однако пожелания к разработке даются разными людьми и в системах понятий, которые лишь косвенно соотносятся с предметной областью и поведением пользователя, решающего задачи из этой области. Не следует ожидать, что связи между требованиями будут ясно прослеживаться, что заранее будет сформулирована система объектов, которые воплощаются в программном изделии. Все, на что можно рассчитывать, получая сведения о требованиях, – это неформальное представление о том, кто будет работать с системой и зачем ему это нужно. Как следствие, в задачу анализа входит выявление взаимосвязей и взаимозависимостей, устранение противоречий, например, путем выработки рациональных компромиссов.

Следует отметить, что компромисс вовсе не означает достижение удовлетворительных результатов. Простейший, быть может, несколько утрированный, но показательный пример. Пусть одно требование к интерфейсу системы утверждает, что управление должно осуществляться только с помощью мыши, а другое указывает на недопустимость иного управления, нежели посредством горячих клавиш. Неразумный компромисс в таком случае – реализация одних воздействий с помощью мыши, а других – через горячие клавиши. В результате неудовлетворенными оказываются оба инициатора работ. Правильнее было бы построить две версии системы, одна из которых удовлетворяет первому требованию, а другая – второму. Конечно, в этом случае придется специально позаботиться о согласованном ведении версий, но уже на уровне реализационных механизмов, которые пользователям не видны.

Как показывает пример, в идеале нужно стремиться не к устранению противоречий, а тому, чтобы превращать их в идеи решений. К сожалению, по разным причинам это удается далеко не всегда.

– **Требования всегда уникальны.**

При формулировке требований как регламента разработки всегда нужно учитывать свойства или значения свойств, по которым они различаются: не существует двух равно значимых требований. Это не так, если рассматривать исходный материал для требований.

Тем не менее не следует сразу отбрасывать некоторое новое требование только потому, что оно кажется похожим на ранее рассмотренные. Необходимо проанализировать, какие дополнительные стороны оно характеризует, и получить аргументированный ответ на вопрос, действительно ли данное требование является новым. По существу, утверждение об уникальности требований определяет то, как они должны быть представлены в проекте в результате анализа (требование к требованиям).

– **Набор требований чаще всего является компромиссом.**

Это компромисс между пожеланиями инициаторов работ, направленный на максимально возможное расширение сферы применения системы. Существует много заинтересованных лиц, чьи усредненные требования должны быть удовлетворены в рамках выполняемых ими функций в прикладной области. Противоречия между требованиями, возникающие в связи с этим, ставят перед разработчиками проблемы, обычным способом преодоления которых является компромисс. В большинстве случаев компромисс можно считать лишь удовлетворительным, но никак не хорошим решением.

– **Требования изменяются.**

Фиксируемые в заказе на разработку требования к системе, претендующей на широкую сферу применения и долгую жизнь, не являются незыблемыми. Они изменяются как из-за учета новых факторов и пожеланий, так и в связи с выявлением особенностей проекта в ходе его разработки. Следовательно, необходимо строить аналитическую работу так, чтобы иметь возможность оперативно изменять получаемые результаты и учитывать в них изменения и дополнения исходной информации.

– **Требования зависят от времени.**

Это положение указывает на то, что пробное и экспериментальное знакомство с первыми получаемыми результатами (программными и документными), вероятно, повлечет за собой корректировку требований. Как следствие, нужно иметь в виду, что при выпуске очередной версии рабочих продуктов или при переходе от релиза к релизу вполне реальна ситуация проведения анализа требований вновь, а потому анализ и следующие за ним этапы должны быть организованы так, чтобы было как можно меньше переделок программ и документов.

– **Требования очень трудно оценивать.**

Это многоаспектное положение указывает на то, что на вопросы значимости требования, с одной стороны, а с другой – цены его реализации зачастую не удается найти удовлетворительных ответов.

Относительно просто оцениваются время и ресурсы, необходимые для разработки программного кода, отвечающего требованию. Значительный разброс оценок может оказаться и для этих параметров. Он связан, например, с различиями квалификации сотрудников. Но это ни в какое сравнение не идет с проблемами оценки постановки задачи на программирование и встраивания кода в систему. Подобные проблемы объективно обусловлены тем, что уровень системы значительно сложнее уровня составляющих ее компонентов, а достоверность, обозримость и точность информации системного характера, как правило, недостаточны для оценочного оперирования (по крайней мере, для программирования это так).

Не лучшим образом обстоит дело и с оцениванием потребительской значимости требования. Это также связано с необходимостью действовать на системном уровне, но уже на уровне системы деятельностей потенциальных пользователей и других инициаторов работ. Простых утверждений о том, что изучение прикладной области позволит выявить актуальность и что заказчик лучше других знает реальные потребности пользователей, явно недостаточно, а исследования внешних систем деятельностей, вовлекающих в себя программный продукт, слишком дорого стоят, и не без оснований считается, что чаще всего они не окупают себя.

Независимо от уровня первоначальной проработки требований к проекту не стоит думать, что требования всегда будут оставаться неизменными. Необходимо быть готовым к

тому, что в любой момент могут появиться новые требования, одни старые требования изменятся, другие – отпадут. Но основная сложность управления процессом изменения требований заключается не в этом, а в том, что изменения одних требований влияют на другие, и такие влияния нужно отслеживать. Влияние изменений требований естественным образом распространяется на все рабочие продукты проекта, в том числе на программные рабочие продукты.

Любое предложение по развитию конструируемой системы может быть классифицировано как требование одного из трех видов:

– дополняющее, которое отражает ранее не рассматривавшийся аспект системы;

– модифицирующее, которое изменяет одно или несколько уже существующих требований;

– отменяющее, принятие которого исключает одно или несколько ранее принятых требований.

Вид требования отражает различия анализа нового требования в контексте существующих соглашений. Целью такого анализа является поддержка целостности системы требований: нахождение противоречий между требованиями, разрешение противоречий, а при невозможности этого – достижение приемлемых компромиссов. Следует отметить, что требования могут оказаться противоречащими не только друг другу, но и уже принятым проектным решениям. В работах с меняющимися требованиями значительное место занимает отслеживание связей проекта, благодаря которому определяется деятельность, необходимая как для реализации требований, так и для распространения изменений, связанных с требованиями к проекту.

Таким образом, вопрос о том, принять или отклонить требование, является очень ответственным, зачастую влекущим за собой цепь связанных решений на всех уровнях проектирования. Чтобы сделать ответ на него обоснованным, необходимо выполнение как минимум двух условий:

– требования должны быть заданы в виде, допускающем однозначное представление в моделях уровня анализа и конструирования, и способ такого представления должен быть унифицирован для всего проекта;

– в проекте должны инструментально поддерживаться связи между требованиями и между требованиями и другими компонентами рабочих продуктов, и эта поддержка должна быть обеспечена.

Представление требований и пожеланий, исходящее от инициаторов работ, ни в коей мере не способствует соблюдению указанных условий. Следовательно, они должны быть трансформированы, т.е. преобразованы к виду, приспособленному для анализа. Прохождение исходного требования через последовательность трансформаций от одного представления к другому, сопровождающееся соответствующим анализом, называется **трассировкой требования**. Основное назначение трассировки в том, чтобы в любой момент развития проекта сохранялась целостность и непротиворечивость конструируемой системы, реализующей принятые требования .

В первую очередь трассировке подвергаются требования, предъявленные первоначально, т.е. до того, как проект начал развиваться. Но было бы неправильно ограничиваться только ими, поскольку связи новых требований с уже сложившейся системой требований, как явные, так и обнаруживаемые в ходе анализа, также требуют соответствующего анализа и других работ, необходимых для реализации. Это, а также то, что для отбора и анализа требований, поступающих в ходе работ над проектом, появляется дополнительный критерий реализационной совместимости с принятыми требованиями, отличает трассировку новых и первоначальных требований.

В результате трансформаций строятся представления требований, вид которых приспособлен для выяснения целесообразности реализации требований. Если на некотором

уровне трансформаций установлено, что данное требование отвергается, то его дальнейшие преобразования не производятся. Выделяются следующие представления требований:

1) **Исходное представление** – текстовое описание пожеланий к системе, заданное в свободной форме. Это описание, в частности, может фактически содержать несколько требований, отражающих разные аспекты проекта, – элементарные составляющие требования.

2) **Унифицированные представления** – исходное представление требования разбивается на элементарные составляющие, которые описываются в базовом виде, приспособленном для дальнейшего использования на всех проектных уровнях. В частности, здесь могут применяться формализованные описания элементарных составляющих требования. Во всяком случае, на уровне унифицированного представления достигается однозначность понимания требований.

3) **Типизированное представление** – каждое из элементарных составляющих требования приписывается к некоторому типу. В результате формируется набор атрибутов элементарных требований и их значений. Эта информация допускает почти формальное сопоставление элементарных требований с различными требованиями, уже представленными в проекте. Сопоставление проводится на разных уровнях иерархии типов требований к системе.

4) **Модельные представления уровня анализа** – образы элементарных требований как элементы аналитических моделей системы.

Для трассировки важно знать, какие модели и в каких частях затрагивает то или иное требование, что позволяет отслеживать связи, возникающие при трансформации требований. Не менее существенно иметь возможность оперативно получать соответствующую информацию о текстах программ и о документации.

Если требование принимается на уровне анализа, то трассировка продолжается на следующих уровнях и можно говорить о продолжении последовательности трансформаций в реализации требования в компонентах программного изделия:

1) **Модельные представления уровня конструирования** – образы элементарных требований в диаграммах классов, состояний и других компонентах архитектуры системы. На этом уровне требования принимаются или отклоняются в зависимости от их соответствия уже разработанной части проекта.

2) **Программные представления** – программные рабочие продукты и их фрагменты, которые рассматриваются в качестве образов требований, представленных очередной **версией** системы.

3) **Документные представления** – фрагменты документов, сопровождающих программный код и предназначенных для **поддержки** деятельности пользователей.

Схема на рис. 1 иллюстрирует приведенную последовательность трансформаций. Первые три представления требований изображены в виде совокупностей стрелок, которые при переходе от одного представления к другому становятся все более упорядоченными.

Иерархия типов требований представлена на рисунке следующим образом. Верхний уровень – это абстрактный тип, свойства которого присущи требованиям всех типов (они сводятся к стандартизованному набору операций объединения, пересечения атрибутов, сравнения значений атрибутов и др.). Можно сказать, что $T_{абстр}$ задает регламент, которого следует придерживаться при оперировании требованиями. Следующий уровень содержит четыре обязательных типа: $T_{экон}$, $T_{функ}$, $T_{инт}$ и $T_{эфф}$, которые объединяют требования экономического характера (пределы стоимости, рентабельность и пр.), функциональные требования, требования к интерфейсу и эффективности (производительности). Многоточием обозначены типы, которые, добавляются из-за специфики проекта. $T_a(a1,...,an_a)$, $T_b(b1,...,bn_b)$, $T_c(c1,...,cn_c)$, ..., $T_z(z1,...,zn_z)$ – это конкретные типы, которым приписываются элементарные составляющие требований (в скобках указаны их атрибуты).

Модельные представления уровней анализа и конструирования изображены в виде условных схем различных видов. Программные и документные представления – это текстовые файлы, пиктограммы которых показаны на рисунке.

На схеме представлен блок, обозначающий глоссарий проекта. Это очень важный инструмент согласования понятий, используемых в программной разработке. Глоссарий может пополняться на любой стадии трассировки требований, когда появляются новые понятия, смысловую трактовку которых нужно зафиксировать. Тем самым глоссарий отражает текущее понимание проекта в целом. Важно подчеркнуть, что когда разработчики не занимаются ведением глоссария, система понятий проекта все равно складывается, но стихийность этого процесса приводит к дополнительным издержкам коммуникаций сотрудников.

Приведенная схема демонстрирует, что в той или иной форме вынуждены делать разработчики для преодоления трудностей управления требованиями. Она может рассматриваться в качестве проекции жизненного цикла на задачи анализа требований. Каждое требование, поступающее для анализа, проходит вполне традиционные этапы жизненного цикла, правда, в несколько специфичном виде: учитываются только те работы, которые имеют отношение к моделированию требований. Но эта абсолютизация трассировки не является недостатком. Напротив, явное выделение задач управления требованиями уже само по себе способствует более успешному их решению.

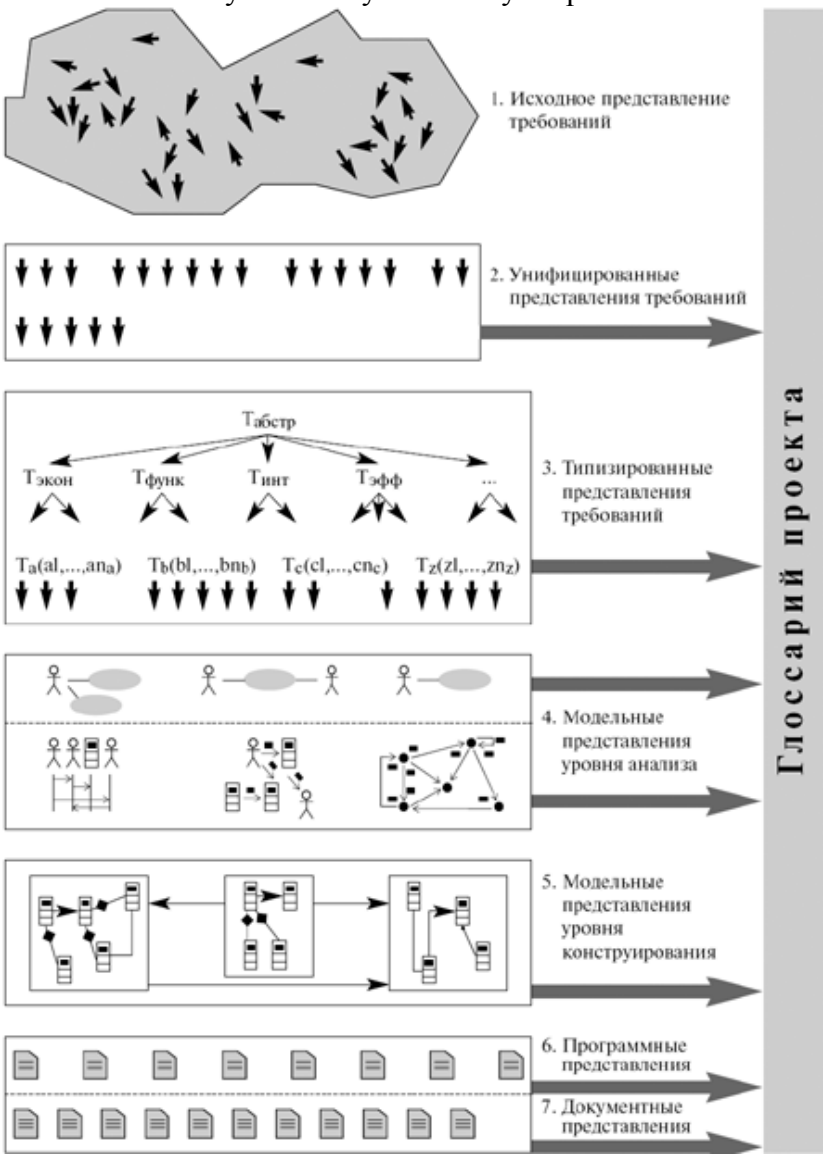


Рисунок 11. Схема трансформации требований

Виды требований к ПО. Перечень требований по ГОСТ 34.602-89, ГОСТ 19.201-78

В нашей стране госты серий 19 и 34 часто применяются при создании программ и автоматизированных систем, особенно, когда в качестве заказчиков выступают государственные или крупные коммерческие организации.

Стандарты ГОСТ серии 34 обладают следующими достоинствами:

– приемлемый уровень разумности и исполнимости;

– широкая распространенность по сравнению с другими стандартами, привычность терминологии и базовых понятий большинству заказчиков

– Минимальный набор жестких требований и возможность адаптации требований стандартов под конкретные условия тех или иных проектов.

Из сравнительно большого спектра ГОСТов серии 34 обычно применяются следующие:

– ГОСТ 34.003-90 «Автоматизированные системы. Термины и определения»;

– ГОСТ 34.201-89«Виды, комплектность и обозначение документов при создании автоматизированных систем»;

– ГОСТ 34.601-90«Автоматизированные системы. Стадии создания»;

– ГОСТ 34.602-89«Техническое задание на создание автоматизированной системы»;

– ГОСТ 34.603-92«Информационная технология. Виды испытаний автоматизированных систем»;

– РД 50-34.698-90«Автоматизированные системы. Требования к содержанию документов».

Несмотря на ряд достоинств, стандарты ГОСТ серии 34 обладают и рядом существенных недостатков.

– Эти ГОСТы не образуют целостную систему, поскольку разработка данной серии ГОСТов была прервана в начале 1990-х годов и с тех пор эти ГОСТы не актуализируются.

– Автоматизированные системы не рассматриваются как инструмент автоматизации бизнес-процессов и, как следствие, в ГОСТах серии 34 не прорабатываются вопросы изменения бизнес-процессов организации, вызванные внедрением автоматизированной системы.

– Очень поверхностно рассматриваются аспекты обслуживания внедренной автоматизированной системы: обучение персонала, выполнение регламентных процедур и т. п.

– Целый ряд ключевых понятий современного управления проектами, таких как риски, программы проектов и портфели проектов, в ГОСТах серии 34 отсутствуют вовсе, а как следствие, их проработка стандартом не предусмотрена.

Методологии проектирования ПО мобильных устройств. Общие принципы. Технологии проектирования. CASE средства проектирования

Системное проектирование сложных комплексов программ для информационных систем охватывает период их жизненного цикла, начиная от формулирования первичного замысла на создание или модернизацию информационной системы и до начала детального проектирования и разработки программных средств.

Результатом этого периода работ должно быть согласованное и формализованное разработчиком и заказчиком представление о целях, назначении, функциональных задачах и качестве будущих программных средств, способных удовлетворить надежды и запросы пользователей.

Основная цель системного проектирования – подготовить, обосновать и согласовать замыслы и решения заказчика (потребителя) и разработчика (поставщика) о необходимости, направлениях и концепции создания или модернизации существующего программного средства и изменениях его качества на базе современных стандартов.

Методы и средства системного проектирования должны подготавливать эффективную технологическую базу для обеспечения всего жизненного цикла программного средства требуемого качества.

Характеристики качества комплексов программ должны анализироваться и формулироваться в начале их жизненного цикла и определять эффективность всех последующих процессов. Поэтому целесообразно выделять задачи, которые необходимо решать при подготовке и первичном развитии процессов жизненного цикла, обеспечивающих впоследствии высокое качество программного средства.

Результатом этих работ должны быть системный проект, техническое задание и контракт на продолжение проектирования или решение о его нецелесообразности и прекращении.

В системном проекте должны быть обобщены и отражены следующие основные результаты выполненных исследований и разработок:

- обобщенный анализ проведенного обследования объекта информатизации, функций существующей информационной системы, качества ее основных программных компонентов и базы данных;
- совокупность предварительных исходных требований к функциям и характеристикам качества комплекса программ;
- оценки имеющихся и потенциально доступных ресурсов (финансовых, вычислительных средств, специалистов) для обеспечения всего жизненного цикла и требуемого качества проекта комплекса программ;
- результаты предварительного анализа архитектуры комплекса программ на основе моделей и прототипов аналогичных систем, позволяющие наметить планы разработки и всего жизненного цикла проекта программного средства;
- цели, задачи и функции предполагаемой новой или модернизированной информационной системы, обобщенные в концепции создания соответствующего программного средства;
- проекты планов жизненного цикла, гарантирования требуемого качества программного средства, защиты и обеспечения безопасности его функционирования;
- результаты технико-экономического обоснования целесообразности и основных направлений продолжения проектирования и всего жизненного цикла программного средства;
- результаты анализа существующей и возможной инструментальной среды разработки, а также системы обеспечения качества, перспективы их развития и совершенствования;
- предварительный план организации работ, требования к составу и квалификации специалистов для выполнения проекта и всего жизненного цикла программного средства;

- проект формализованного технического задания и спецификации требований к программному средству, а также предложения по его финансированию и обеспечению ресурсами;
- системный проект, обобщающий проведенные исследования и разработки, позволяющий заключить контракт между разработчиком и заказчиком на финансирование и продолжение проектирования и/или на весь жизненный цикл программного средства.

Системное проектирование сложных программных средств начинается с обследования объекта информатизации, системного анализа предметной области и выявления потребности в создании или модернизации комплекса программ с определенными функциями и качеством (анализ «AS-IS»).

Аналитики-консультанты совместно с потенциальными разработчиками и заказчиком или пользователями должны проводить анализ прикладной области и объекта информатизации, разрабатывать стратегию разработки и технико-экономическое обоснование реализуемости выдвигаемых требований.

Разработка исходных требований для технического задания на проект программного средства начинается с анализа результатов обследования объекта и оценки доступных ресурсов для реализации проекта. Эта деятельность требует специальной организации специалистов высшей квалификации и тесной совместной работы представителей заказчика и разработчика. Они должны подготовить исходные данные и документы, в которых содержатся предварительные требования и пожелания к функциональным и конструктивным характеристикам качества программного комплекса. Затем ими должна проводиться сложная работа по предварительному упорядочению, обобщению и выбору приоритетов требований для их реализации в проекте. Наличие обычно ряда неформализованных, неструктурированных и противоречивых содержательных требований заказчика и разработчика требует их совместной обработки, согласования и корректировки.

В соответствии с возможностями реализации необходимо сформулировать требования заказчика к процессам и результатам обработки информации в спецификациях требований к комплексу программ и его программным и информационным компонентам. Должна быть предусмотрена корректировка, конкретизация и развитие предварительных требований в процессе системного проектирования при тесном взаимодействии заказчика и разработчика. Для крупномасштабных проектов программных средств целесообразно использовать специальные решения в процессе отработки требований, которые следует учесть в системном проекте и техническом задании, а также применять для контроля их реализации.

В процессе системного проектирования последовательно уточняются характеристики объекта и среды разработки, вследствие чего появляется возможность более полно и точно спланировать и обосновать весь жизненный цикл программного средства. Одновременно уточняются перечни частных работ и приближенные графики их выполнения. Предполагаемая длительность эксплуатации, прогнозируемый тираж и число версий программного средства отражаются на плане, технико-экономических показателях, качестве и технологическом процессе разработки. Основные технико-экономические показатели процесса разработки необходимо оценивать с учетом конкретных требований к объекту разработки, ограничений на ресурсы, а также характеристик инструментальных средств и коллектива специалистов.

Таким образом, последовательное прогнозирование, планирование и системное управление проектом обеспечивают рациональное использование ресурсов в процессе создания сложного программного средства гарантированного качества. Если необходимые требования к функциям и качеству программного средства не могут быть удовлетворены при доступных ресурсах, технологиях и специалистах, то возможны решения по прекращению дальнейшей разработки.

Прогнозы и анализ вариантов технологических процессов проектирования программного средства, их технико-экономических показателей и характеристик объекта

разработки являются основой для выбора, предварительного планирования и системного анализа всего процесса создания программного средства. Достоверность планов и прогнозов определяется точностью сведений об объекте разработки, характеристиках технологической среды и прототипов, принятых за основу при планировании. Таким образом, производится технико-экономическое обоснование проекта, определяются приближенные значения трудоемкости и длительности всей разработки программного средства, а также число необходимых специалистов, что позволяет оценить предварительный план создания программного средства в заданных условиях, ресурсах и сроках. На этом этапе невозможно составить жесткий план их выполнения.

Проведенные оценки проекта программного средства позволяют осуществить предварительный выбор основных методов и инструментальных средств для проведения последующего детального и рабочего проектирования и поддержки всего жизненного цикла программного средства. Кроме того, должны подготавливаться адаптация средств автоматизации, применительно к особенностям объекта и среды проектирования. Разрабатываются проекты руководств для специалистов, выделяемых на данный проект, и осуществляется их обучение. Интегрированные инструментальные средства служат для формализации знаний заказчика на этапе проведения обследования, анализа и подготовки технического задания, а также для проектирования концептуальной и логической структур комплексов программ и баз данных. При этом должно активно использоваться моделирование и тестирование корректности системных решений.

В процессе системного проектирования должны определяться состав и структура технологических и эксплуатационных документов для поддержки всего жизненного цикла программного средства. Эти документы должны обеспечивать реализацию процессов жизненного цикла ПС, планирования и управления, регистрировать выполнение требуемых действий, формализовать систему качества. При этом следует подготовить требования к документации и обеспечить их реализацию, которая должна быть написана в стандартизированных терминах, допускающих единственную интерпретацию.

Системный проект программного средства новой или модернизированной ИС должен содержать достаточно полные требования к функциям и характеристикам качества комплекса программ, описание и графическое представление его архитектуры, базы данных и взаимодействия компонентов, предполагаемую модель жизненного цикла, предварительные планы последующих этапов и работ. Кроме того, в него должны входить проекты технического задания и контракта на детальное проектирование и весь жизненный цикл программного средства. Если заказчик удовлетворен результатами системного проектирования, то возможно оформление акта завершения работ и утверждение системного проекта комплекса программ с требуемыми характеристиками качества новой или модернизированной информационной системы. Для системного анализа и проектирования требуются специалисты высокой квалификации (системные аналитики), так как ничтожные ошибки наиболее сильно отражаются на эффективности и качестве всего жизненного цикла программного средства.

Технология проектирования системы – это совокупность методологий проектирования системы, а также методов и средств организации проектирования (управления процессом создания и модернизации проекта системы).

В основе технологии проектирования лежит технологический процесс, который определяет действия, их последовательность, состав исполнителей, средства и ресурсы, требуемые для выполнения этих действий. Технологический процесс проектирования системы в целом делится на совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий.

Технологии проектирования, применяемые в настоящее время, предполагают поэтапную разработку системы.

Технология проектирования определяется как совокупность трех составляющих:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

Технологические инструкции, составляющие основное содержание технологии, должны состоять из описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция, и описаний самих операций.

Технология проектирования, разработки и сопровождения должна удовлетворять следующим общим требованиям:

- поддержание полного жизненного цикла программного обеспечения;
- обеспечение гарантированного достижения целей разработки информационных систем с заданным качеством и в установленное время;
- обеспечение возможности выполнения крупных проектов в виде подсистем (т.е. возможность декомпозиции проекта на составные части, разрабатываемые группами исполнителей ограниченной численности с последующей интеграцией составных частей). Для повышения эффективности работ необходимо разбить проект на отдельные, слабо связанные по данным и функциям, подсистемы. Реализация подсистем должна выполняться отдельными группами специалистов. При этом необходимо обеспечить координацию ведения общего проекта и исключить дублирование результатов работ каждой проектной группы, которое может возникнуть в силу наличия общих данных и функций;
- возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек). Это обусловлено принципами управляемости коллектива и повышения производительности за счет минимизации числа внешних связей;
- минимальное время получения работоспособной информационной системы (сроки реализации отдельных подсистем). Реализация информационной системы в целом в короткие сроки может потребовать привлечения большого числа разработчиков, при этом эффект может оказаться ниже, чем при реализации в более короткие сроки отдельных подсистем меньшим числом разработчиков;
- возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта;
- независимость выполняемых проектных решений от средств реализации информационной системы (систем управления базами данных (СУБД), операционных систем, языков и систем программирования);
- технология должна быть поддержана комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях жизненного цикла.

Проект – это комплекс формально организованных мероприятий для достижения цели создания сложной системы с требуемыми характеристиками качества при ограниченных ресурсах.

Цель управления проектом – рациональное использование и предупреждение потери ресурсов путем сбалансированного распределения их по частным работам на протяжении всего жизненного цикла объекта с заданным качеством.

Управление проектом – это особый вид деятельности, включающий постановку задач, подготовку решений, планирование, организацию и стимулирование специалистов, контроль хода работ и использования ресурсов при создании сложных систем. Целевое управление проектами возникло из необходимости вырабатывать и реализовывать сложные системы с заданными функциями в максимально короткие сроки при ограниченных

ресурсах. Задачи целевого управления такими работами – сводить воедино усилия исполнителей (специалистов разной квалификации, подрядчиков и субподрядчиков), добиваясь, чтобы они выступали как команда, а не как разрозненная группа функциональных специалистов при создании компонентов систем. В результате должны обеспечиваться целостность системы и высокое качество решения главных задач при сбалансированном использовании ресурсов на все функциональные задачи.

Методологической базой целевого планирования и управления проектами является системный анализ, который предполагает:

- обследование объектов и среды проектирования, для предварительной формулировки целей, назначения и задач проекта;
- исследование и сопоставление альтернативных действий, которые должны приводить к достижению поставленных целей проектирования;
- сравнение альтернатив по величине достигаемого эффекта проекта в зависимости от затрат на его достижение (желательно, по показателю «эффективность / стоимость»);
- учет и анализ влияния неопределенностей характеристик альтернатив на эффект проекта.

Чтобы найти и проанализировать все разумные альтернативы, обычно недостаточно одного специалиста, и необходимо участие в системном анализе специалистов разной квалификации. Не во всех системах и задачах оказывается доступным точный количественный подход. Во многих случаях приходится ограничиваться качественным анализом свойств, факторов и их влияния на конечный результат и возможный эффект проекта. Поэтому оптимизация решений и выбора альтернатив может ограничиваться оценкой экспертов. Базой эффективного управления проектом является план, в котором задачи исполнителей частных работ должны быть согласованы с выделяемыми для них ресурсами, а также между собой по результатам и срокам их достижения.

План проекта должен отражать: рациональное сочетание целей, стратегий действий, конкретных процедур, доступных ресурсов и других компонентов, необходимых для достижения поставленной основной цели проекта с заданным качеством.

Планирование проектов должно обеспечивать компромисс между требующимися характеристиками создаваемой системы и ограниченными ресурсами, необходимыми на ее разработку и применение. По мере уточнения исходных данных об объекте разработки, внешней среде применения и ресурсах, в процессе системного анализа и проектирования возрастает достоверность планирования, которая обычно проходит следующие этапы:

- первичное прогнозирование возможных характеристик проекта на базе обобщения данных подобных прототипов ранее реализованных проектов и создание концепции проекта;
- подготовка предварительного рабочего плана выполнения этапов и частных работ с учетом допустимых затрат ресурсов на их реализацию в процессе разработки системного проекта;
- управление детализацией и реализацией плана проекта, его оперативной корректировкой и перераспределением ресурсов в соответствии с особенностями развития частей проекта;
- обобщение и накопление результатов планирования и управления конкретным проектом для использования этих данных в качестве прототипов при разработке последующих проектов.

На каждом этапе должен проводиться поиск эффективных технических и экономических решений реализации проекта. В результате процессы планирования проекта и его выполнения развиваются параллельно. Первичное прогнозирование характеристик проекта и подготовка плана при системном проектировании происходит при некоторых фиксированных исходных данных, не учитывающих динамику возможного исполнения плана. На этой стадии отсутствует оперативная обратная связь процесса реального

выполнения плана с его первичным вариантом. Важнейшая задача при разработке такого плана – минимизировать число связей и сложность взаимодействия между компонентами проекта, а также между исполнителями отдельных компонентов. При прогнозировании развития проекта оцениваются характеристики объекта и среды разработки, и выбираются наиболее подходящие, в соответствии с поставленными целями и имеющимися ресурсами.

После создания системного проекта появляется и действует динамическая обратная связь на план со стороны процесса его исполнения. Реализация проекта зависит от результатов выполнения частных работ и может требовать оперативной корректировки плана. При реализации плана определяющими являются организация, стимулирование и контроль развития проекта. Контроль обеспечивает исходные данные для координации компонентов данной организации в соответствии с планом конкретной задачи. Для этого необходимо следить за ходом исполнения проекта на всем протяжении его жизненного цикла и сравнивать запланированные и фактические результаты работ.

Контроль является функцией управления и должен иметь средства регулирования поведения отдельных личностей и коллектива проектировщиков в целом. Одновременно обеспечивается наблюдение за состоянием системы и ее характеристиками качества, что позволяет устанавливать частные компромиссы с используемыми ресурсами. Объектами контроля при этом являются:

- технические характеристики реализованных компонентов проекта, показатели качества процессов и результатов выполнения отдельных работ;
- затраты ресурсов на выполнение частных работ и реализацию компонентов проекта (трудоемкость, стоимость, время, материальные ресурсы);
- графики работ, степень их выполнения, наличие и причины отклонений реализации частных работ от планов, угроза нарушения сроков контракта.

Для интеграции усилий специалистов и эффективного использования ресурсов проекта должен выделяться руководитель, управляющий проектом - главный конструктор. Он активно участвует в планировании, организации и контроле основных внутренних и внешних организационных мероприятий, необходимых для достижения основной цели проекта. Все ресурсы и исходные данные, необходимые для эффективного выполнения проекта, управляющий получает от функциональных подразделений и специалистов. Задачи управляющего проектом состоят в прямом воздействии на подчиненных и координации их работ, стимулировании и контроле деятельности исполнителей частных работ и их взаимодействии.

Для того чтобы процесс достижения целей был рациональным, лицо, принимающее решение (управляющий), должно иметь выбор среди альтернативных действий, ведущих к цели. Наличие альтернатив и сомнения по поводу того, какая из них лучше, определяют возможность эффективного решения проблем и оптимизации путей их достижения.

Для получения, накопления и применения достоверных данных об объектах управления и альтернативах необходима информационная система обеспечения проекта. Такая информационная система представляет собой комплекс формальных и неформальных каналов обмена информацией между участниками проекта, ее накопления и обработки. Следует учитывать, что любая групповая деятельность связана со сложным комплексом неформальных отношений между исполнителями. Степень формализации может варьироваться от утверждаемых руководителями планов и подробных технических заданий до личных бесед между разработчиками. Регулярный обмен информацией позволяет осуществлять:

- сбор исходных данных о состоянии, достигнутом качестве компонентов проекта и использованных ресурсах;
- диспетчерское управление ресурсами и частными исполнителями работ;
- сравнение текущих результатов частных работ с техническими заданиями, спецификациями и планом;

- корректировку технических результатов работ, сроков и используемых ресурсов в соответствии с изменением требований в процессе развития проекта.

Таким образом, целевое управление проектами позволяет планировать, контролировать и анализировать информацию о состоянии и тенденциях изменения объекта разработки, его качестве и затраченных ресурсах. При этом непрерывно должны сохраняться основные цели проекта и главные пути ее достижения. Это позволяет рассматривать альтернативы технических решений и предотвращает от сосредоточения внимания на частных задачах или вариантах решений, которые кажутся полезными и интересными, но мало отражаются на достижении главной цели.

Результаты проектирования содержательно представляются в виде спецификации. При этом необходимо добиться высокого уровня формализации этих результатов, представляя проект в ясной форме, понятной как заказчикам, так и разработчикам. Для этого с использованием современных CASE систем строят модели подсистем систем АСОИУ.

Эти модели должны служить базой при разработке схем потоков управления и данных, описывающих процессы их обработки, и интегрироваться с отработанными моделями бизнес-процессов для комплексного исследования функционирования прототипов.

Предварительный анализ и моделирование процессов обработки данных при системном проектировании должны проходить этапы от установления базовых отношений между понятиями, через определение интерфейсов доступа и атрибутов, к проекту модели состояний и взаимодействий между реальными объектами и процессами программного средства.

При построении описания системы, выполняемом ее разработчиком, принципиальными являются два организационных момента:

- специалисты (заказчики или пользователи создаваемой системы) должны активно участвовать в процессе анализа и реализации ее описания;
- каждый шаг описания должен обязательно документироваться.

Моделирование процессов обработки данных при системном проектировании преследует две основные цели:

- моделирование проблемно-ориентированных бизнес-процессов и конкретных функциональных задач с целью исследования принципов, методов и характеристик обработки информации и принятия решений для последующего их использования в проектах информационных систем;
- моделирование архитектуры объектов и процессов, их взаимодействия, предполагаемых для применения в конкретном проекте информационной системы, без особенностей их функциональных характеристик.

Наглядными и удобными в работе являются графические представления описаний проектных решений, которые позволяют создавать прототипы программных средств. Они обеспечивают эффективную обратную связь между разработчиком и потенциальным пользователем для оценки реализации требований, корректировки функций и качества компонентов, а также форм пользовательского интерфейса. Для этого разработан целый ряд методов моделирования, структурного анализа и проектирования.

Современные инструментальные CASE-средства обеспечивают широкие возможности выбора процессов моделирования, автоматизированного анализа системных предложений и выработки первичных требований к предполагаемому проекту программного средства. Схемы потоков данных, потоков управления, сущность, связь и другие составляют комплекс удобных и гибких графических методов и средств описания систем, облегчающих взаимопонимание между разработчиками и заказчиками на разных уровнях детализации функций, качества и архитектуры программного средства.

Концепция создаваемой информационной системы должна включать предварительные требования к программному средству, основные понятия и термины. Она является первым исходным документом, согласованным с заказчиком для создания комплекса программ. На основе этого описания формируется предварительное техническое

задание на систему и ее основные компоненты. При использовании методов разработки программных средств описание системы подлежит переводу на соответствующий графический язык. Наряду с разработчиком, специалисты (заказчики или пользователи создаваемой концепции программного средства) должны активно участвовать в процессе анализа и реализации ее описания.

Одним из наиболее эффективных направлений сокращения затрат и повышения качества комплексов программ является активное использование методического, технологического, алгоритмического и программного задела предшествующих проектов.

Математические модели и прототипы различных компонентов и функций информационных систем обеспечивают возможность применять готовые решения, а также выделять и исследовать принципиально новые методы и процессы для реализации их в программном средстве.

Прототипирование позволяет наглядно представить заказчику и пользователю функции информационной системы, виды и динамику применения экранов, отчетов и форм запросов, а также откорректировать их для развития программных средств на всех этапах жизненного цикла.

Методами математического моделирования должны создаваться варианты, фрагменты и компоненты прототипа программного средства и выделяться возможные методы реализации предполагаемых функций и обеспечения их качества. Для этого следует анализировать и выбирать прототипы комплексов программ, характеристики которых наиболее близки к создаваемой версии программного средства, и которые позволили бы получить в результате объекты с необходимыми характеристиками качества. На их основе возможно прогнозировать процессы разработки и достигаемые показатели качества вновь создаваемого программного средства. Этим же целям способствует предварительное распределение ресурсов, доступных для создания проекта.

В случае, когда математическую модель создать сложно, возможно использование структурного подхода, заключающегося в ее декомпозиции модели на автоматизируемые функции. Система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты согласованы.

Все наиболее распространенные методологии структурного подхода базируются на следующих принципах:

- принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- принцип иерархического упорядочивания путем организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне;
- принцип абстрагирования, заключающийся в выделении существенных аспектов системы и отвлечения от несущественных;
- принцип формализации, заключающийся в необходимости строгого методического подхода к решению проблемы;
- принцип непротиворечивости, заключающийся в обоснованности и согласованности элементов;
- принцип структурирования данных, заключающийся в том, что данные должны быть структурированы и иерархически организованы.

CASE-технология (Computer Aided Software Engineering) представляет собой методологию проектирования информационных систем, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, производить ее анализ на всех этапах разработки и сопровождения информационных систем и разрабатывать приложения в соответствии с информационными требованиями пользователей [10, 11, 20, 22, 45]. Большинство существующих CASE-средств основано на методологиях

структурного и 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.

CASE-средства – это программные средства, поддерживающие процессы создания и сопровождения информационных систем, включая анализ и формулировку требований, проектирование прикладного программного обеспечения (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным программным обеспечением и техническими средствами образуют полную среду разработки АСОИУ.

При использовании CASE-технологий необходимо учитывать следующее:

- CASE-средства не обязательно дают немедленный эффект;
- 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.

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

- понимание ограниченности существующих возможностей и способность принять новую технологию;
- готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;
- четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

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:

- достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки программного обеспечения;
- 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 программного обеспечения;
- положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;
- приемлемый уровень отдачи от инвестиций в CASE-средства.

Структурное и функциональное проектирование ПО. SADT.

SADT (Structured Analysis and Design Technique, методология Росса) – методология структурного анализа и проектирования, основанная на графическом представлении системы разными языковыми средствами. Основные положения данной методологии зафиксированы в стандарте IDEF0.

В IDEF0 [1 – 3] система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Процесс моделирования системы в IDEF0 начинается с определения контекста, то есть наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта (области) моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно определить, что входит в систему, а что лежит за ее пределами.

Точка зрения (viewpoint) – указание на должностное лицо, с позиции которого разрабатывается модель. У модели может быть только одна точка зрения. Изменение точки зрения приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения.

Формулировка цели (purpose) выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

Область моделирования (scope) описывает круг функций системы. При формулировании области необходимо учитывать два компонента: широту и глубину. Широта подразумевает определение границ моделирования, а глубина определяет, на каком уровне детализации модель является завершенной.

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм, разработанных с определенной целью и с выбранной точки зрения.

Каждая такая диаграмма содержит работы и стрелки.

Работы или функциональные блоки (activity) обозначают поименованные процессы, функции или задачи, которые выполняются в течение определенного времени и имеют распознаваемые результаты. Название функционального блока – это глагол или глагольный оборот.

На каждой диаграмме отображается от трех до шести работ. Работы имеют доминирование – они размещаются на диаграмме по степени важности. Самой доминирующей работой диаграммы может быть либо первая из требуемой последовательности, либо планирующая или управляющая. Наиболее доминирующая работа располагается в левом верхнем углу диаграммы, наименее доминирующая – в правом нижнем. Работы на одной диаграмме нумеруются последовательно в порядке их доминирования.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелка (arrow) – направленная линия, состоящая из одного или нескольких сегментов, которая моделирует канал, передающий данные от источника (начальная точка стрелки) к потребителю (конечная точка с «наконечником»).

В IDEF0 различают пять типов стрелок:

- вход (input) – материал или информация, которые используются или преобразуются работой для получения результата. Допускается, что работа может не иметь ни одной стрелки входа. Входные стрелки рисуются как входящие в левую грань работы;
- выход (output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Выходные стрелки рисуются как исходящая из правой грани работы;

- управление (control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелки управления рисуются как входящие в верхнюю грань работы;
- механизм (mechanism) – ресурсы, которые выполняют работу. Стрелки механизма рисуются как входящие в нижнюю грань работы;
- вызов (call) – специальная стрелка, указывающая на другую модель. Стрелка вызова рисуется как исходящая из нижней грани работы.

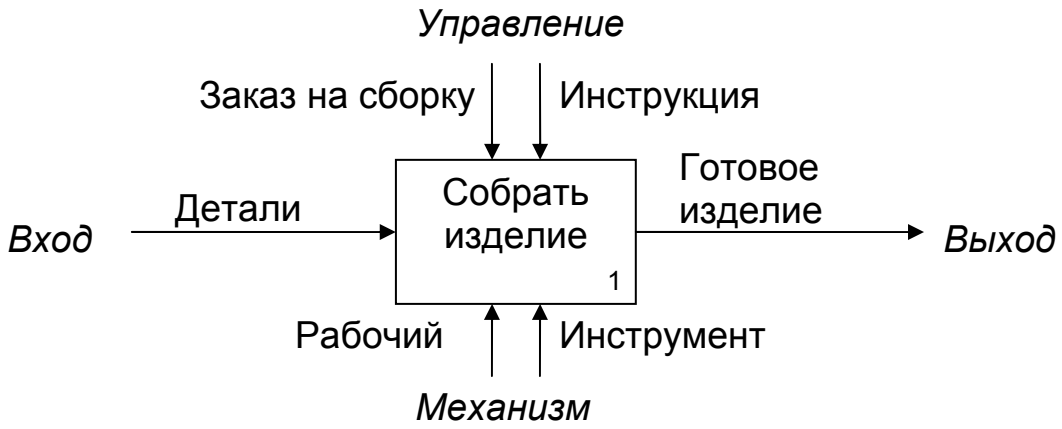


Рис. 1. Типы стрелок

Все работы на диаграмме являются преобразующими, то есть преобразуют входы в выходы под действием управлений при помощи механизмов.

Стрелка обычно представляет набор объектов. Поэтому они могут разветвляться и соединяться различными способами. Сегменты стрелок, за исключением стрелок вызова, помечаются существительным или оборотом существительного.

Внутренние стрелки используются для связи работ между собой. В IDEF0 требуется только пять типов взаимосвязей между блоками с помощью стрелок для описания их отношений: вход, управление, обратная связь по входу, обратная связь по управлению, выход-механизм (см. рис. 2).

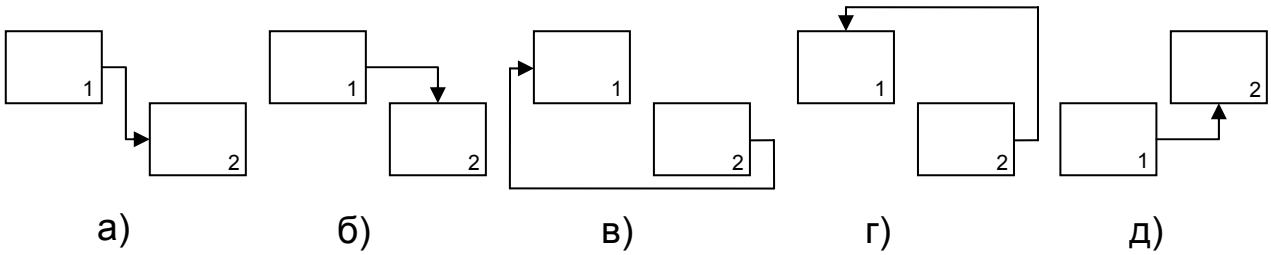


Рис. 2. Типы взаимосвязи между работами: а) вход, б) управление, в) обратная связь по входу, г) обратная связь по управлению, д) выход-механизм

Следует отметить ряд особенностей построения диаграмм в IDEF0, связанных с использованием стрелок управления:

- в связи с тем, что поступление данных на вход функционального блока само по себе не является причиной их обработки, вход всегда должен быть сопровожден соответствующим управлением;
- главное различие между управлением и входом заключается в том, что входные данные преобразуются, а управление либо регламентирует преобразование, либо инициирует его;
- в случае, когда данные входа малозначительны либо разделение входа и управления не обосновано, следует использовать управление;
- регламентирующее управление (стандарты, регламенты и т.п.) используется для определения требований к функции – для начала ее выполнения обязательно нужно инициирующее управление;

- поскольку одна диаграмма IDEF0 часто содержит описание нескольких случаев последовательного выполнения функций, рекомендуется строить диаграммы так, чтобы можно было по управлению проследить каждую последовательность.

Каждая работа (функциональный блок) может быть разбита на более мелкие работы, взаимосвязанные между собой. Этот процесс называется функциональной декомпозицией (decomposition), а диаграммы, которые описывают каждую работу (функциональный блок) и взаимодействие работ, называются диаграммами декомпозиции.

Вершиной древовидной структуры взаимосвязанных диаграмм является контекстная диаграмма, которая имеет шифр А-0. Контекстная диаграмма (context diagram) представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Контекстная диаграмма содержит только один функциональный блок, соединенный с границами диаграммы при помощи граничных стрелок.

Граничные стрелки служат для описания взаимодействия системы с окружающим миром. Идентификация граничных стрелок осуществляется с помощью ICOM-кодов, которые содержат префикс, соответствующий типу стрелки (Input, Control, Output, Mechanism).

После описания системы в целом на контекстной диаграмме в виде одного функционального блока, производится его разбиение на крупные фрагменты. Диаграмма, декомпозирующая работу на контекстной диаграмме называется диаграммой декомпозиции верхнего уровня и имеет шифр А0.

Диаграммы декомпозиции функциональных блоков диаграммы А0 в соответствии с номерами блоков, проставленных по порядку доминирования, будут иметь шифр А1, А2 и т.д. Номер каждой последующей диаграммы декомпозиции состоит из номера родительской диаграммы и номера блока (например, А121).

Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Наиболее важные свойства системы обычно выявляются на верхних уровнях иерархии и уточняются по мере декомпозиции.

Кроме контекстной диаграммы и диаграмм декомпозиции, модель IDEF0 может содержать диаграмму дерева узлов и диаграмму только для экспозиции (For Exposition Only, FEO). Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмма для экспозиции строится для иллюстрации отдельных фрагментов модели, альтернативной точки зрения, либо для специальных целей.

Рассмотрим пример модели IDEF0 для задачи выполнения заказов. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Контекстная диаграмма А-0 (см. рис. 3) показывает, что ведение учета заказов предусматривает обработку параметров заказов и данных заказчиков при поступлении новых заказов и информационных запросов, а также в случае отмены заказов. При этом система выводит сообщения об изменении заказов и сведения об обработанных заказах.

Диаграмма декомпозиции А0 (см. рис. 4) иллюстрирует ход обработки заказов. При поступлении нового заказа его параметры обрабатываются в блоке «Принимать заказы». В случае если данный заказчик обратился впервые, происходит обработка его данных в блоке «Вести учет заказчиков» по управлению «Сообщение о новом заказчике». Данные о событии «Отмена заказа» поступают в качестве управления в блок «Контролировать исполнение», в результате чего в блок «Принимать заказы» передается соответствующий запрос и данные отмененного заказа.

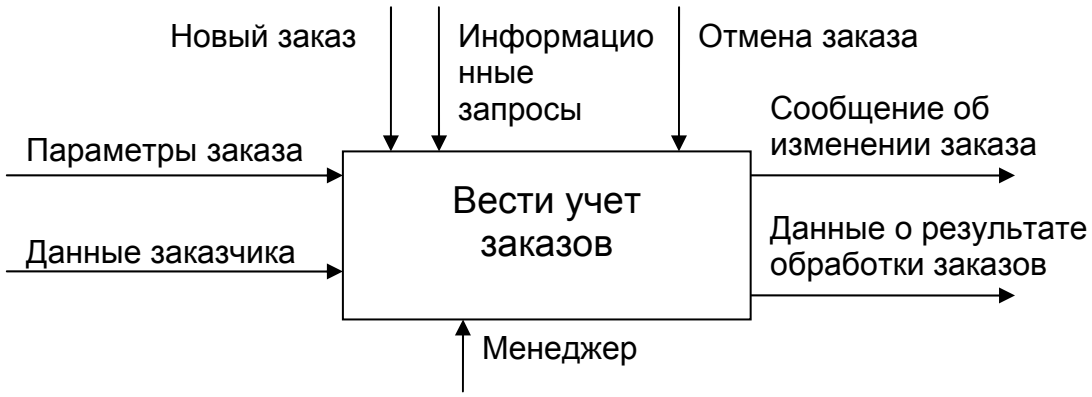


Рис. 3. Контекстная диаграмма А-0

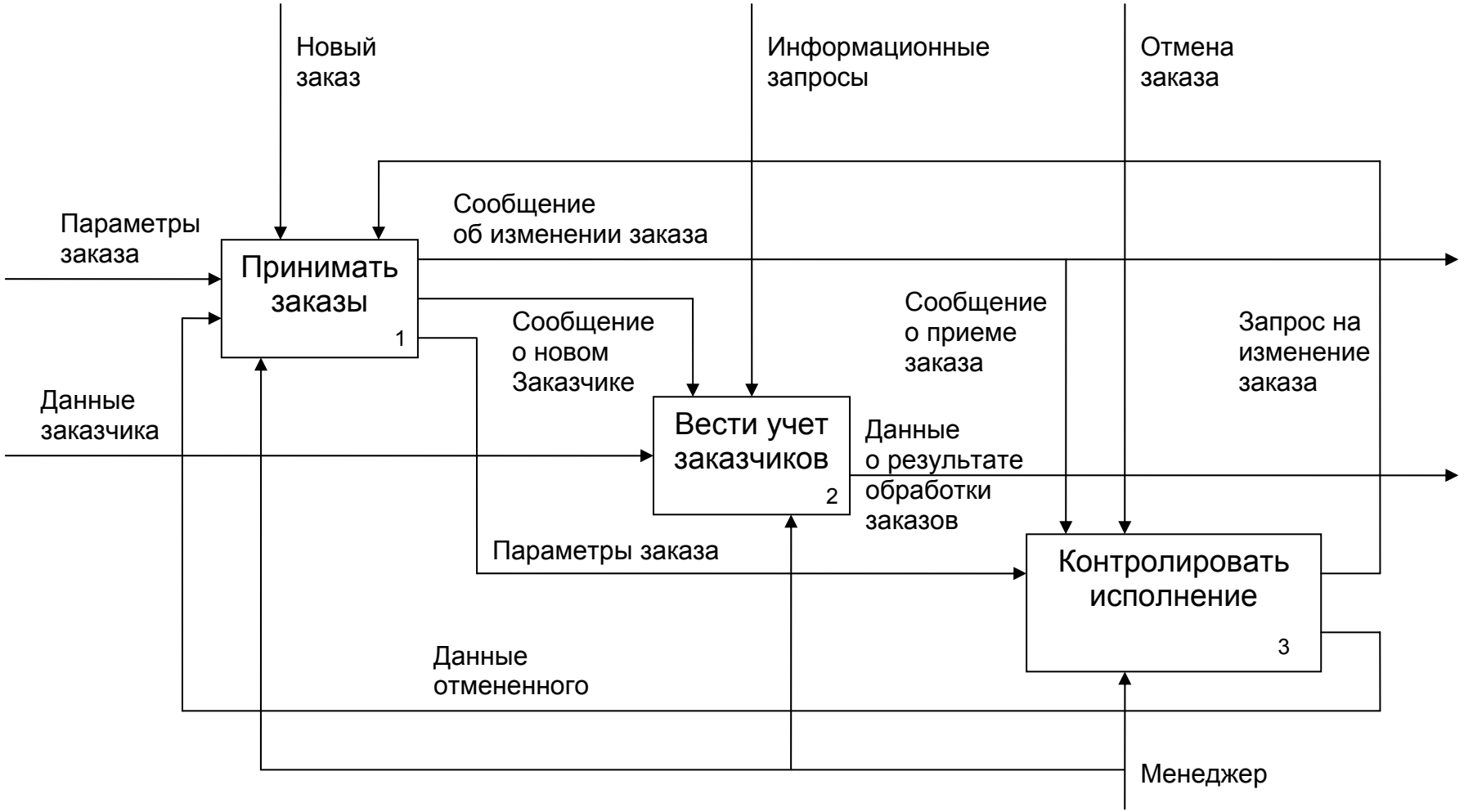


Рис. 4. Диаграмма декомпозиции верхнего уровня А0

Формализация требований к процессам и потокам данных. Описание требований к информационному обеспечению

Диаграммы потоков данных (Data Flow Diagrams, DFD) описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между бизнес-функциями, так и системы в целом с внешней информационной средой. Модель DFD [1 – 2] представляет собой набор диаграмм, отражающих различные аспекты модели, с учетом выбранной точки зрения и цели моделирования.

Работы в DFD представляют собой функции системы, преобразующие входную информацию в выходную в соответствии с действиями, задаваемыми именами работ. Каждая работа имеет уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса работы во всей модели.

Внешние сущности (ссылки) указывают на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за рамками данной модели. Внешняя сущность является источником или приемником данных извне модели. Внешние сущности обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах с целью повышения наглядности.

Потоки данных используются для моделирования передачи информации (или физических компонентов) из одной части системы в другую. Потоки изображаются на диаграмме именованными стрелками, ориентация которых указывает направление движения информации. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы.

В DFD также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями. В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя. В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой.

Хранилище данных – это место накопления информации внутри системы. Хранилища данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между работами. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое.

Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Хранилище данных может содержать информацию длительного хранения или временную информацию. Имя хранилища должно идентифицировать его содержимое. На одной диаграмме может присутствовать несколько копий одного и того же хранилища данных.

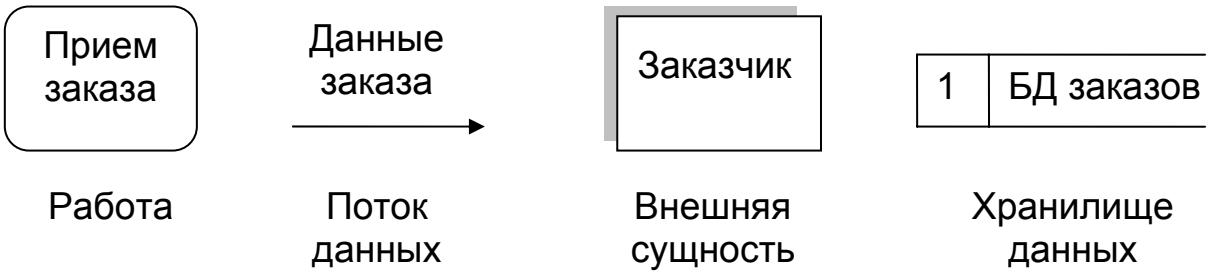


Рис. 1. Графические примитивы DFD

Построение диаграмм DFD производится «сверху вниз» в соответствии с целью моделирования и точкой зрения. Сначала строится диаграмма контекста, описывающая

исследуемую систему целиком. Затем следует диаграмма первого уровня для описания наиболее работ. При этом внешние сущности дублируются на диаграмме декомпозиции, а работы разбиваются. Дальнейшее уточнение может при необходимости производиться с помощью более подробных диаграмм на следующих уровнях.

Результатом моделирования является набор диаграмм, отражающих различные аспекты модели, которая объединяет в себе одну или несколько диаграмм потоков данных.

Пример модели DFD для задачи выполнения заказов приведен на рис. 6 – 7. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Данная модель иллюстрирует процесс выполнения заказа. После получения заказа производится сохранение его параметров, выбор соответствующего товара и выставление счета. После получения данных об оплате счета, производится отгрузка товара. Ведение базы данных товаров производится менеджером. Директору формируется отчет об обслуживании заказчиков.

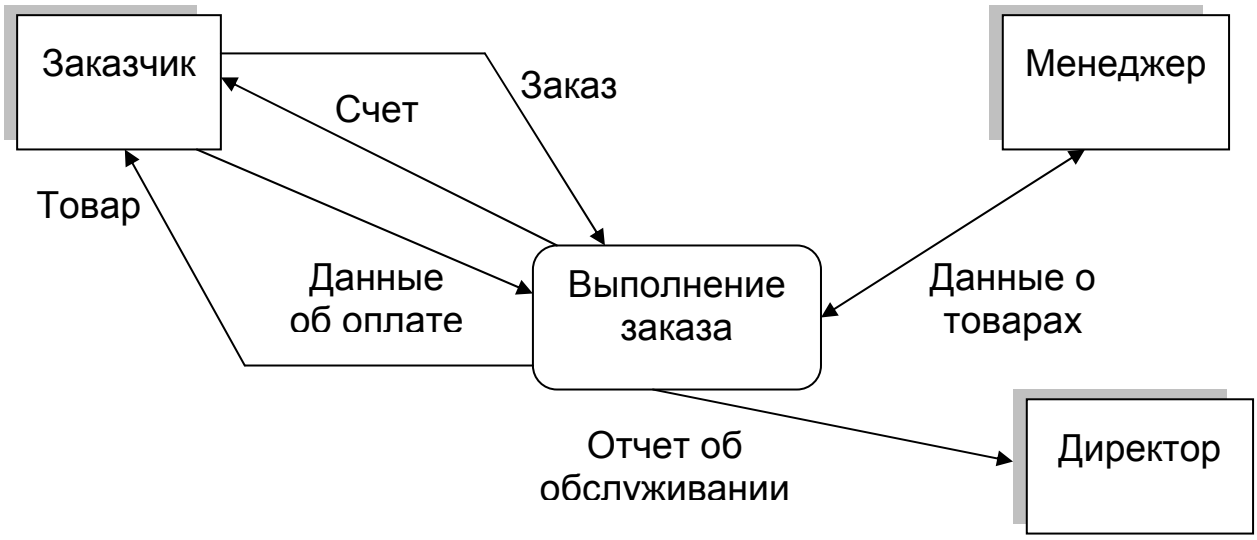


Рис. 2. Диаграмма контекста DFD

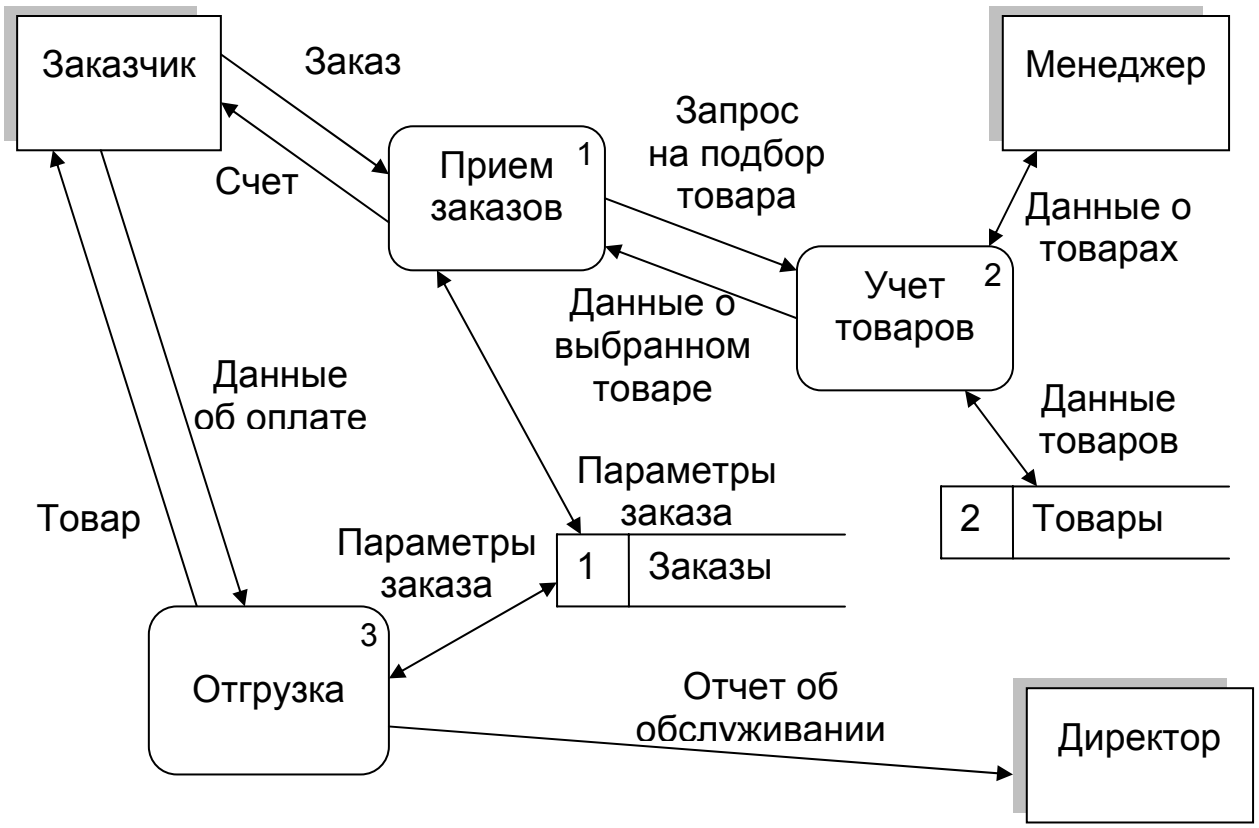


Рис. 3. Диаграмма декомпозиции DFD

Методология моделирования IDEF3 (workflow diagramming) предназначена для описания логики взаимодействия работ и последовательности их выполнения. Диаграммы IDEF3 [1 – 2] позволяют сфокусировать внимание на течении процессов и на отношениях процессов и важных объектов, являющихся частями этих процессов.

Каждая диаграмма в IDEF3 описывает какой-либо сценарий бизнес-процесса – последовательность изменений свойств объекта, которые необходимо выполнить за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования. Моделирование начинается с формулировки точки зрения, цели и области моделирования.

Единицы работы (Unit of Work, UOW), также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор). Другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее идентификатор не должен вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи:

- связь предшествования (Precedence) – показывает, что прежде чем начнется работа-приемник, должна завершиться работа-источник;
- связь отношения (Relational) – показывает связь между двумя работами или между работой и объектом ссылки;

- поток объектов (Object Flow) – показывает участие некоторого объекта в двух или более работах, как, например, если объект производится в ходе выполнения одной работы и потребляется другой работой.

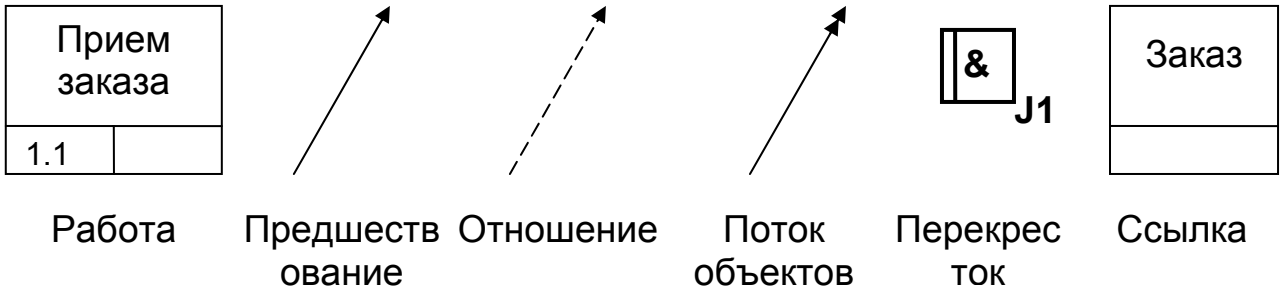


Рис. 4. Графические примитивы IDEF3

Имя стрелки должно ясно идентифицировать отображаемый объект. Связь предшествования показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Если результатом работы-источника становится объект, необходимый для запуска работы-цели, используется стрелка потока объектов. Отношение показывает, что работа-источник не обязательно должна закончиться, прежде чем работа-цель начнется; более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Рис. 5. Временная диаграмма последовательности выполнения работ

Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Перекрестки применяются, когда окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ.

Различают два типа перекрестков:

- перекресток слияния (Fan-in Junction) – узел, собирающий множество стрелок в одну, указывая на необходимость условия завершенности работ-источников стрелок для продолжения процесса;
- перекресток ветвления (Fan-out Junction) – узел, в котором единственная входящая в него стрелка ветвится, показывая, что работы, следующие за перекрестком, выполняются параллельно или альтернативно.

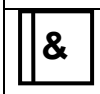
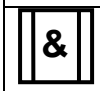
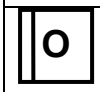
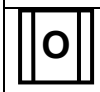
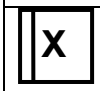
В IDEF3 стрелки могут сливаться и разветвляться только через перекрестки. Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Перекресток не может использоваться одновременно для слияния и для разветвления.

На одной диаграмме IDEF3 может быть создано несколько перекрестков различных типов. Определенные сочетания перекрестков для слияния и разветвления могут приводить к

логическим несоответствиям. Чтобы избежать конфликтов, необходимо соблюдать следующие правила:

1. Каждому перекрестку для слияния должен предшествовать перекресток для разветвления;
2. Перекресток, имеющий одну стрелку на одной стороне, должен иметь более одной стрелки на другой.
3. Перекресток для слияния «И» не может следовать за перекрестком для разветвления типа синхронного или асинхронного «ИЛИ», типа исключающего «ИЛИ»;
4. Перекресток для слияния типа исключающего «ИЛИ» не может следовать за перекрестком для разветвления типа «И»;

Таблица 2. Типы перекрестков

Обозначение	Наименование	Пояснение	
		при слиянии стрелок	при разветвлении стрелок
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Объект ссылки (Referent) в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. В качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки.

Методология IDEF3 позволяет декомпозировать работу многократно, то есть работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Декомпозиция может быть сценарием или описанием. Описание включает все возможные пути развития процесса. Сценарий является частным случаем описания и иллюстрирует только один путь реализации процесса.

Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы может состоять из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме. Для описания номер декомпозиции равен единице. Для сценария номер декомпозиции всегда больше единицы.

При создании сценария или описания необходимо придерживаться дополнительных ограничений. В сценарии или описании может существовать только одна точка входа. За

точкой входа следует работа или перекресток. Для описания может существовать только одна точка выхода. Сценарий может иметь несколько точек выхода.

IDEF3 позволяет внести информацию в модель различными способами. Например, логика взаимодействия может быть отображена графически в виде комбинации перекрестков. Та же информация может быть отображена в виде объекта ссылки типа ELAB в случае, если комбинация перекрестков занимает значительное место и затрудняет расположение работ на диаграмме.

Таблица 3 – Типы объектов ссылки

Тип	Цель описания
OBJECT	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток
UOB (Unit of behavior)	Применятся, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа «Контроль качества» может быть использована в процессе «Изготовления изделия» несколько раз, после каждой единичкой операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elabora-tion)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на перекрестках

Пример модели IDEF3 для задачи выполнения заказов приведен на рис. 6. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Данная диаграмма иллюстрирует процесс обработки заказа. После того, как заказ принят, происходит сохранение данных о заказе и заказчике (если этой информации в базе данных нет). После чего производится выбор товара, и в случае оплаты – отгрузка, а в противном случае – отмена заказа.

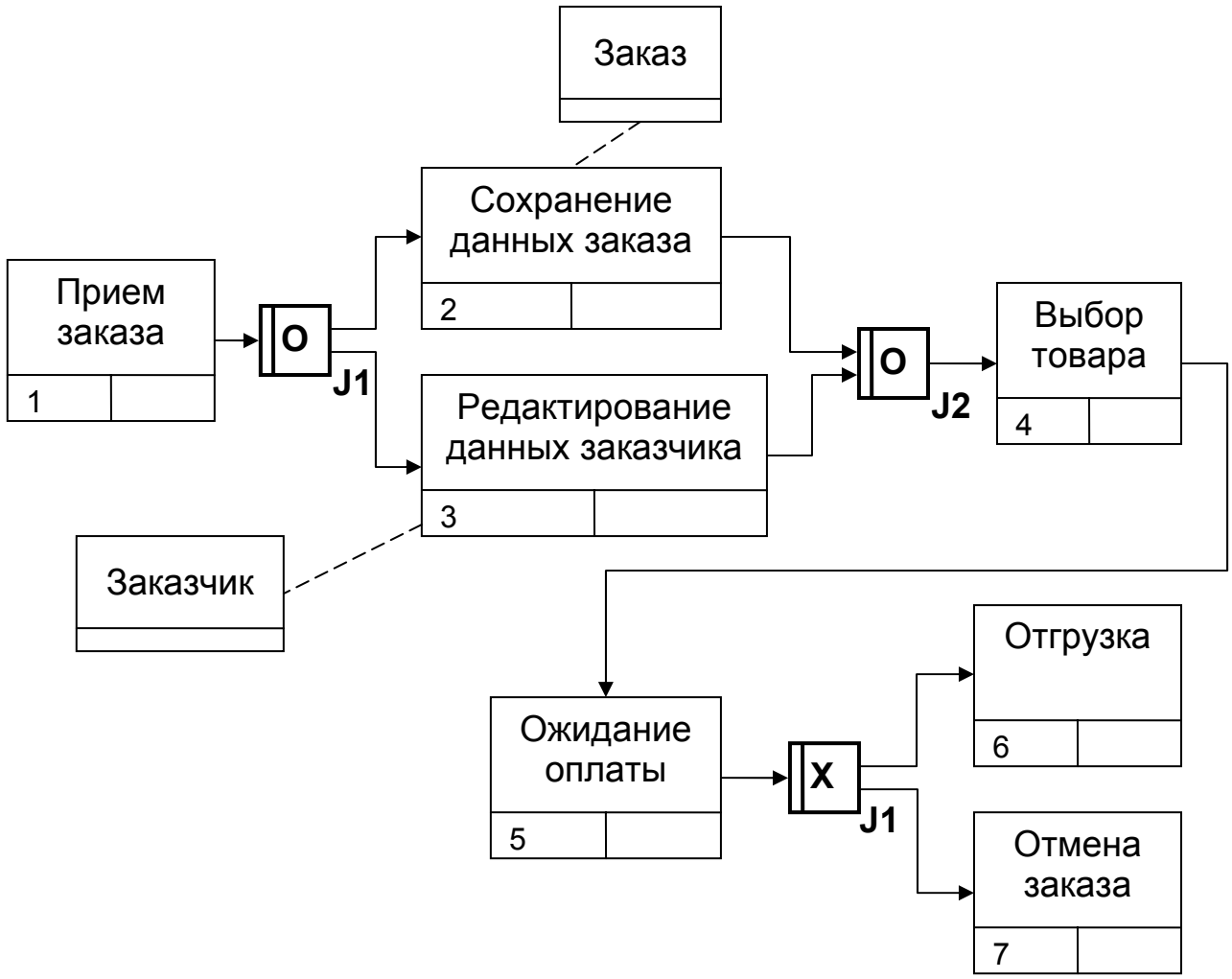


Рис. 6. Процесс IDEF3

Язык UML. Общие принципы проектирования. Диаграммы вариантов использования и классов

Язык UML (Unified Modeling Language) представляет собой унифицированный язык визуального моделирования, который разработан для специфицирования (создания спецификации), конструирования, визуализации, и документирования компонентов программного обеспечения и бизнес-процессов. Язык UML [4 – 9] может быть использован для построения концептуальных и логических моделей сложных систем самого различного целевого назначения.

Конструктивное использование языка UML основывается на применении общих принципов объектно-ориентированного анализа и проектирования:

- принцип абстрагирования, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций;
- принцип многомодельности, который представляет собой утверждение о том, что никакая единственная модель не может с достаточной степенью адекватности описывать различные аспекты сложной системы. При этом наиболее общими представлениями сложной системы принято считать статическое (структурное) и динамическое (описание логики процессов) представления;
- принцип иерархического построения моделей сложных систем, который предписывает рассматривать процесс построения модели на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.

Процесс проектирования заключается в построении модели, записанной в графической нотации. При этом соблюдаются общие принципы структурного проектирования: нисходящая разработка, иерархическое построение модели, строгая формализация и четкая семантика.

Представления о модели сложной системы фиксируются на языке UML в виде специальных графических конструкций – диаграмм:

- вариантов использования (use case);
- классов (class);
- поведения (behavior):
  - состояний (state chart);
  - деятельности (activity);
  - взаимодействия (interaction):
    - последовательности (sequence);
    - кооперации (collaboration);
- реализации (implementation):
  - компонентов (component);
  - развертывания (deployment).

Диаграмма вариантов использования описывает функциональное назначение системы. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействие которых с системой отображается в виде взаимосвязанных вариантов использования.

Актером (Actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему. Вариант использования (Use case) служит для описания сервисов, которые система предоставляет актеру или прецедентов использования системы.

Взаимодействие экземпляров актеров и вариантов использования между собой описывается с помощью отношений:

- ассоциации – служит для обозначения специфической роли актера в отдельном варианте использования;

- включения – указывает, что некоторая последовательность поведения одного варианта использования включает в качестве составного компонента определенное поведение другого варианта использования (отношение определяется стрелкой от включающего к включаемому);
- расширения – отмечает тот факт, что один из вариантов использования может присоединить к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования (отношение определяется стрелкой от расширяющего к расширяемому);
- обобщения – применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями родительских вариантов и участвуют во всех отношениях родительских вариантов (отношение определяется стрелкой от потомка к предку). Дочерние варианты использования могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения;

Пример диаграммы вариантов использования описан на рис. 1. Вариант использования по приему заказа в обязательном порядке включает ввод параметров, поэтому используется отношение «include». Выбор специальной формы оплаты возникает только при необходимости, это расширяющий вариант использования и применяется отношение «extend». Прием VIP заказа – это специальный случай приема заказа, имеющего высокую важность.

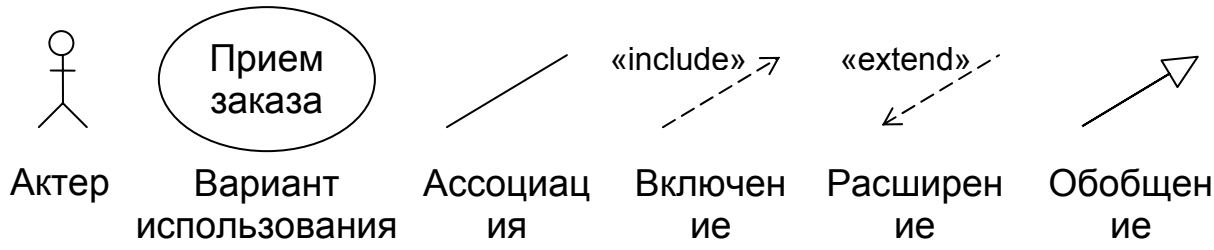


Рис. 1. Графические примитивы диаграммы вариантов использования UML

Диаграмма классов служит для представления статической структурной модели системы в терминологии классов объектно-ориентированного проектирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений. При этом на данной диаграмме не указывается информация о временных аспектах функционирования системы.

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Описание класса состоит в определении атрибутов (свойств) и методов (операций или сервисов).

Интерфейс в языке UML – это семантическая и синтаксическая конструкция, используемая для специфицирования методов класса.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений:

- ассоциации – соответствует наличию некоторого отношения между классами;
- агрегации – частный случай ассоциации, когда один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности;
- композиции – частный случай агрегации, при котором выделяется специальная форма отношения «часть-целое», при которой составляющие части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его части;
- обобщения – отношение между более общим элементом (родителем или предком) и более частным и специальным элементом (дочерним или потомком).

- зависимости – указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, выраженное в том, что некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели;

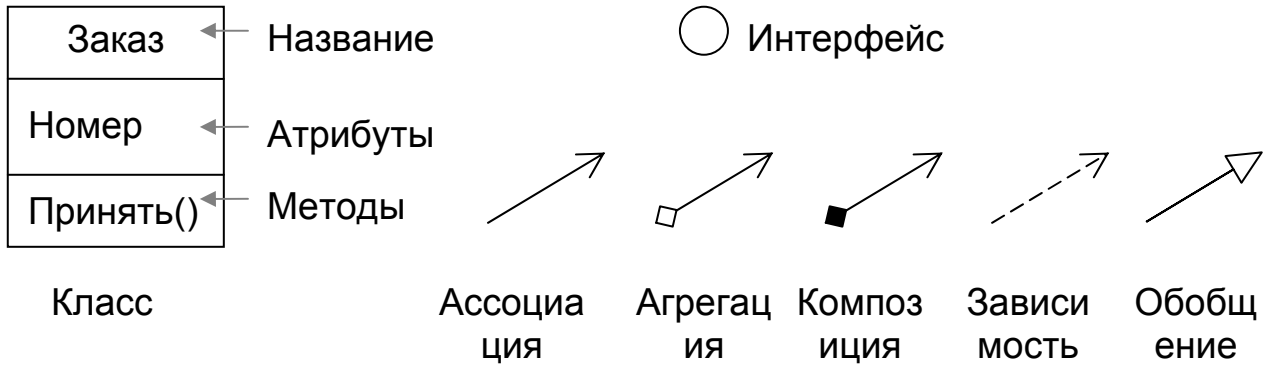


Рис. 2. Графические примитивы диаграммы классов UML

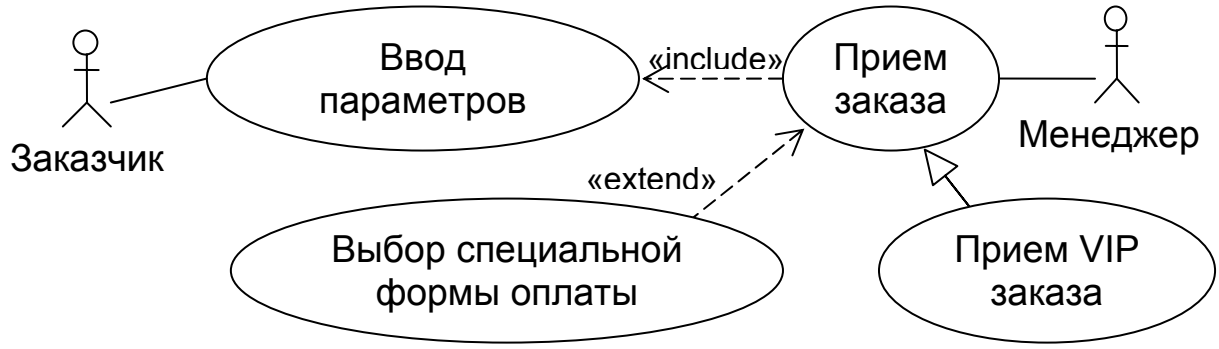


Рис. 3. Пример диаграммы вариантов использования UML

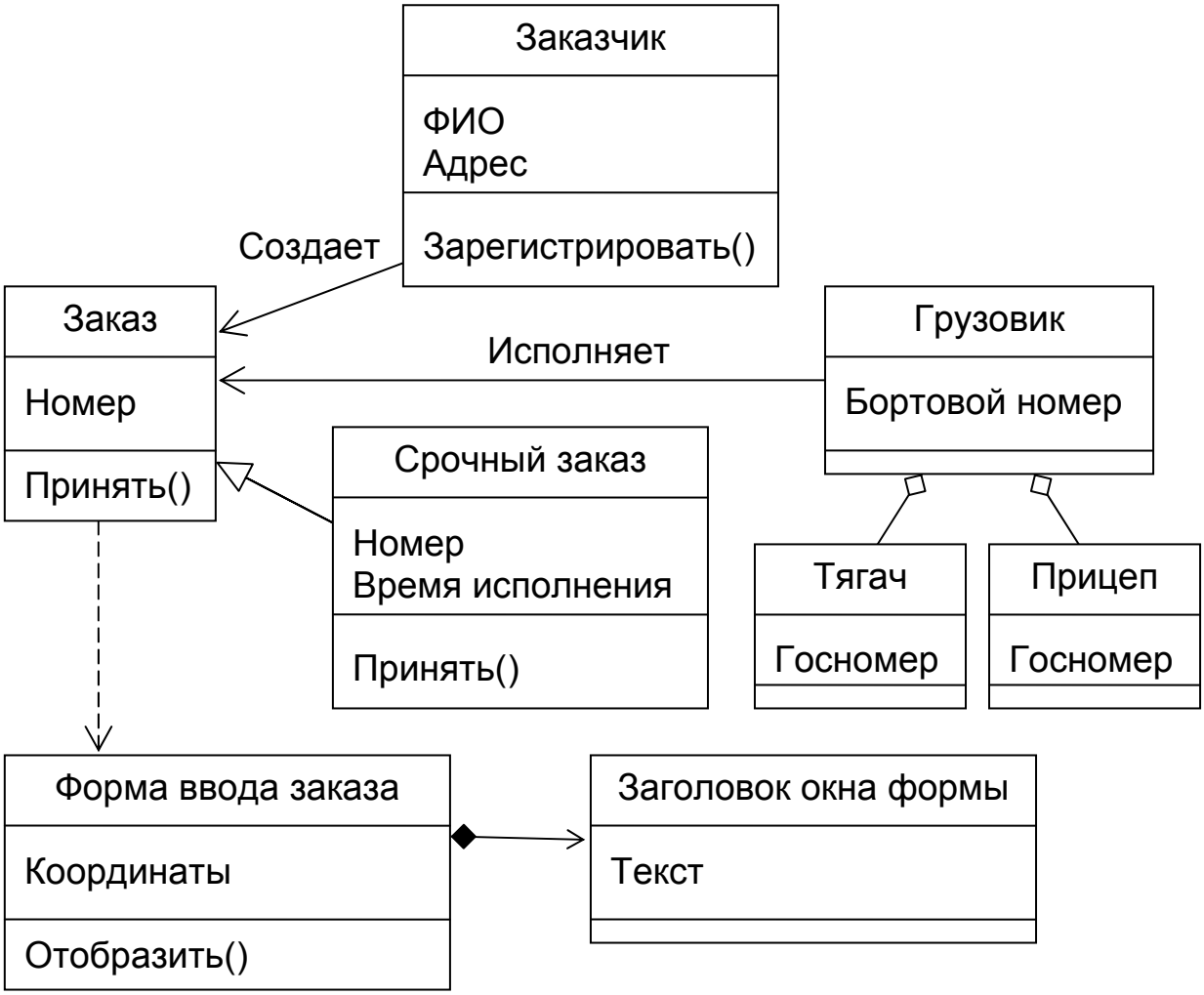


Рис. 4. Пример диаграммы классов UML

Рис. 4 иллюстрирует пример диаграммы классов. Заказчик задает заказ, а грузовик его исполняет, что описано с помощью соответствующих сущностных классов, связанных ассоциацией. Срочный заказ наследует все атрибуты обобщенного заказа и имеет свой атрибут – время исполнения. Форма ввода заказа позволяет вводить его параметры. Грузовик – это сущность, содержащая тягач и прицеп, которые, впрочем, могут выступать в системе и независимо. Заголовок окна формы существовать в отрыве от формы не может – поэтому используется композиция.

Язык UML. Разработка модели ПО для мобильных устройств, учитывающих требования Заказчика

Диаграмма состояний описывает процесс изменения состояний одного или нескольких экземпляров классов, т. е. моделирует все возможные изменения в состоянии конкретного объекта, которые вызваны внешними воздействиями со стороны других объектов или извне. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Главное назначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, которые отражают динамический или функциональный аспект его поведения. При этом изменение их отдельных значений будет отражать изменение состояния. Состояние определяется именем и списком внутренних действий (деятельностей), которые выполняются в процессе нахождения моделируемого элемента в данном состоянии и характеризуются меткой действия (entry, exit, do, include).

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершины этого графа – состояния и некоторые другие типы элементов автомата (псевдосостояния), отображаемые соответствующими графическими символами.

Дуги графа служат для обозначения переходов из состояния в состояние. Срабатывание перехода зависит от наступления некоторого события и от выполнения определенного условия, называемого сторожевым. На базе простых переходов возможна организация сложных – соединение (две и более входящих дуг) и ветвление (две и более исходящих дуг). Диаграммы состояний могут быть вложены друг в друга, образуя составные состояния, которые могут быть последовательными, параллельными и историческими – т.е. запоминающими; синхронизирующими.

Существует два частных случая состояния. В начальном состоянии объект находится по умолчанию в начальный момент времени. В конечном состоянии объект находится после завершения работы автомата в конечный момент времени. Эти состояния не содержат никаких внутренних действий (псевдосостояний).

Рис. 1. Графические примитивы диаграммы состояний UML

Диаграммы деятельности позволяют описать особенности процедурного и синхронного управления, обусловленного выполнением внутренних деятельностей и действий (элементарных операций). Диаграмма деятельности является специальным видом диаграммы состояний, на которой для отображения действий введены специальные состояния, а на переходах отсутствует сигнатура событий. При этом на этой диаграмме допускается использование и обычных состояний.

Основным направлением использования диаграмм деятельности является визуализация особенностей реализации методов классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции некоторого класса либо ее части.

Символы ветвления используется для описания условных переходов. Разделение (concurrent fork) и слияние (concurrent join) используются для разделения и слияния параллельных вычислений или потоков управления. Дорожки (swim lanes), изображенные на рис. 21 позволяют разделять выполнение процессов между классами или объектами (а также подразделениями в случае описания бизнес-процесса с помощью диаграммы деятельности).

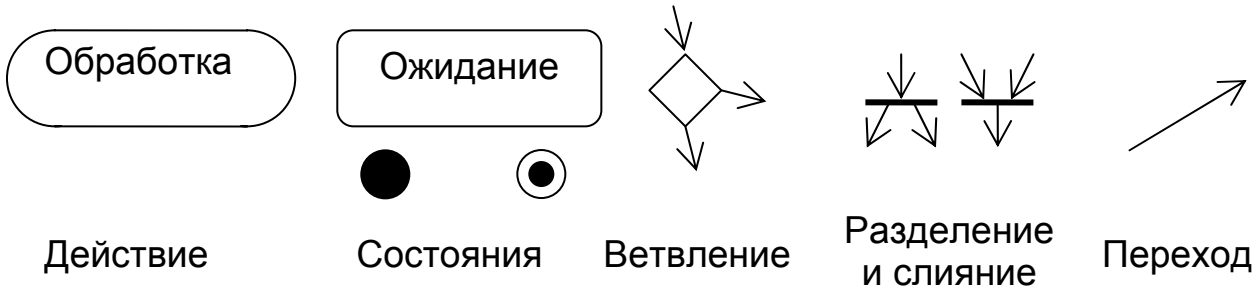


Рис. 2. Графические примитивы диаграммы деятельности UML

На рис. 3 приведен пример диаграммы состояний для описания изменения Заказа. В случае, когда заказ принят и может быть исполнен, он принимается к исполнению, прерывание которого возможно в случае отмены или успешного завершения.

На рис. 4 приведена диаграмма деятельности системы по приему заказа. Интерфейсный модуль позволяет вводить параметры заказа и отображать результат его исполнения. Система обработки заказа проводит проверку на выполнимость, а система планирования назначает исполнителя.

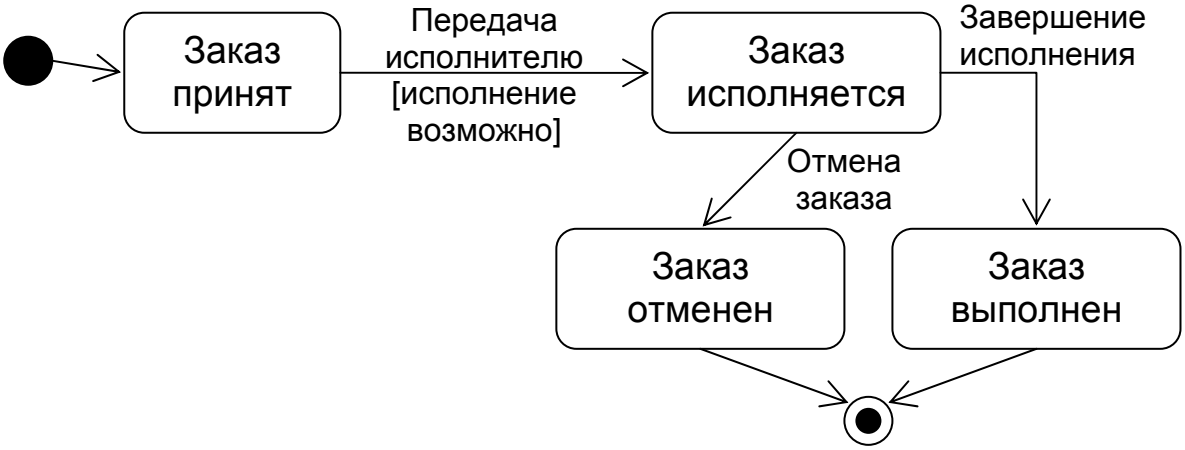


Рис. 3. Пример диаграммы состояний UML.

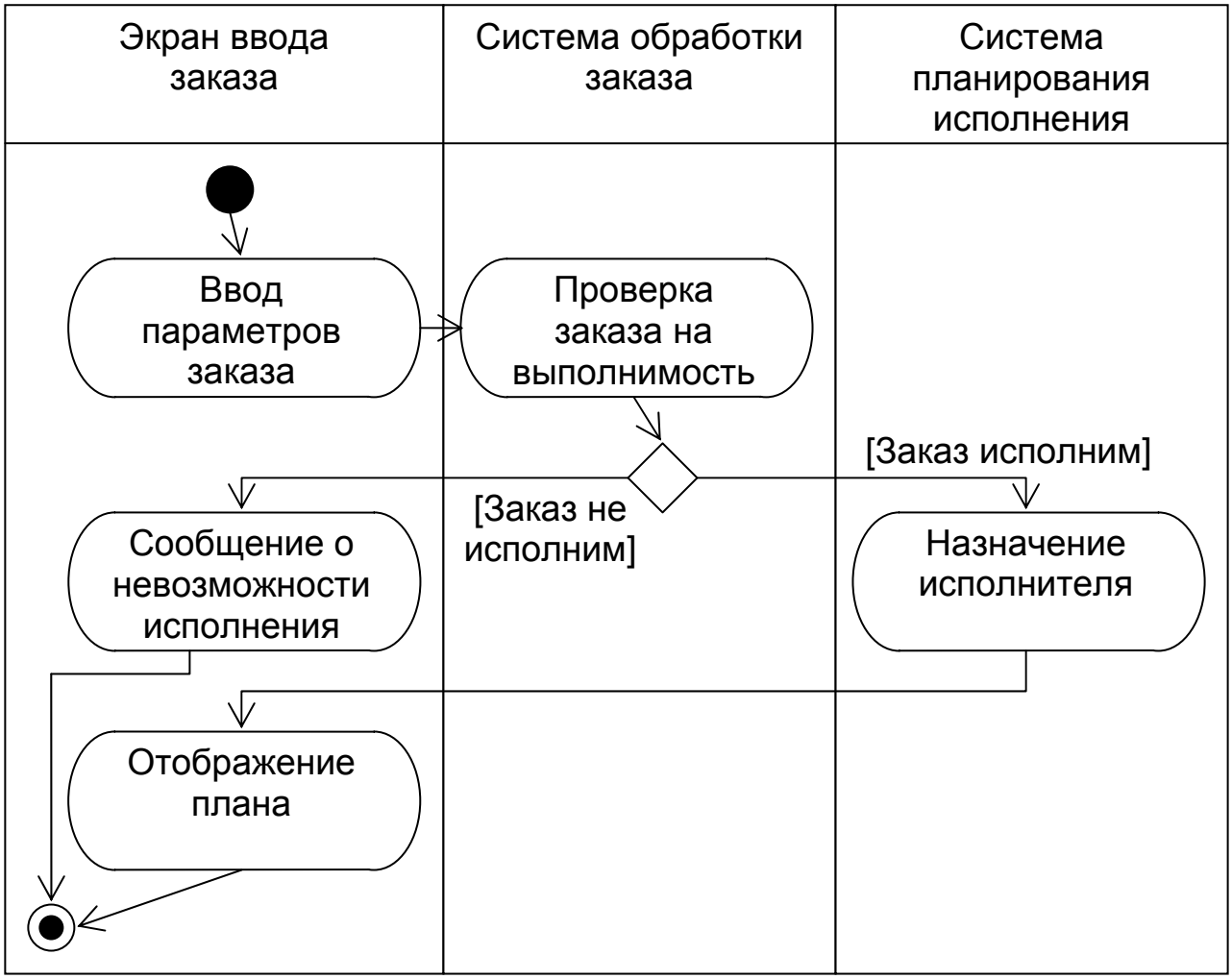


Рис. 4. Пример диаграммы деятельности UML

Диаграмма последовательности отображает взаимодействие объектов – экземпляров классов во времени, выраженное в обмене сообщениями. При этом возможные статические ассоциации между объектами не показываются. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени (вертикальная временная ось направлена сверху вниз).

Линия жизни объекта, изображаемая в виде вертикальной линии, служит для обозначения периода времени, в течение которого объект активен, то есть участвует во взаимодействии.

Взаимодействия объектов реализуются посредством сообщений, которые образуют порядок по времени своего возникновения. Сообщение – законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

В языке UML могут встречаться несколько разновидностей сообщений:

- вызов процедур, выполнение операций или обозначение отдельных вложенных потоков управления;
- обозначение простого (не вложенного) потока управления;
- асинхронное сообщение между двумя объектами в некоторой процедурной последовательности;
- возврат из вызова процедуры.

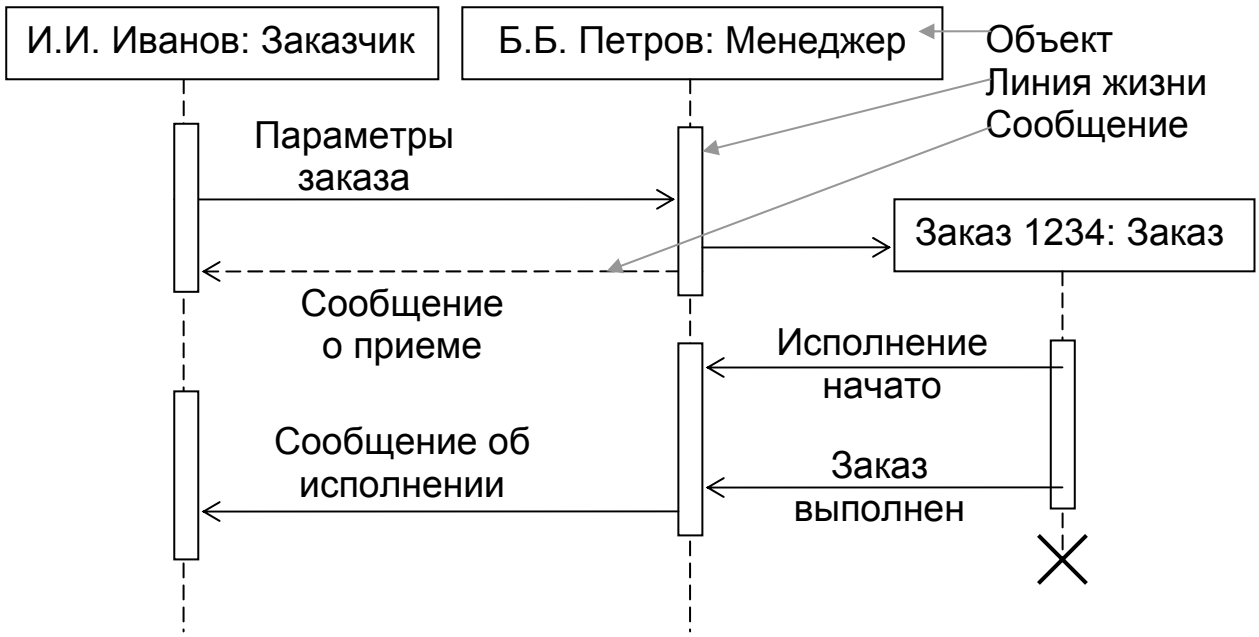


Рис. 5. Графические примитивы диаграммы последовательности UML и пример диаграммы

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов, отображаются в виде прямоугольников. Ассоциации между объектами указываются в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи – потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

Сообщение на диаграмме кооперации специфицирует коммуникацию между двумя объектами, один из которых передает другому некоторую информацию. При этом первый объект ожидает, что после получения сообщения вторым объектом последует выполнение некоторого действия.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. С другой стороны, на этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров.

Кооперация может быть представлена на двух уровнях:

- на уровне спецификации – показывает роли классификаторов и роли ассоциаций в рассматриваемом взаимодействии;
- на уровне примеров – указывает экземпляры и связи, образующие отдельные роли в кооперации.

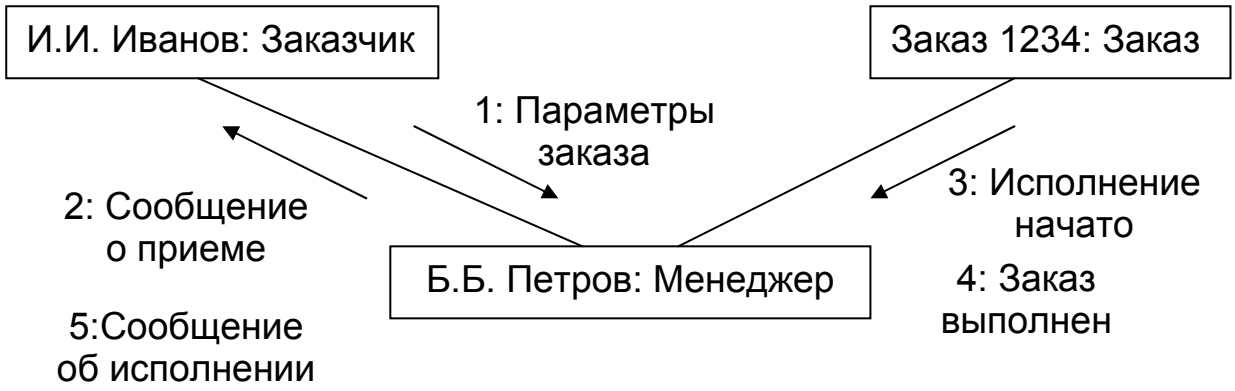


Рис. 6. Графические примитивы диаграммы кооперации UML и пример диаграммы

Диаграмма компонентов описывает особенности физической реализации системы в момент перехода от логического представления к конкретной реализации системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

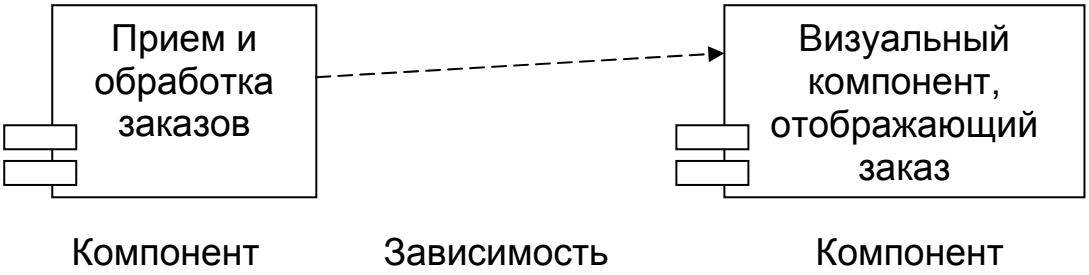


Рис. 7. Графические примитивы диаграммы компонентов UML и пример диаграммы

Диаграмма развертывания (размещения) применяется для представления общей конфигурации и топологии распределенной автоматизированной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

На рис. 25 представлен классический пример трехзвенной архитектуры автоматизированной системы в виде диаграммы развертывания UML.

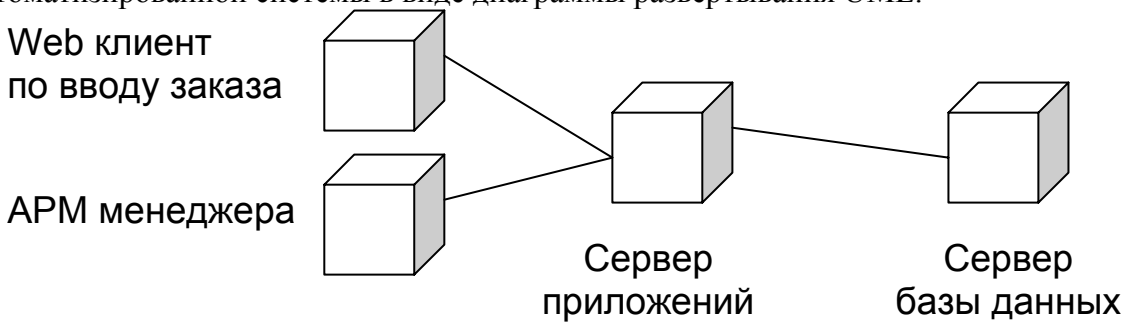


Рис. 8. Графические примитивы диаграммы развертывания UML и пример диаграммы

Процессно-ориентированный подход. ARIS. BPMN

Методология ARIS (Architecture of Integrated Information Systems – архитектуры интегрированных информационных систем) состоит в описании модели процессов [10] в виде четырех интегрированных между собой представлений:

- функциональное представление содержит описание функций бизнес-процесса или системы, отдельных подфункций (операций) и их взаимосвязи между собой;
- информационное представление описывает состояния информационных объектов (данных), и события, приводящие к их изменению;
- организационное представление определяет совокупность организационных единиц и их взаимосвязей;
- управляющее представление описывает взаимосвязи между указанными представлениями.

Каждое представление содержит разные диаграммы (ARIS поддерживает разнообразные графические нотации), которые по времени их возникновения относят к трем последовательным уровням или этапам проработки представления (см. Таблицу 4):

- на уровне описания требований происходит определение целей моделирования, языка предметной области и программного решения рассматриваемой задачи, которое базируется на результатах анализа проблем бизнеса и позволяет описать формализованные требования к системе;
- на уровне спецификации проекта концептуальные понятия, сформулированные на предыдущем уровне формулировки требований, трансформируются в категории, методы и алгоритмы в терминах информационных технологий;
- на уровне описания реализации спецификация проекта трансформируется в конкретные аппаратные и программные компоненты.

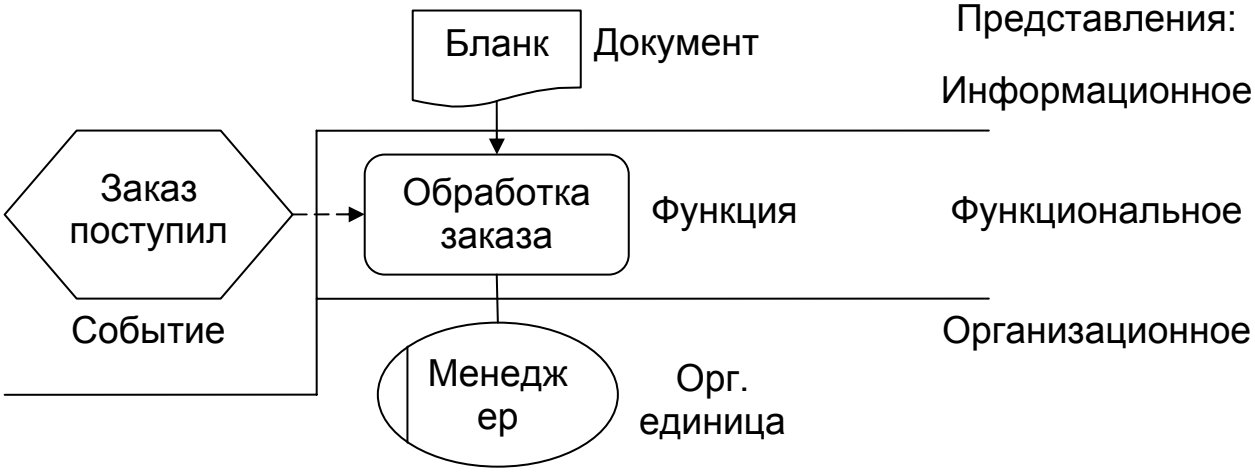


Рис. 1. Представления ARIS и базовые графические примитивы

Таблица 1. Уровни представлений ARIS

Уровни проработки	**Представления**		
	Функциональное	**Информационное**	**Организационное**
Описание требований	Иерархия функций в виде дерева	ER модель в нотации Чена (Сущности, Связи и Атрибуты)	Иерархия организационных единиц в виде организационной схемы (Должности, Типы сотрудников, Сотрудники)
Спецификация	Иерархия типов прикладных систем и	Реляционная диаграмма и	Топология сети (Узлы и компоненты

	типов модулей, реализующих функции	диаграмма атрибутов (Отношения и Атрибуты)	оборудования)
Реализация	Диаграмма прикладной системы, отражающей фактическую реализацию	Табличная диаграмма (Таблицы и Поля)	Диаграмма сети с учетом конкретного местоположения

Управляющее представление содержит нотации VAD и eEPC/PCD, которые описаны ниже.

Моделирование АСОИУ по методологии ARIS может содержать следующие этапы:

1. Определение цели моделирования и точки зрения на проект.
2. Построение диаграммы VAD и определение управляющей модели на высоком уровне.
3. Построение диаграмм eEPC/PCD, описывающих логику бизнес-процессов в виде взаимодействующих событий, функций, информационных объектов и организационных единиц.
4. Построение иерархий функций, данных и организационных единиц на уровне описания требований путем обобщения или уточнения абстракций, появившихся на диаграммах VAD и eEPC/PCD.
5. Проработка функционального, информационного и организационного представлений на уровнях спецификации и реализации.
6. Корректировка eEPC/PCD и VAD диаграмм по результатам проработки функционального, информационного и организационного представлений.

Диаграммы VAD (Value Added Chain Diagrams) описывают цепочки процессов, добавляющих стоимость, и используются для представления процесса на верхнем уровне. Процессы, или некоторые группы функций соединяются между собой пунктирной стрелкой, которая имеет тип «is predecessor of» (является предшественником). При этом возможно существование обратной связи.

Между процессами отображаются потоки материальных ресурсов и информации. Для описания инфраструктуры, необходимой для выполнения процесса, используются организационные единицы и ресурсы (типа «Product Service» и «Information Service»). Связь с процессами обозначается сплошной линией, при этом может быть указан тип связи.

Пример диаграммы VAD для процесса обработки заказа представлен на рис. 2. Указанные процессы добавляют стоимость в смысле обеспечения прибыльности. Между процессами передаются информационные и материальные объекты.

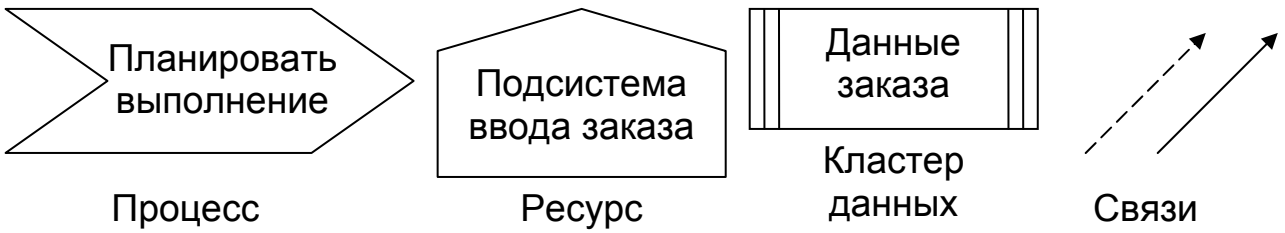


Рис. 2. Специфичные графические примитивы VAD диаграмм

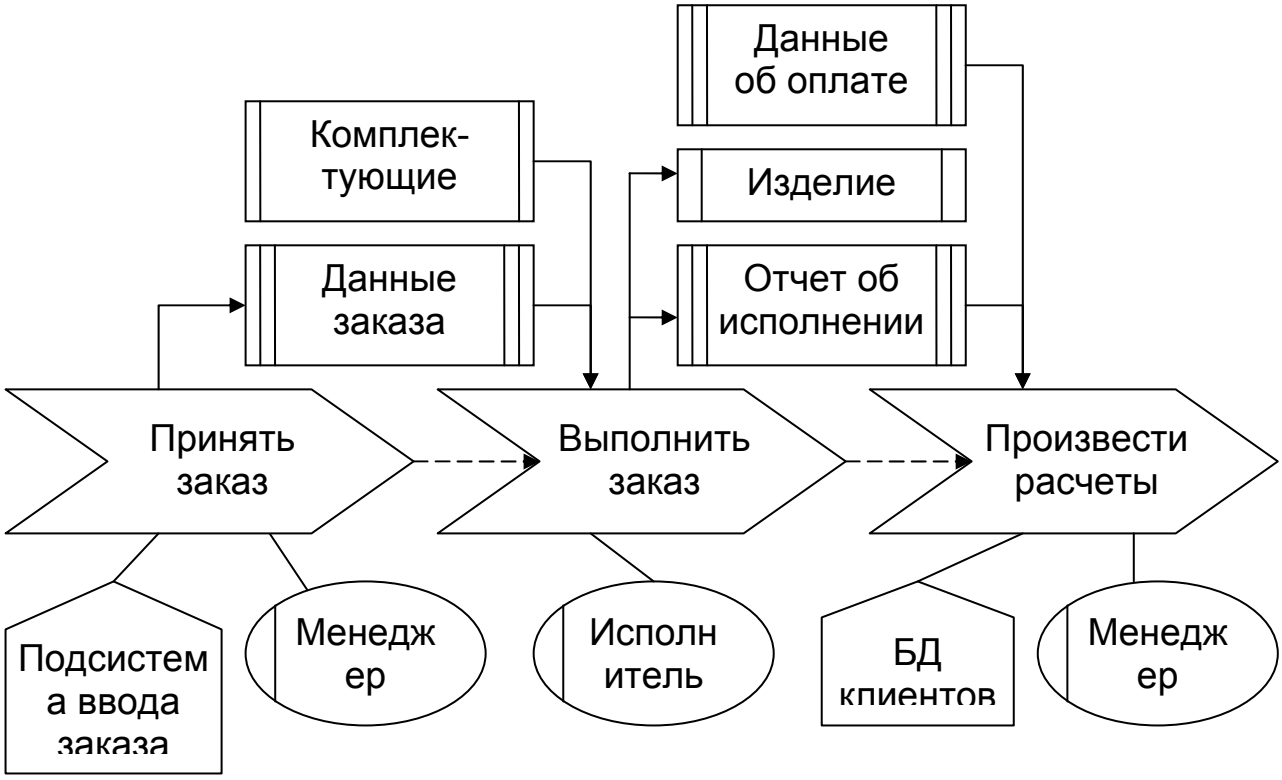


Рис. 3. Пример диаграммы VAD

Расширенные цепочки процессов, управляемых событиями (eEPC – Extended Event Driven Process Chains) предназначены для описания сложных процессов и содержат события, функции, информационные объекты и организационные единицы (см. рис. 4 – 5).

Событие служит для отображения возможных результатов выполнения функций и инициируют выполнение новых функций. Каждая функция должна быть инициирована событием и завершена событием. В каждую функцию не может входить более одной стрелки, инициирующей ее выполнение, и выходить не более одной стрелки, описывающей завершение ее выполнения.

Для описания ветвления используются операторы. Возможны разные соединения операторов и функций кроме двух: после события нельзя использовать операторы ИЛИ и исключающего ИЛИ, так как решение может приниматься только в процессе выполнения функции.

Для соединения событий и функций используются пунктирные линии, для соединения функций, информационных объектов и организационных единиц – сплошные.

Графические элементы на диаграмме eEPC могут располагаться в произвольном порядке, однако обычно выбирают направление слева направо или сверху вниз.

Диаграммы цепочки процессов (PCD – Process Chain Diagrams) являются разновидностью диаграмм eEPC. На этих диаграммах графические элементы распределяются по столбцам, что в некоторых случаях облегчает их читаемость. Такое представление лучше подходит при документировании, но неудобно при описании процессов с разветвленной логикой.

Пример диаграммы PCD для процесса обработки заказа приведен на рис. 6. Первый и второй столбцы диаграммы PCD отображают логическую последовательность выполнения процесса – они содержат события и функции, связанные между собой пунктирными стрелками и операторами.

Количество столбцов может быть расширено. Возможно добавление таких столбцов, как «Носитель», «Прикладная система» и «Ресурс» (заполняемых соответствующими

объектами) и «Экран», «Пакет», «Диалог» (заполняемых аналогично столбцу «Руководство»).

Диаграммы процесса могут быть использованы при анализе реальных бизнес-процессов, определении недостатков их организации или причин неэффективности. Такими недостатками могут быть дезинтеграция между ручной обработкой и обработкой с помощью информационных технологий или организационная дезинтеграция. Кроме того, проявляются лишние входы (процедурная избыточность данных) и задержки в выполнении.

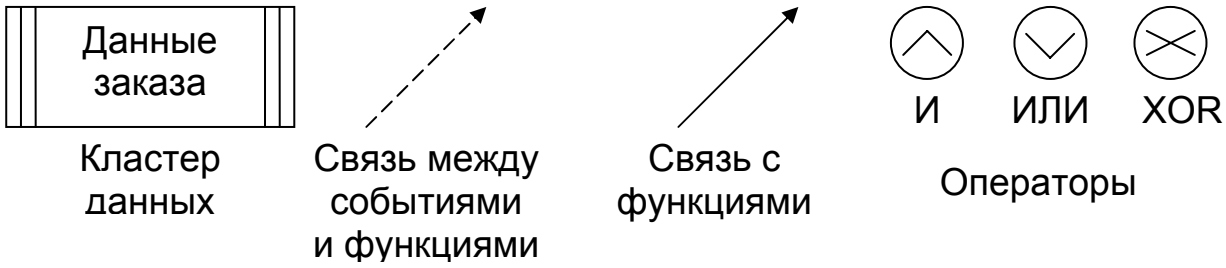


Рис. 4. Специфичные графические примитивы eEPC и PCD диаграмм

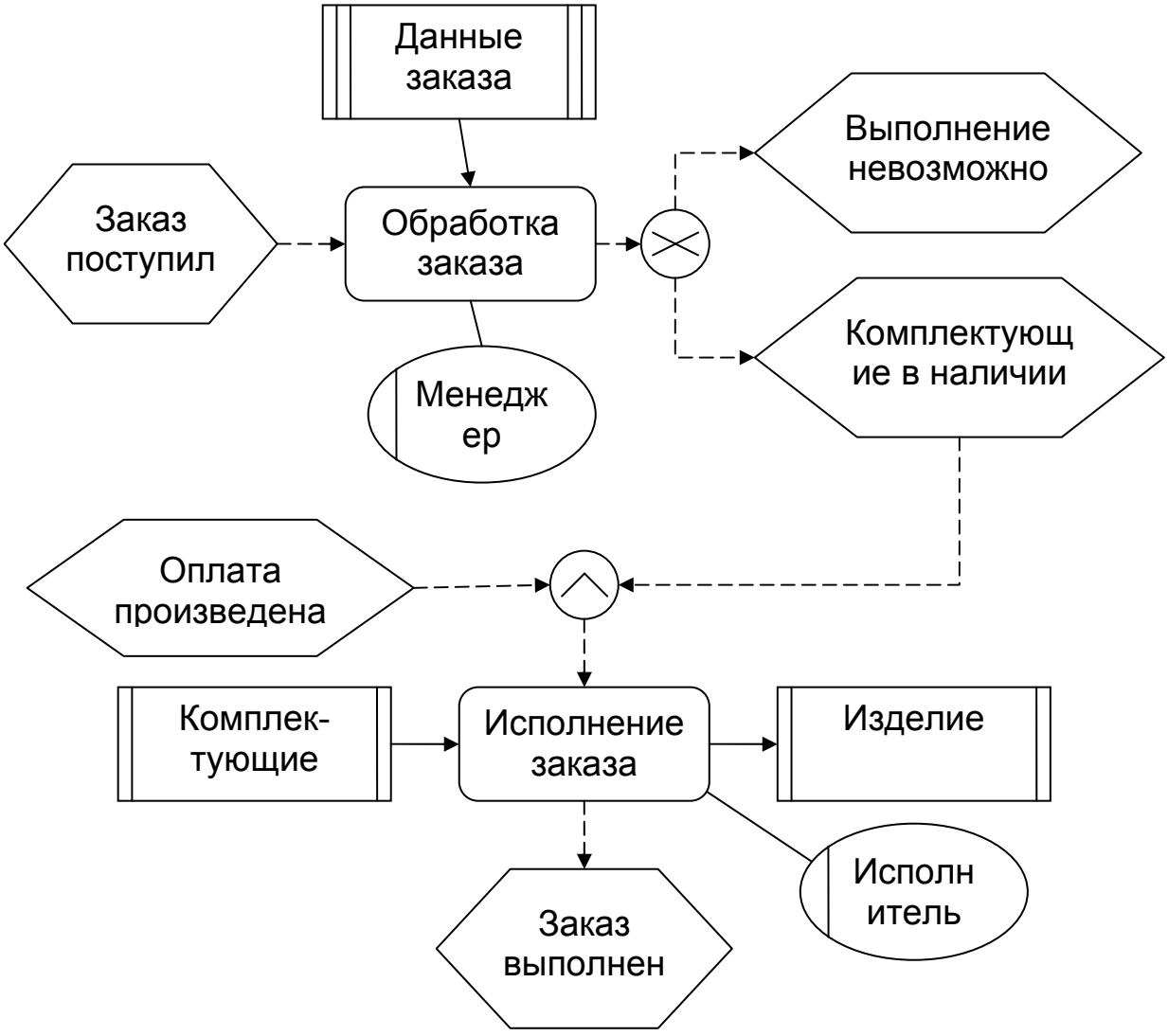


Рис. 5. Пример eEPC диаграммы

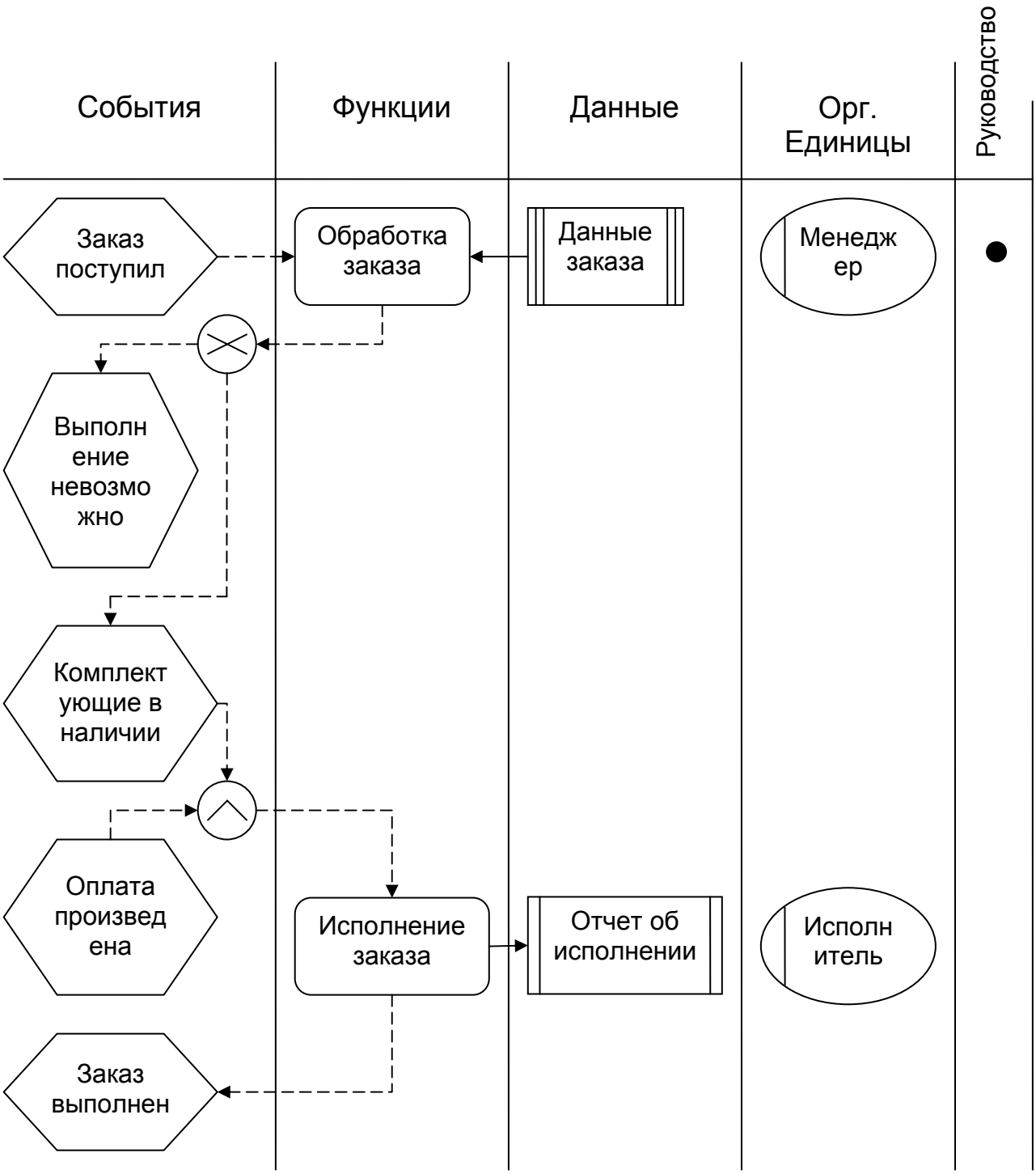


Рис. 6. Пример PCD диаграммы

1. Цели и задачи сбора и формализации требований к ПО мобильных устройств. Основные определения. Место в жизненном цикла ПО.
2. Источники информации о требованиях. Организация взаимодействия с Заказчиком. Способы и технологии проектирования требований к ПО.
3. Виды требований к ПО. Перечень требований по ГОСТ 34.602-89, ГОСТ 19.201-78
4. Методологии проектирования ПО мобильных устройств. Общие принципы. Технологии проектирования. CASE средства проектирования
5. Структурное и функциональное проектирование ПО. SADT.
6. Формализация требований к процессам и потокам данных. Описание требований к информационному обеспечению
7. Язык UML. Общие принципы проектирования. Диаграммы вариантов использования и классов
8. Язык UML. Разработка модели ПО для мобильных устройств, учитывающих требования Заказчика
9. Процессно-ориентированный подход. ARIS. BPMN

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

СОГЛАСОВАНО	**УТВЕРЖДАЮ**
Управление образовательных программ	Проректор по учебной работе
_____/ А.В. Дорошин /	_____/ Ф.В. Гречников /
"_____"_____ 20___ г.	"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

Методы проектирования и поддержки требований к программному

Цикл, в рамках которого происходит освоение модуля (дисциплины)	Профессиональный цикл
Часть цикла	Вариативная часть
Код учебного плана	230100.68
Факультет	Информатики
Кафедра	Информационных систем и технологий

Курс	5		
Семестр	A		
Лекции (СЛ)	18		
Семинарские и практические занятия (СП)	0		
Лабораторные занятия (СЛР)	36	Экзамен	
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	A
Самостоятельная работа (СРС)	54		
Всего (Всего с экзаменами)	108		

Наименование стандарта, на основании которого составлена рабочая программа:

230100 Информатика и вычислительная техника Приказ № 554 от 09.11.09

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

Иващенко А.В. _____
(подпись)

Заведующий кафедрой:

Прохоров С.А. _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

Информационных систем и технологий

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

способен к самостоятельному обучению новым методам исследования, к изменению научного и научно-производственного профиля своей профессиональной деятельности (ОК-2);использует на практике умения и навыки в организации исследовательских и проектных работ, в управлении коллективом (ОК-4);применять перспективные методы исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий (ПК-1);формировать технические задания и участвовать в разработке аппаратных и/или программных средств вычислительной техники (ПК-4);применять современные технологии разработки программных комплексов с использованием CASE-средств, контролировать качество разрабатываемых программных продуктов (ПК-6)

1.2 Цели и задачи изучения модуля (дисциплины)

Цель курса: Ознакомить будущих магистров с современными технологиями и методами сбора и формализации требований и проектирования ПО мобильных устройствЗадачи курса: Ознакомить студентов с основными теоретическими принципами проектирования ПО мобильных устройств, методами проектирования и формализации требований к ПО, методами актуализации требований в ходе жизненного цикла ПО

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

В результате освоения дисциплины студент должензнать основные принципы и методологии анализа и формализации требований к ПО, проектирования ПО и написания ТЗ на разработку ПО мобильных устройств, уметь применять применять перспективные методы исследования и решения профессиональных задач по разрабо
тке ПО мобильных устройств на основе знания мировых тенденций развития вычислительной техники и информационных технологий, владеть методологией сбора и формализации требований к ПО мобильных устройств

1.4 Связь с предшествующими модулями (дисциплинами)

Архитектура современных распределенных системТехнологии и инструментальные средства разработки ПО для мобильных устройств

1.5 Связь с последующими модулями (дисциплинами)

Менеджмент разработки ПОСовременные языки программирования и паттерны проектирования разработки программного обеспеченияТехнология разработки программного обеспечения

2 Содержание рабочей программы (модуля)

Семестр 1		

СЛ 0,1667 18 часов 0,5001 ЗЕТ	Активные 0	Цели и задачи сбора и формализации требований к ПО мобильных устройств. Основные определения. Место в жизненном цикла ПО
		Источники информации о требованиях. Организация взаимодействия с Заказчиком. Способы и технологии проектирования требований к ПО
		Виды требований к ПО. Перечень требований по ГОСТ 34.602-89, ГОСТ 19.201-78
		Методологии проектирования ПО мобильных устройств. Общие принципы. Технологии проектирования. CASE средства проектирования
		Структурное и функциональное проектирование ПО. SADT.
		Формализация требований к процессам и потокам данных. Описание требований к информационному обеспечению
		Язык UML. Общие принципы проектирования. Диаграммы вариантов использования и классов
		Язык UML. Разработка модели ПО для мобильных устройств, учитывающих требования Заказчика
		Процессно-ориентированный подход. ARIS. BPMN
	Интерактивные 0	
	Традиционные 0	
СП 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СЛР 0,3333 36 часов 0,9999 ЗЕТ	Активные 0	Разработка модели ПО по методологии SADT
		Разработка диаграмм потоков данных
		Разработка модели ПО на языке UML. Диаграммы вариантов использования
		Разработка модели ПО на языке UML. Диаграммы классов

		Разработка модели ПО на языке UML. Описание требований к поведению
		Разработка модели ПО на языке UML. Описание требований к взаимодействию и реализации
	Интерактивные 0	
	Традиционные 0	
КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СРС 0,5 54 часов 1,5 ЗЕТ	Активные 0	Исследование особенностей требований к мобильным устройствам
		Изучение типовых решений для мобильных устройств в сети Интернет
		История методологий проектирования ПО и анализ наилучших методов для мобильных устройств
		Психологические особенности разработки ПО для мобильных устройств
		Требования по эргономике и удобству пользовательского интерфейса
		Требования по безопасности мобильных устройств
		Виды нефункциональных требований и особенности их проектирования
		Зависимость конкурентоспособности и экономической привлекательности ПО мобильных устройств в зависимости от требований к нему
		Мобильные игры. Casual игры. Особенности привлечения покупателей и методы управления интересом
	Интерактивные 0	
	Традиционные 0	

3 Инновационные методы обучения

Использование интерактивной доски во время проведения лекционных занятийПроведение деловой игры

4 Технические средства и материальное обеспечение учебного процесса

Компьютеры, оргтехника, проектор;

5 Учебно-методическое обеспечение

5.1 Основная литература

1 Репин, В.В. Процессный подход к управлению [Текст] : моделирование бизнес-процессов / В. В. Репин, В. Г. Елиферов. - Изд. 5-е. - М. : РИА "Стандарты и качество", 2007. - 404 с.2 Леоненков, А.В. Самоучитель UML [Текст] : базы и банки данных / Александр Леоненков. - 2-е изд. - СПб. : БХВ-Петербург, 2006. - 427 с.3 Ларман, К. Применение UML 2.0 и шаблонов проектирования [Текст] : введ. в объект.-ориентир. анализ, проектирование и итератив. разработку : [пер. с англ.] / Крэг Ларман. - 3-е изд. - М. [и др.] : Вильямс, 2007.

5.2 Дополнительная литература

1 Рамбо, Джеймс. UML [Текст] : Спец. справ.: [Пер. с англ.] / Д. Рамбо, А. Якобсон, Г. Буч. - СПб. : Питер : Питер бук, 2002. - 652 с. - (Справочник)2 Вайсфельд, М. Объектно-ориентированный подход: Java, .Net, C++ [Текст] : пер. с англ. / Мэтт Вайсфельд. - 2-е изд. - М. : КУДИЦ-ОБРАЗ, 2005. - 336 с.3 Гранд, М. Шаблоны проектирования в Java [Текст] : кат. попул. шаблонов проектирования, проиллюстрир. при помощи UML : [пер. с англ.] / Марк Гранд. - М. : Новое знание, 2004. - 558 с.4 Ларман, К. Применение UML и шаблонов проектирования [Текст] : введ. в объект.-ориентир. анализ, проектирование и унифицир. процесс ИР : [пер. с англ.] / Крэг Ларман. - 2-е изд. - М. [и др.] : Вильямс, 2004. - 619 с.5 Вендров, А.М. Проектирование программного обеспечения экономических информационных систем [Текст] : [UML CASE : учеб. для вузов по специальностям "Прикладная информатика (по обл.)" и "Прикладная математика и информатика"] / А. М. Вендров. - 2-е изд., перераб. и доп. - М. : Финансы и статистика, 2005. - 543 с.6 Орлов, С.А. Технологии разработки программного обеспечения [Текст] : разраб. слож. програм. систем : [учеб. для вузов по специальности "Програм. обеспечение вычисл. техники и автоматизир.систем" направления подгот. дипломир. специалистов "Информатика
и вычисл. техника"] / С. А. Орлов. - СПб. и др. : Питер : Питер принт, 2004. - 526 с.

5.3 Электронные источники и интернет ресурсы

1 UML. Унифицированный язык моделирования [Электронный ресурс] : версия 1.0. - Электрон. текстовые дан. - М. : Центр ЮрИнфоР, 2003. - 1 эл. опт. диск (CD-ROM).2 Куприянов, А.В. Технологии проектирования программных комплексов [Электронный ресурс] : [учеб. мультимедиа комплекс] / Куприянов А. В. ; Самар. гос. аэрокосм. ун-т им. С. П. Королева. - Электрон. текстовые и граф. дан. (1,46 Мбайт). - Самара : СГАУ, 2006.

5.4 Методические указания и рекомендации

1 Проектирование автоматизированных информационных систем по методологии SADT [Текст] : метод. указания к лаб. работам / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева ; [сост. В. П. Дерябкин, А. В. Иващенко]. - Са
мара : СГАУ, 2004. - 49 с.